



US Army Corps
of Engineers®

Simulation of Coastal Storm Surge and Rainfall Flooding Scenarios at Camp Lejeune with GSSHA

*by Nawa Raj Pradhan, Charles W. Downer,
Samantha N. Sinclair, and Clay LaHatte*

PURPOSE: This Coastal and Hydraulics Engineering Technical Note (CHETN) describes the development of a computer model to examine flooding scenarios with pre-defined storm surge and extreme rainfall due to tropical systems, using the Gridded Surface Subsurface Hydrologic Analysis (GSSHA) numerical modeling system (Downer et al. 2005). This model serves as a platform for future deployment of GSSHA models on large basins in this type of setting with enhanced Military Hydrologic Simulator capabilities.

BACKGROUND: The military faces multiple environmental hazards around the world. Simulations of various environmental conditions can be used to help troops understand, assess, and prepare for potential hazards. Hydrologic simulations can be used to assess water-related hazards, such as flooding, high currents, erosion, and drought.

Recognition of this need has led to the development of the Military Hydrologic Simulator software tool. The Military Hydrologic Simulator is being developed as an interface to facilitate data input, model runs, and simulation output visualizations for large spatial scales. The performance of the long-term simulation capabilities in this study will help the Hydrologic Simulator to manage large temporal scales, and in operational mode, to inform flood hazards.

The Camp Lejeune GSSHA watershed model discussed in this document is the first in a series of pre-defined scenarios that provide a computational test bed for the Military Hydrologic Simulator. For the purposes of defining hazards, seven different broad operating environments were identified: coastal, riverine, tropical, high mountain, high plains, arid upland, and humid upland. Broadly speaking, Camp Lejeune is in a coastal region and represents environmental hazards found in coastal areas. The main hazards are flooding due to storm surge, unique along the coast, but include hazards also found in other environments such as the extreme rainfall typical of tropical regions, and riverine flooding. As intended here, the *coastal* environment means not just the coastline but a general environmental region where low-relief areas are subjected to extensive flooding due to storm surge, intense rainfall, and subsequent river flooding due to tidal influence and excess runoff. Operationally, troops would need to be aware of all potential flood hazards in the operating environment, not just storm surge immediately along the coast.

This scenario demonstrates the capability of deploying the GSSHA model on a large basin in a coastal environment.

Study Area. The Marine Corps Base Camp Lejeune is located in coastal North Carolina, intersected by the New River. The study boundary, shown in Figure 1, encompasses the watershed

for the New River, as well as inter-connected coastal areas, and covers a total area of 1449 square kilometers including the catchment area and the extended coastal area.



Figure 1. Camp Lejeune study area with stream network.

Data Requirements. Hydrologic simulations require spatial and temporal input data. When running simulations, it is imperative to use data that most accurately represent the region of interest. GSSHA model inputs were developed with the Watershed Modeling System (WMS), a graphical user interface that has tools and algorithms that have been used for more than 15 years to create physics-based watershed models.

Land Use Data. Spatial data products depicting land use and land cover (LULC) support hydrologic model parameterization. The 30-meter (m) resolution LULC digital dataset of the year 2011 was downloaded from the National Land Cover Database web site (<https://www.mrlc.gov/>). Figure 2 shows the land use classification in the study area. As shown in the figure, the watershed is largely undeveloped, with forest and wetlands being the dominant land uses. Less than 15% of the watershed is listed as developed. LULC classification was used to assign overland flow roughness parameter values, as shown in Table 1.

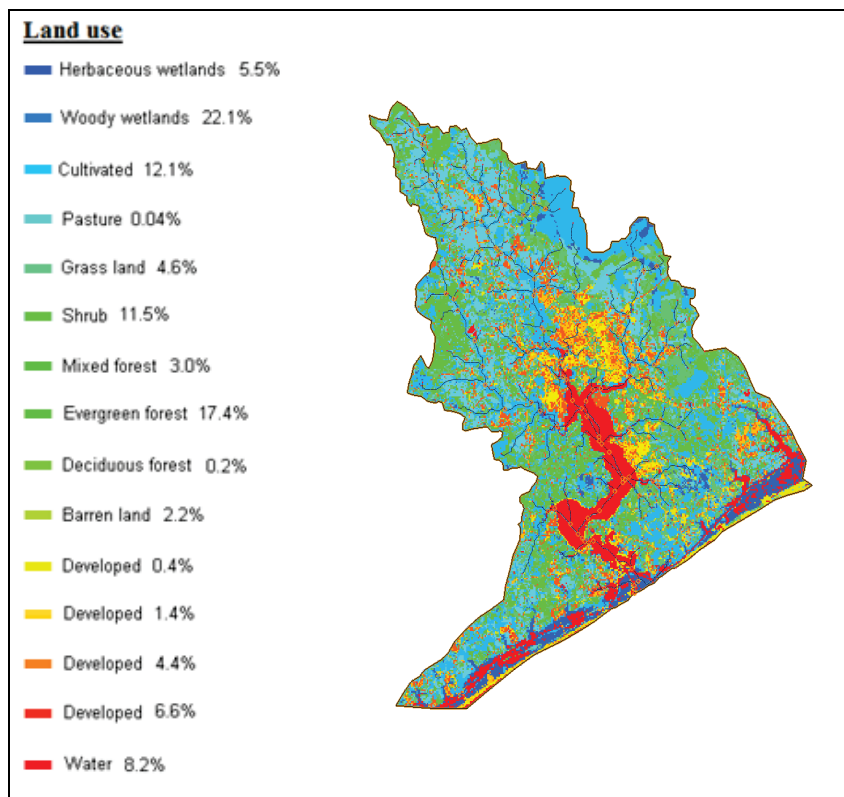


Figure 2. Camp Lejeune land use index map employed in the GSSHA model.

Table 1. Overland flow roughness values.		
ID	Land Cover Type	Roughness Value
11	Water	0.012500
21	Developed, open space	0.012500
22	Developed, low intensity	0.062500
23	Developed, medium intensity	0.001250
24	Developed, high intensity	0.017125
31	Barren	0.062500
41	Deciduous Forest	0.240000
42	Evergreen Forest	0.240000
43	Mixed Forest	0.240000
52	Shrub/Scrub	0.187500
71	Grassland	0.162500
81	Pasture	0.312500
82	Cultivated Crops	0.125000
90	Woody Wetland	0.237500
95	Emergent Herbaceous Wetlands	0.250000

Soils Data. Digitized soil texture data at 30 m resolution were obtained from the Soil Survey Geographic Database (SSURGO) web-site https://www.nrcs.usda.gov/wps/portal/nrcs/detailfull/soils/survey/?cid=nrcs142p2_053369. Soils are predominantly sandy loams, as shown in Figure 3. Soil type classification was used to assign the infiltration parameters for the Green and Ampt with Redistribution model. Parameter values used in the model are shown in Table 2.

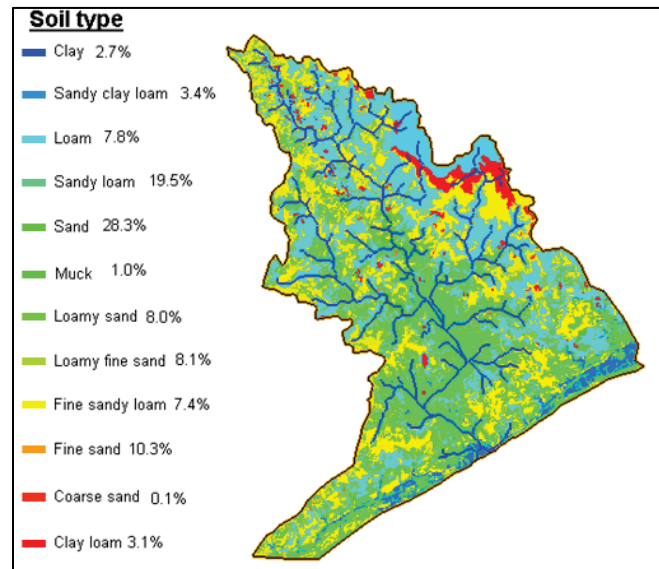


Figure 3. Camp Lejeune soil type index map employed in GSSHA model.

Table 2. Green and Ampt infiltration model parameter values for corresponding soil index map.

ID	Soil Type	Hyd Cond	Cap Head	Porosity	Pore Distribution	Res Saturation	Field Capacity	Wilting Point
1	Clay Loam	0.200	20.88	0.464	0.242	0.075	0.318	0.197
2	Coarse Sand	23.560	2.000	0.400	0.700	0.019	0.085	0.029
3	Fine Sand	1.000	6.000	0.425	0.630	0.022	0.110	0.029
4	Fine Sandy Loam	1.000	14.000	0.425	0.350	0.045	0.220	0.045
5	Loamy Fine Sand	3.000	8.000	0.425	0.520	0.040	0.140	0.040
6	Loamy Sand	5.980	6.130	0.437	0.553	0.035	0.125	0.055
7	Muck	0.500	18.000	0.440	0.230	0.080	0.380	0.185
8	Sand	23.56	4.950	0.437	0.694	0.080	0.091	0.033
9	Sandy Loam	2.180	11.010	0.453	0.378	0.041	0.207	0.095
11	Loam	1.320	8.890	0.463	0.252	0.027	0.270	0.117
15	Sandy Clay Loam	0.300	21.850	0.398	0.319	0.068	0.255	0.148
18	Clay	0.060	31.630	0.475	0.165	0.090	0.396	0.272

Digital Elevation Model (DEM) and Bathymetry. The DEM used for the study is a raster dataset from the Shuttle Radar Topography Mission (SRTM) with a 10 m resolution. Elevation data depicting the bathymetry of the estuary and lower channel were not included in the DEM data and thus had to be obtained separately and merged with the DEM dataset to create a complete elevation model to build the GSSHA model of the area. The bathymetry data were obtained as a point shape file from a pre-existing ADvanced CIRCulation (ADCIRC) hydrodynamic model (<https://adcirc.org/home/documentation/adcirc-related-publications/>) of the area. The following steps were followed in the processing of the point bathymetry data for the Lejeune GSSHA model:

1. The bathymetry points were converted to a Triangulated Irregular Network (TIN).
2. The TIN was clipped to the channel area.
3. The clipped TIN was converted to a DEM raster.
4. The resulting bathymetry DEM was merged into the overall SRTM DEM to obtain a complete elevation model, with estuary bathymetry, for Lejeune.
5. For computational efficiency, the 10 m resolution elevation model was re-sampled to 150 m resolution to represent the Lejeune GSSHA elevation model as shown in Figure 4.

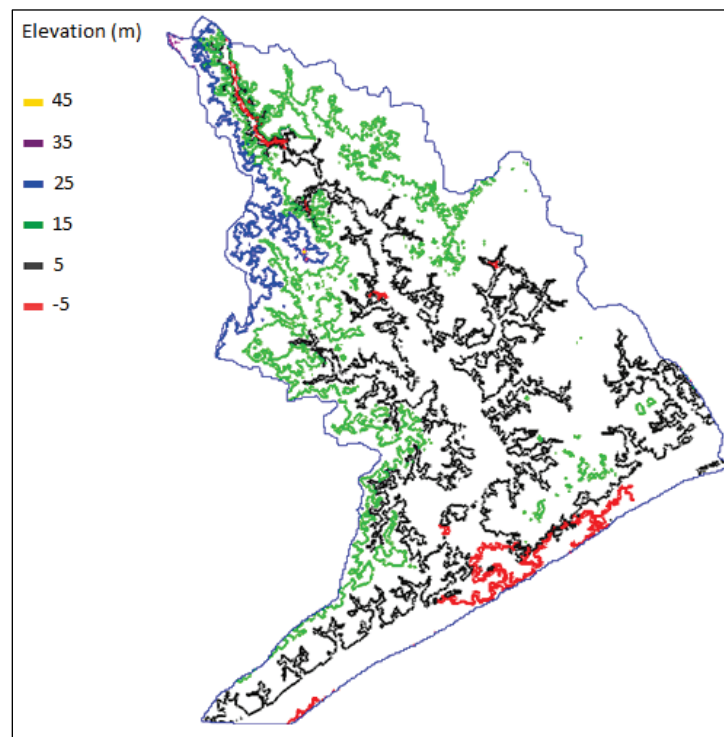


Figure 4. Camp Lejeune elevation map employed in the GSSHA model.

For the lower channel portion of the GSSHA stream network, channel cross sections were assigned for each stream segment based on bathymetry of the stream channel segment as shown in Figure 5 and Figure 6.

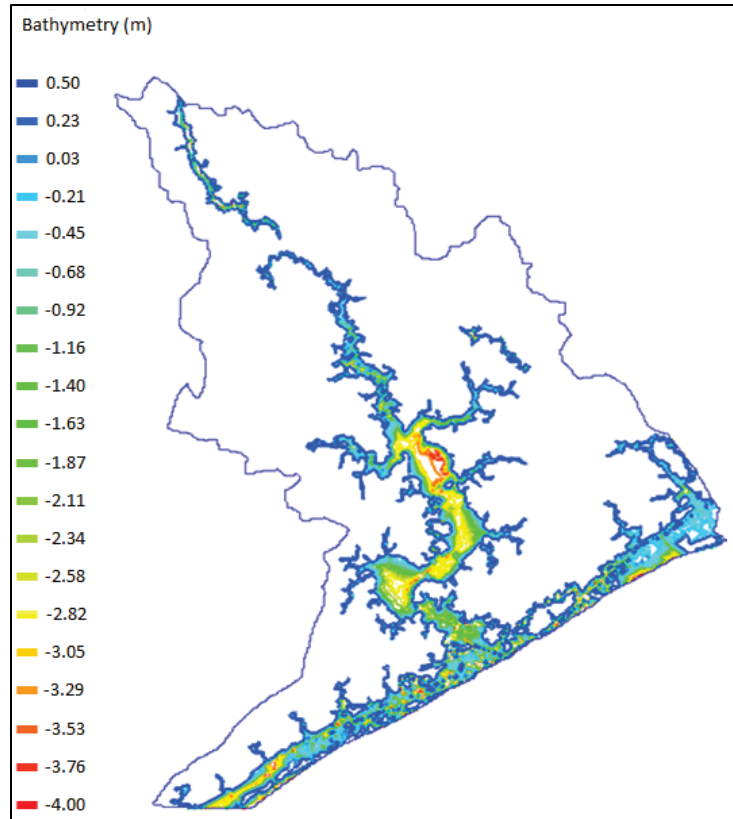


Figure 5. Camp Lejeune channel bathymetry employed in the GSSHA model.

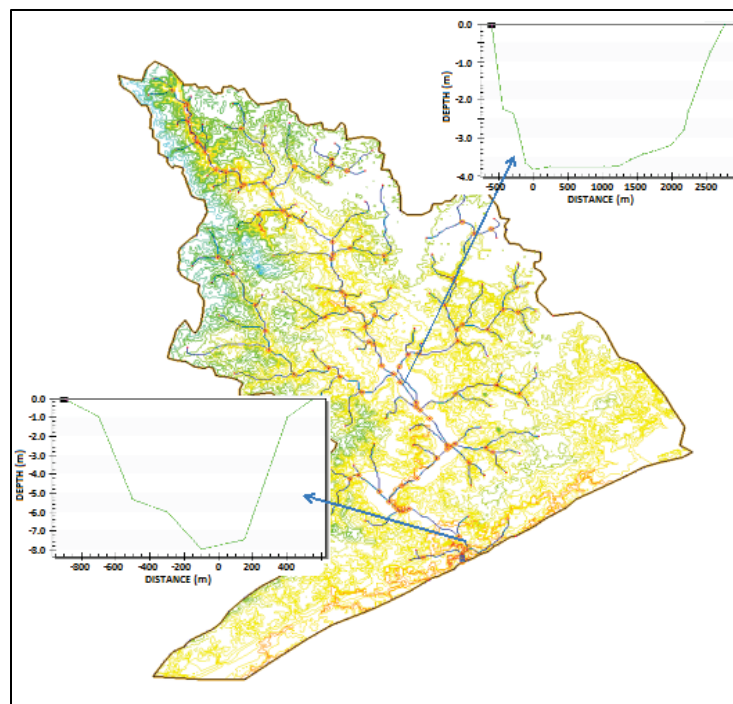


Figure 6. Typical channel cross sections.

CAMP LEJEUNE GSSHA MODEL: One-dimensional (1D) infiltration, two-dimensional (2D) overland flow, and 1D stream flow simulate the runoff generation process, transport process of runoff to a stream, and channel transport process to catchment outlet, respectively. The Green and Ampt infiltration model with soil moisture redistribution scheme (Ogden and Saghafian 1997) was employed. The runoff generated in each pixel is transferred to the lower head cell through a 2D finite volume numerical solution of the diffusive wave equation. The surface runoff is coupled to channel routing where lateral inflow from surface runoff grid cells is numerically routed with a 1D finite volume solution of the diffusive wave equation. Overbank stream flows are allowed to flood back onto the overland flow plane. A tidal surge boundary is used to simulate the effects of storm surge flooding within the model domain.

Infiltration Model. The rate of infiltration is governed by soil physical properties, which vary with the soil type or texture. The soil map in Figure 3 was employed to determine the soil physical properties. Following are the soil physical parameters in the infiltration model (cm = centimeter; h = hour):

- a. effective porosity ($\text{cm}^3 \text{cm}^{-3}$)
- b. field capacity ($\text{cm}^3 \text{cm}^{-3}$)
- c. wilting point ($\text{cm}^3 \text{cm}^{-3}$)
- d. residual saturation ($\text{cm}^3 \text{cm}^{-3}$)
- e. pore size distribution (cm cm^{-1})
- f. saturated hydraulic conductivity (cm h^{-1})
- g. wetting front suction head (cm)

Apart from hydraulic conductivity, the parameters in Table 2 were estimated from literature values (Rawls and Brakensiek 1983).

Routing Model. GSSHA employs explicit finite volume schemes to route water for 1D channels and 2D overland flow, where flows are computed based on heads and volumes are updated based on the computed flows. The friction slope between one grid cell and its neighbors is calculated as the difference in water surface elevations divided by the grid size. Compared with the kinematic wave approach, this diffusive wave approach allows GSSHA to route water through pits or depressions, and regions of adverse slope, as well as compute backwater effects. The Manning formula relates flow depth to discharge. Hydrological models implement the Manning equation to relate surface roughness to flow rate, in which case the hydraulic roughness is represented by the Manning's roughness coefficient, n . For each land use type shown in Figure 2, Manning's roughness coefficient (n) values were defined as per GSSHA wiki, <http://www.gsshawiki.com>), shown in Table 1.

Channel Geometry. Channel geometry was obtained from the bathymetry as shown in Figure 5. Typical cross sections in the main channel are as shown in Figure 6.

Initial Soil Moisture. A physics-based, spatially distributed hydrologic model like GSSHA is sensitive to the initial state of soil moisture. Satellite imagery can be a good source for acquiring soil moisture data in a data-sparse area (Pradhan et al. 2012). The Normal Difference Vegetation

Index was employed to derive crop coefficients as defined by Rocha et al. (2010). A relationship between soil moisture and crop coefficients, as defined by Pradhan et al. (2012), was employed to estimate the soil moisture which is shown in Figure 7.

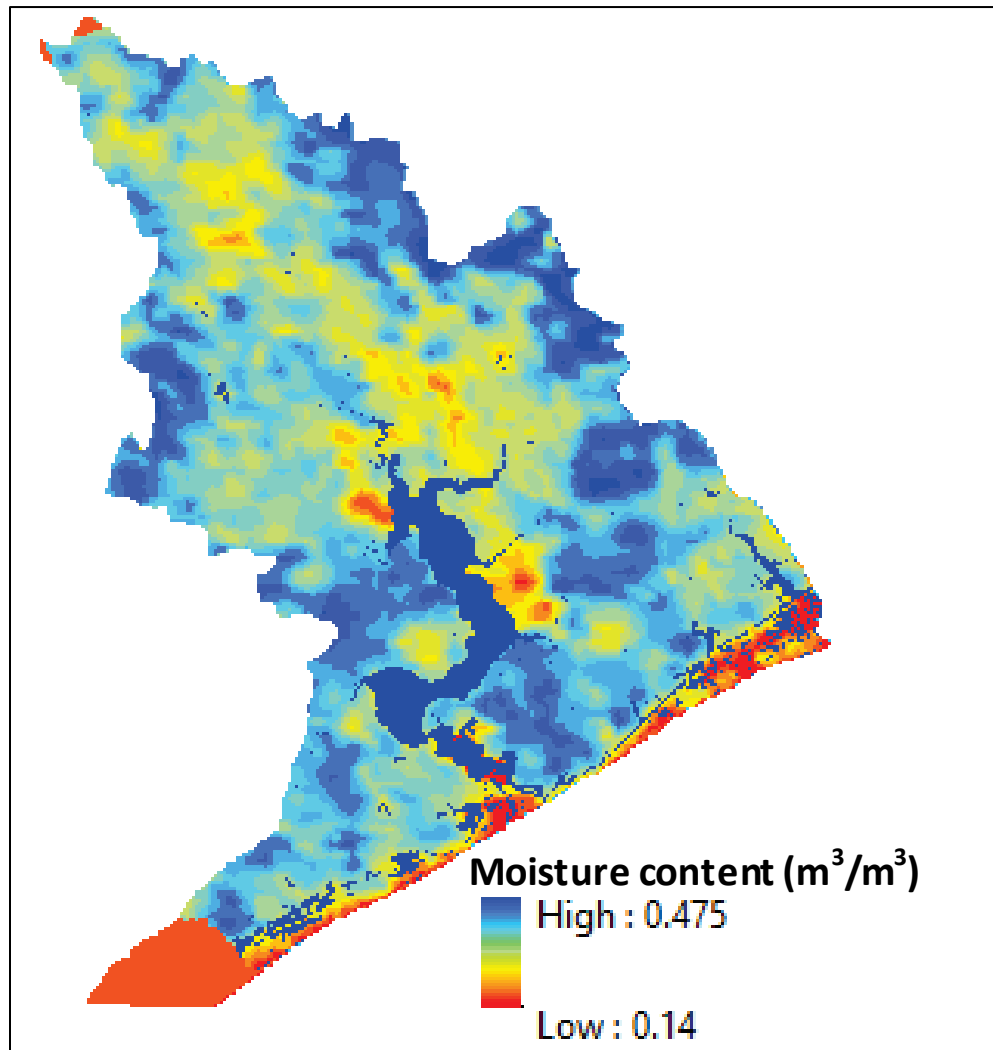


Figure 7. Camp Lejeune initial soil moisture employed in the GSSHA model.

GSSHA Control Project Cards: Table 3 lists the GSSHA project input output cards, values, and a description of the card meaning used in the Lejeune model development.

Table 3. Project file control cards.		
Card	Input Value	Description
GRIDSIZ	150	grid resolution (m)
ROWS	424	number of rows
COLS	339	number of columns
TOT_TIME	1440	total simulation time (min)
TIMESTEP	20	simulation time step (s)
OUTROW	344	outlet row address
OUTCOL	203	outlet column address
OUTSLOPE	0.001	channel slope at outlet
MAP_FREQ	30	time interval of map output (min)
HYD_FREQ	30	output time interval of channel discharge (min)
MAP_TYPE	1	map output in a generic WMS format
OVERBANK_FLOW	—	allow channel overbank flow
DIFFUSIVE_WAVE	—	channel routing method
OVERTYPE	ADE	overland flow solver – Alternating Direction Explicit
INF_REDIST		infiltration method – Green and Ampt with soil moisture redistribution
RAIN_INV_DISTANCE		rainfall distribution - inverse distance weighting
WATERSHED_MASK	Lejeune.msk	watershed boundary file (input file)
ELEVATION	Lejeune.ele	elevation file (input file)
PRECIP_FILE	Lejeune.gag	precipitation file(input file)
MAPPING_TABLE	Lejeune.cmt	process parametric values linker(input file)
FLINE	Lejeune.map	meant for WMS visualization
CHANNEL_INPUT	Lejeune.cif	stream network description (input file)
STREAM_CELL	Lejeune.gst	streams link to the grid (input file)
IN_HYD_LOCATION	Lejeune.ihl	channel internal output locations (input file)
OUT_HYD_LOCATION	Lejeune.ohl	time series of discharge output (output file)
DEPTH	Lejeune.dep	maps of overland depth output file (output file)
SURF_MOIST	Lejeune.sur	maps of soil moisture output file (output file)
MOISTURE	Lejeune.mst	map of initial soil moisture content (input file)
OUTLET_HYDRO	Lejeune.otl	discharge at the outlet (output file)
SUMMARY	Lejeune.sum	simulation summary report (output file)
OV_BOUNDARY		indicates special overland boundary condition exists
XYBDYINPUT_OVDEPTHINTERP	Lejeune_xyts.txt	storm surge time series (output file)

SIMULATIONS: An extreme event-based return period scenario and an actual hydro-meteorological data-based long-term scenario were deployed in the simulation.

Return Period Rainfall Event Simulation. The 24-hour 100-year return period rainfall event as defined in Hershfield (1961) was employed in the GSSHA model as rainfall forcing. U.S. Department of Agriculture Soil Conservation Service Type 1 distribution was used for obtaining hourly rainfall from the 24-hours-accumulated 100-year return period rainfall. A precipitation event of this magnitude might be expected to occur in the context of a tropical storm, which would have an associated storm surge, as discussed below.

Storm Surge. Figure 8 shows 10 years of water level at Wrightsville Beach, NC ([station ID # 8658163), <https://tidesandcurrents.noaa.gov/>. Wrightsville Beach station is one of the closest tides/water level measurement station to Camp Lejeune monitored by National Oceanic and Atmospheric Administration (NOAA). Figure 8 shows that maximum surge level over a span of 10 years is approximately 1.5 m, which is not that critical when compared to other historical surges. Hurricane Katrina in 2005 produced a maximum storm surge of more than 8 m in southern Mississippi, with a storm surge height of 8.5 m in Pass Christian (Knabb et al. 2005). Typhoon Haiyan produced a maximum storm surge of approximately 8 m in Leyte Gulf in Philippines (Lee and Kim 2015). Therefore, storm surge similar to the maximum storm surge of Katrina and Haiyan was hypothetically employed in the coastline of Camp Lejeune to provide a case of potential flood hazard that could occur relatively far inland and not just along the coast line. The following paragraphs in this section explain the details of this purposefully exaggerated potential flood hazard.

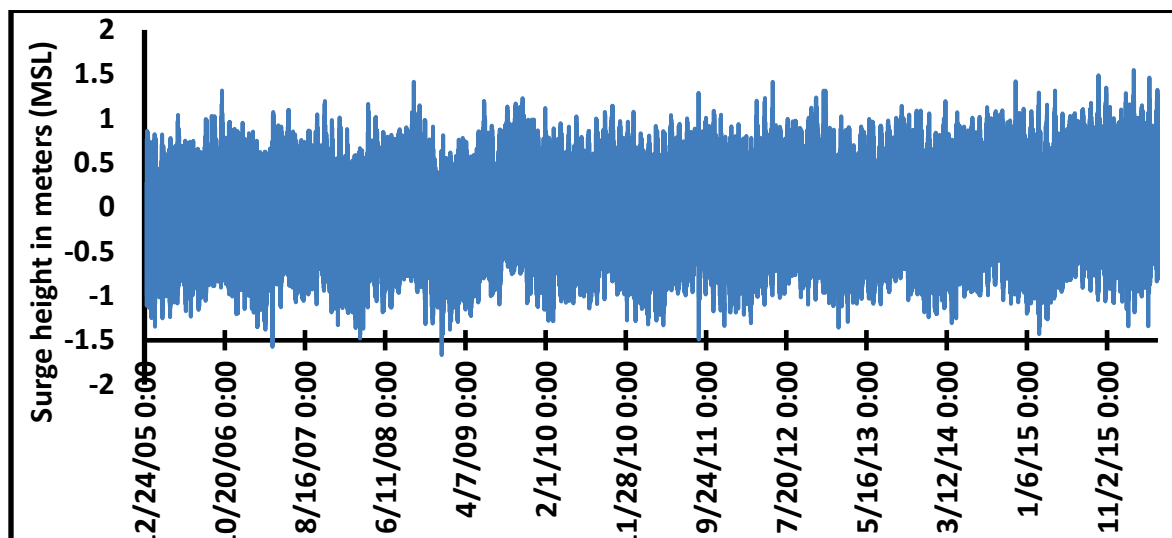


Figure 8. Record of 10 years water level at National Oceanic and Atmospheric Administration Wrightsville Beach station 8658163.

The storm surge was simulated with a time-varying stage boundary condition for the entire coastal boundary, as shown by the red line in Figure 9, to develop the Lejeune sea surge inland inundation model. The storm surge boundary cells, as shown by the red line in Figure 9, were allocated with time series of surge wave. Two arbitrary stations, as shown by blue stars in Figure 9, at Universal Transvers Mercator (UTM) Zone 18 north coordinates as listed in Table 4, were set as surge

stations. Table 4 lists two hypothetical time series of surge values, similar in magnitude of Hurricane Katrina surge at Pass Christian in Mississippi (Knabb et al. 2005) and Typhoon Haiyan surge at Leyte Gulf in Philippines (Lee and Kim 2013). The surge values of these locations are interpolated during the simulation for the entire storm surge boundary (in red) in Figure 9.

Table 4. Time series of tidal surge.

Time (min)	Surge Height (m)	
	Point 1 UTM Zone 18n, 3821719.9m N, 282336.8m E	Point 2 UTM Zone 18n, 3825498.7m N, 288888.7m E
0	2.0	6.5
30	4.5	8.0
60	2.5	6.5

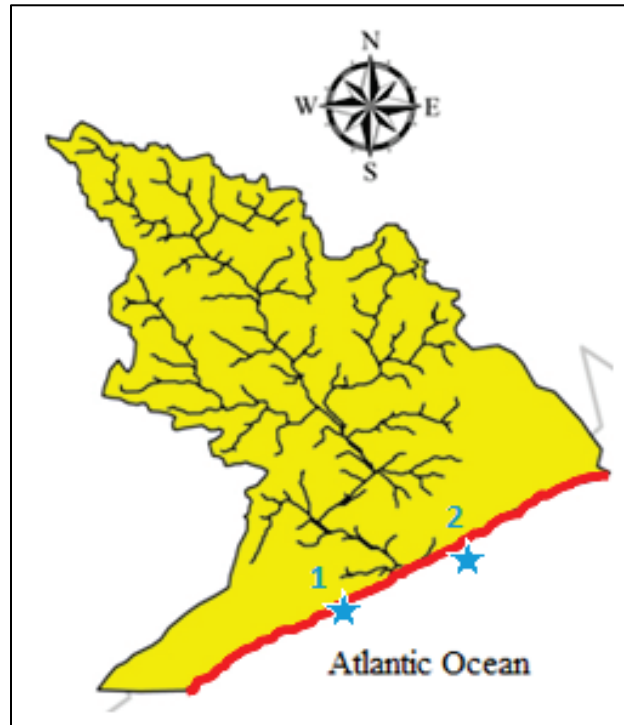


Figure 9. Time-varying boundary used in GSSHA to represent storm surge.

Return Period Simulation Results. Figure 10 shows simulation of the inland inundation due to sea surge and rainfall. The three panels show the initial, building, and peak inundations that occurred during the simulated event. In Figure 10, red indicates lower stages whereas blue indicates higher stages. Lower stages over the domain are replaced by higher stages at the model boundary, which propagate inland as the simulation progresses.

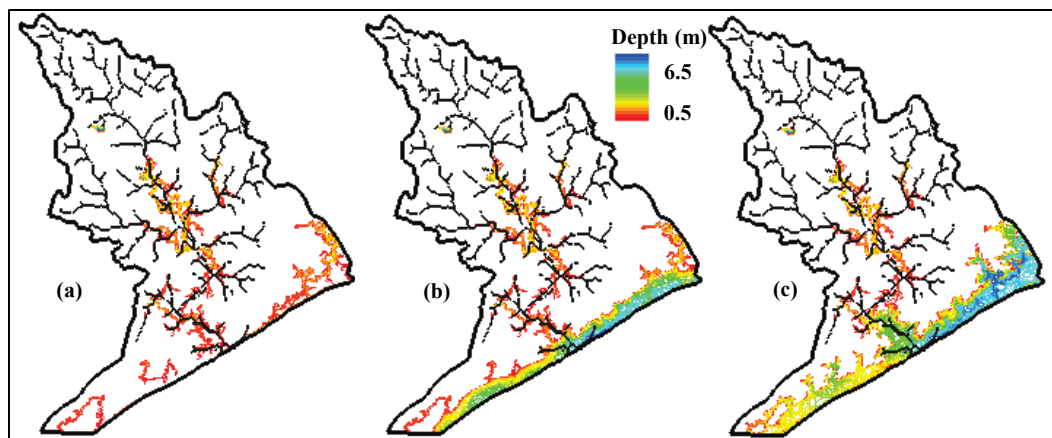


Figure 10. Simulation of the inland inundation due to sea surge and rain: (a) initial stage (b) as surge builds up (c) peak of surge.

Figure 11 shows the simulated soil moisture conditions at the peak and the end of the simulated return period event. Sea-surge boundary condition was absent.

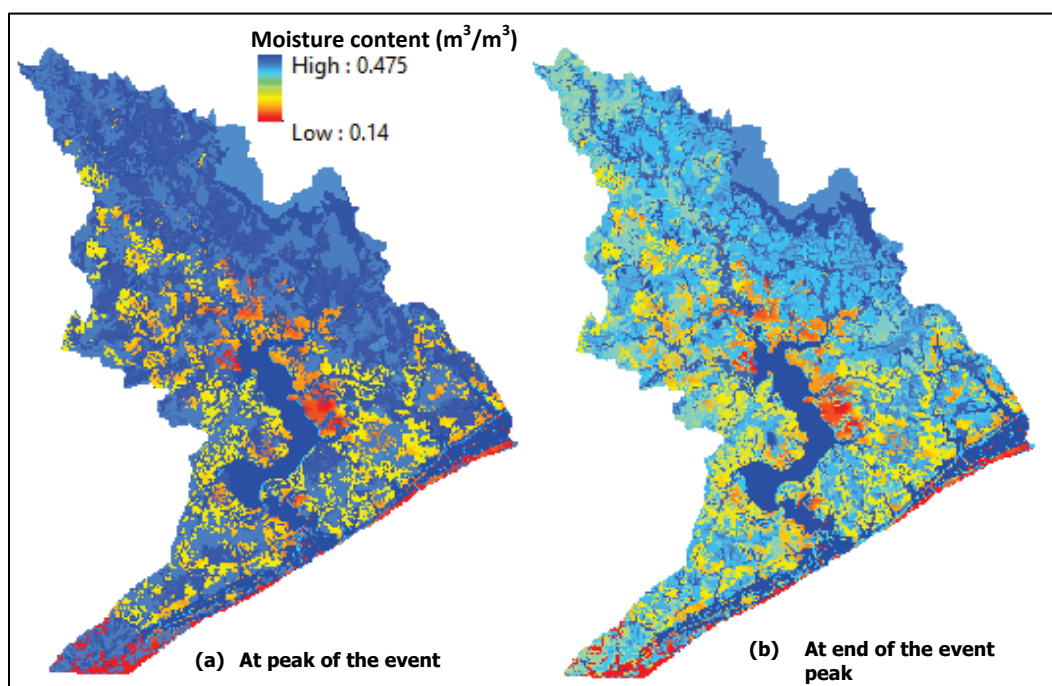


Figure 11. Simulated soil moisture showing the simulated soil moisture conditions at the peak and the end of the simulated return period event. Sea-surge boundary condition was absent.

Long-Term Simulations. The model was developed to be able to, with the addition of a few extra parameters, simulate multi-event precipitation, surge, and hydrometeorological (HMET) data over a record of 10 years from 2006 to 2016. For a long-term multi-event simulation, evapotranspiration plays a significant role in the water balance. Therefore, evapotranspiration was turned on for this long-term simulation, which required additional parameters as shown in Table 5.

During this period, several significant storms impacted coastal North Carolina, including Tropical Storm Ernesto in 2006, Hurricane Earl in 2010, Hurricane Irene in 2011, and Hurricane Sandy in 2012. Surge data associated with these events, which are included in Figure 8, were incorporated in the long-term simulation.

Long-Term Data Acquisition. Surge data were acquired from the Center for Operational Oceanographic Products and Services data portal on the NOAA Tides and Currents webpage (<https://opendap.co-ops.nos.noaa.gov/axis/>). Verified hourly surge data from a single station in Wrightsville Beach, NC, located at 34.21N, -77.7867W, was downloaded in reference to the North American Vertical Datum 1988 (NAVD88) and Greenwich Mean Time. Microsoft Excel was used to properly format the time and date stamp associated with each data point and to convert water level measurements to meters.

Precipitation data were acquired from the NOAA National Climatic Data Center webpage, <https://gis.ncdc.noaa.gov/maps/ncei/cdo/hourly>. Hourly precipitation data from a single station in Jacksonville, NC, located at 34.708N, -77.440W, was downloaded and processed using Excel to format the time and date stamps and to convert precipitation measurements to millimeters. The Time Series Editor, an editing software packaged within the WMS suite, was used to break the precipitation data into individual rainfall events, separating subsequent events by a minimum of 24 hours.

HMET data collected at the New River Marine Corps Air Station were acquired from the U.S. Air Force. Data processing was conducted to format the time and date stamps and convert barometric pressure (in Hg), relative humidity (%), total sky cover (%), wind speed (knots), temperature (F), direct radiation (Watt [W] m⁻²), and global radiation (W m⁻²) into the correct format with GSSHA requiring hourly values of HMET data. Flags are entered for any missing data associated with each hour, and GSSHA computes missing data as described on the GSSHA wiki, https://gsshawiki.com/Continuous:Hydrometeorological_Data). A Python code script was developed to ensure each parameter was correctly formatted and that, when necessary, missing data flags were inserted into the data. The following steps outline how to change the corresponding Python script (located in the Appendix) to customize it for other data of interest.

1. Copy and paste the Python script named “Processing_HMET_for_GSSHA.py” into a directory that contains the raw HMET data that requires reformatting.
2. Open the script in an editor to customize the code.
3. Scroll to the very bottom of the script to the section labeled “MAIN CODE BELOW.”
4. Change the variable “ENTER_YOUR_RAW_HMET_DATA_HERE.csv” to the name of your CSV file containing raw HMET data. This file should be comma delimited, and the date and time associated with the data must be listed in the first two columns.
5. Save and run the Python script.
6. Check the output to ensure the data are formatted correctly. Output data will be space delimited; have a date broken into year, month, and day; time converted into hours; temperature converted into degrees Fahrenheit; and sky cover converted into a percentage.

An example of raw input HMET data and formatted output data is presented in Tables 5 and 6.

Table 5. Example of raw hourly HMET data.								
Date	Hour	Barometric Pressure (in)	Relative Humidity (%)	Sky Cover (8ths)	Wind Speed (KTs)	Temp (C)	Direct Radiation (W/M*M)	Global Radiation (W/M*M)
19730101	0000z	30.09	93.8	8	4	18	0	0
19730101	0100z	30.09	88.1	8	2	21	0	0
19730101	0200z	30.09	88.1	8	4	21	0	0
19730101	0300z	30.09	93.8	8	2	18	0	0
19730101	0400z	30.09	100	8	4	17	0	0
19730101	0500z	30.09	100	8	4	17	0	0
19730101	0600z	30.12	100	8	6	17	0	0
19730101	0700z	30.09	100	8	2	16	0	0
19730101	0800z	30.09	100	8	0	17	0	0

Table 6. Example of correctly formatted hourly HMET data. Note: the data listed here are the same shown in Table 5.										
Year	Month	Day	Hour	Barometric Pressure (in.)	Relative Humidity (%)	Sky Cover (%)	Wind Speed (kts)	Temp (F)	Direct Radiation (W/M*M)	Global Radiation (W/M*M)
1973	1	1	0	30.09	94	100	4	64	0	0
1973	1	1	1	30.09	88	100	2	70	0	0
1973	1	1	2	30.09	88	100	4	70	0	0
1973	1	1	3	30.09	94	100	2	64	0	0
1973	1	1	4	30.09	100	100	4	63	0	0
1973	1	1	5	30.09	100	100	4	63	0	0
1973	1	1	6	30.12	100	100	6	63	0	0
1973	1	1	7	30.09	100	100	2	61	0	0
1973	1	1	8	30.09	100	100	0	63	0	0

Long-Term Parameterization. Additional model development in the form of specifying additional inputs and parameters was required to adapt the single event model to a long-term simulation. The Penman method was used to compute evapotranspiration. The active soil moisture was set to 0.5 m for soil moisture computations. An evapotranspiration mapping table was generated (see Table 7), the values for which were obtained from GSSHA wiki, <http://www.gsshawiki.com>. An overland boundary card was added to the project file to incorporate the surge data into the model

Table 7. Evapotranspiration mapping table values. Note: land-surface albedo and vegetation radiation coefficient are unit-less values.							
ID	11	21	22	23	24	31	41
Description	Water	Developed, Open Space	Developed, Low Intensity	Developed, Medium Intensity	Developed, High Intensity	Barren Land (Rock/Sand/ Clay)	Deciduous Forest
Land-Surface Albedo	0.15	0.15	0.15	0.15	0.15	0.15	0.2
Vegetation Height (m)	0.08	0.04	0.04	0.04	0.04	0.01	17
Vegetation Radiation Coefficient	0.7	0.2	0.2	0.2	0.2	0.2	0.15
Canopy Stomatal Resistance (s/m)	20	70	70	70	70	70	100

ID	42	43	52	71	81	82	90	95
Description	Evergreen Forest	Mixed Forest	Shrub/Scrub	Grassland/ Herbaceous	Cultivated Crops	Cultivated Crops	Woody Wetlands	Emergent Herbaceous Wetlands
Land-Surface Albedo	0.18	0.18	0.15	0.16	0.2	0.2	0.2	0.2
Vegetation Height (m)	5	10	13.5	6	9	15	15	15
Vegetation Radiation Coefficient	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15
Canopy Stomatal Resistance (s/m)	100	100	100	100	60	40	40	40

Long-Term Simulation Results. The long-term simulation model was run continuously for the period of 2006, during which time 24 individual precipitation events occurred. An output hydrograph was generated, which depicts a larger storm in mid-2006, likely a product of the “Mother’s Day Storm” in North Carolina in May 2006 (see Figure 11).

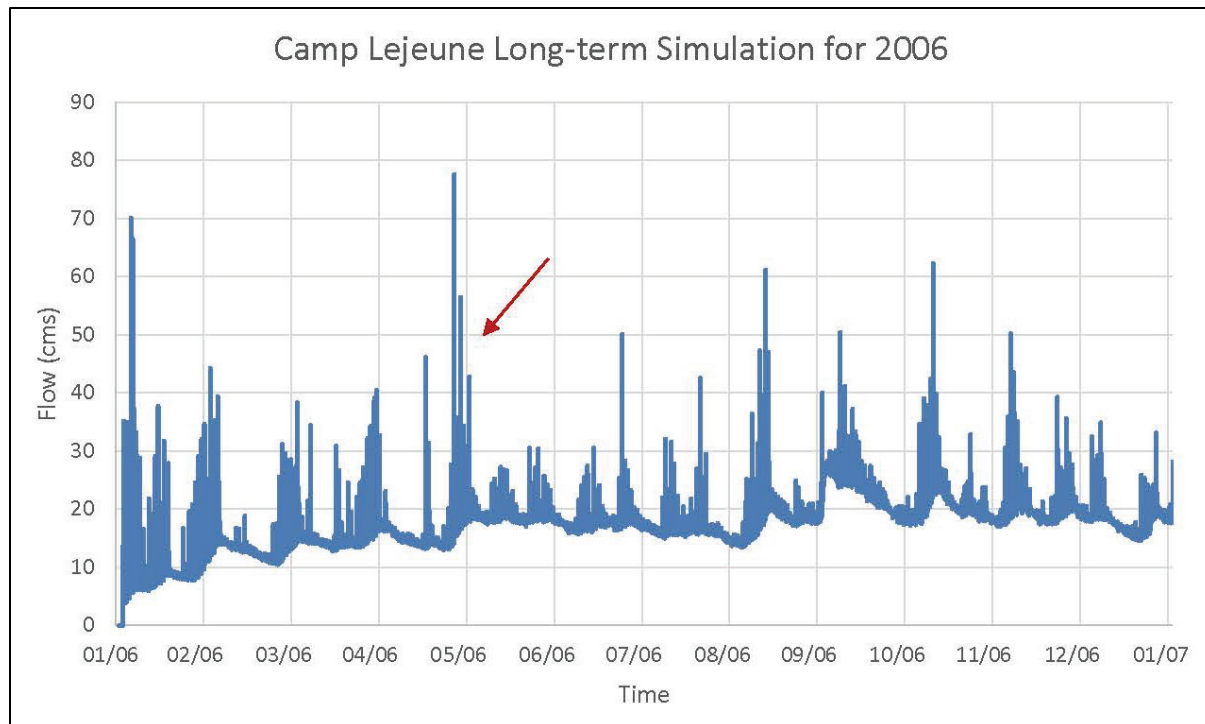


Figure 11. Output hydrograph from 24 precipitation events and corresponding coastal surge in 2006. The red arrow highlights elevated flow conditions likely linked to the “Mother’s Day Storm” that hit North Carolina in early May 2006.

The long-term simulation capability presented in this study aids the Hydrologic Simulator to manage a continuous model run over an extent of temporal scales, in operational mode and also in forecast mode to inform flood hazards.

SUMMARY: A GSSHA model of Camp Lejeune was developed in support of the development of the Military Hydrologic Simulator. The Camp Lejeune model and simulation demonstrate the conditions and hazards of a coastal setting. The GSSHA model was used to simulate overland and stream flooding due to heavy rainfall and coastal storm surge for tropical storm conditions. The model was further developed to simulate a continuous period of 10 years, simulating complex tidal and river flow.

Although Camp Lejeune discharge data were absent for calibration, deployment of GSSHA hydrological model was favorable for the following reasons:

1. Optimized GSSHA performance shows that values of dominating parameters, such as hydraulic conductivity and Manning's roughness, are consistently bounded within a narrow margin and are close to physically defined critical values as defined in the literature (Downer et al. 2016; Ogden et al. 2011; Pradhan et al. 2014; Pradhan et al. 2016).
2. For a distributed hydrological model like GSSHA, satellite remote sensing can provide a broad areal coverage for identifying the physical properties of the catchment.

In this study, a GSSHA model was deployed for an extreme event, a 100-year rainfall return period, and over a long-term observed period of precipitation, both with and without storm surge. All of the coastal inland complex hydrological scenarios deployed in this study are a series of pre-defined scenarios that provide a computational test bed for the Military Hydrologic Simulator.

ADDITIONAL INFORMATION: For additional information, contact Nawa R. Pradhan, Coastal and Hydraulics Laboratory, U.S. Army Research and Development Center, 3909 Halls Ferry Road, Vicksburg, MS 39180, at 601-619-5165, or e-mail: Nawa.Pradhan@erdc.dren.mil. This CHETN should be cited as follows:

Pradhan, N. R., C. W. Downer, S. N. Sinclair, and C. Lahatte. 2019. *Simulation of Coastal Storm Surge and Rainfall Flooding Scenarios at Camp Lejeune with GSSHA*. ERDC/CHL CHETN-I-96. Vicksburg, MS: U.S. Army Engineer Research and Development Center. <http://dx.doi.org/10.21079/11681/33185>.

REFERENCES

- Downer, C. W., F. L. Ogden, J. Neidzialek, and S. Liu. 2005. "GSSHA: A Model for Simulating Diverse Streamflow Producing Processes." *Watershed Models*, Chapter 6, edited by V. P. Singh and D. Frevert. Boca Raton: CRC Press.
- Downer C. W., J. A. Graulau-Santiago, B. E. Skahill, D. M. Weston, N. R. Pradhan, and A. R. Byrd. 2016. *Gridded Surface Subsurface Hydrologic Analysis Modeling for Analysis of Flood Design Features at the Picayune Strand Restoration Project*. ERDC/CHL TR-16-14. Vicksburg, MS: U.S. Army Research and Development Center.
- Hershfield, D. M. 1961. *Rainfall Frequency Atlas of the United States for Durations from 30 Minutes to 24 Hours and Return Periods from 1 to 100 Years*. Technical Paper No. 40. Washington, D C: U.S. Department of Agriculture.
- Knabb, R. D., J. R. Rhome, and D. P. Daniel. 2005. *Tropical Cyclone Report: Hurricane Katrina: 23-30 August 2005*. National Hurricane Center.
- Lee, H. S., and K. O. Kim. 2015. "Storm Surge and Storm Waves Modelling Due to Typhoon Haiyan in November 2013 with Improved Dynamic Meteorological Conditions." *Procedia Eng.* 116: 699–706.
- Ogden, F. L., and B. Saghafian. 1997. "Green and Ampt Infiltration with Redistribution." *J. Irr. and Drain. Engr.* 123(5): 386–393.
- Ogden, F. L., N. R. Pradhan, C. W. Downer, and J. A. Zahner. 2011. "Relative Importance of Impervious Area, Drainage Density, Width Function and Subsurface Storm Drainage on Flood Runoff from an Urbanized Catchment." *Water Resources Research* 47(12). doi:10.1029/2011WR010550.

- Pradhan, N. R., A. R. Byrd, F. L. Ogden, and J. M. H. Hendricks. 2012. "SEBAL Evapotranspiration Estimates for the Improvement of Distributed Hydrologic Model Runoff and Soil Moisture Predictions." *Remote Sensing and Hydrology*, edited by C. M. U. Neale and M. H. Cosh, 435–439. Jackson Hole, WY: IAHS.
- Pradhan, N. R. J., Downer, C. W., and Johnson, B. E. 2014. "A Physics Based Hydrologic Modeling Approach to Simulate Non-Point Source Pollution for the Purposes of Calculating TMDLs and Designing Abatement Measures." *Practical Aspects of Computational Chemistry III*, Chapter 9. Edited by J. Leszczynski and M. K. Shukla, 249–282. New York: Springer, 249–282.
- Pradhan, N. R., A. R. Byrd, M. Jourdan, and J. Ellis. 2016. *Development of Predictive Relationships for Flood Hazard Assessments in Ungaged Basins*. ERDC/CHL CHETN-VIII-8. Vicksburg, MS: U.S. Army Engineer Research and Development Center.
- Rawls, W. J., and D. L. Brakensiek. "A Procedure to Predict Green Ampt Infiltration Parameters. *Adv. Infiltration, Am. Soc. Agric. Eng.* 83: 102–112.
- Rocha, J., A. Perdigao, R. Melo, and C. Henriques. 2010. "Managing Water in Agriculture through Remote Sensing Applications." Edited by R. Reuter. *Remote Sensing for Science, Education, and Natural and Cultural Heritage, Proceedings of EARSeL Symposium 2010*, UNESCO, Paris, 31 May 3 June 2010, 223–230.

APPENDIX

#Python Script: "Processing_HMET_for_GSSHA.py"
"""

This module contains tools for converting hydrometeorological (HMET) data into the format required by GSSHA.

Overview:

=====

The two main functions in this module are "GetMissingString" and "PrintData." Together, these functions will read a CSV file, fill in missing data, and print the GSSHA formatted data to the screen.

Usage:

=====

Typical usage of this module will be something similar to:

```
filename = 'ENTER_YOUR_RAW_HMET_DATA_HERE.csv'
str999, formats = GetMissingString(filename)
PrintData(filename, str999, formats)
```

A script, called "MyScript.py" with the above code can be used to create a GSSHA input file with the ">" operator on the command line. For example:

```
python MyScript.py > GSSHA_HMET_DATA.txt
```

Input file:

=====

This module processes a CSV file that is:

1. Comma delimited
2. Contains the Date and Time in the first two columns

For example, the first two lines of an input file may be:

Date, Time, Barometric Pressure (in), Relative Humidity (%), Sky Cover (8ths), Wind Speed (KTs), Temp (C), Direct Radiation (W/M*M), Global Radiation (W/M*M)

20160101, 2200z, 30.09, 93.8, 8, 4, 18, 0, 0

Output file:

=====

The first two lines of the output for the above input file would look like:

Year Month Day Hour Barometric Pressure (in) Relative Humidity (%) Sky Cover (%) Wind Speed (KTs) Temp (F) Direct Radiation (W/M*M) Global Radiation (W/M*M)

2016 01 01 22 30.09 93.8 100 4 64 0 0

Notice that the output file is space-delimited, the date has been broken into Year, Month, and Day, the time has been changed into Hour, the temperature has been converted from degrees Celsius to Fahrenheit, and the sky cover has been converted from 8ths to a percent.

"""

```
import datetime as dt
import re
```

```
def GetMissingString(csvfile, numColSkip=2):
    """ Returns the format of columns in a CSV file.
    This function reads through a CSV file and determines the format (integer or float) of each
    column.
    Zeros in each column are ignored, only nonzero entries are used to determine the format.
    Args:
    csvfile: A string holding the name of the CSV file.
    numColSkip [default=4]: The number of columns in the file that are skipped before determining
    the format. If the first four columns are year, month, day, and hour,
    then numColSkip should be set to 4, which is the default.
    Returns:
    A list of characters, 'F' for a floating point, 'I' for an integer, or 'U' for unknown. The length of
    the list will be equal to the number of columns in the CSV file minus numColSkip.
    """
    formats = []
    with open(csvfile, 'rU') as f:
        # Read the header line and split the data up by commas
        header = f.readline().strip().strip(',').split(',')
        # Figure out how many total columns are in the file
        numCols = len(header)
        # Create an array to hold the discovered formats
        formats = ['U']*(numCols-numColSkip)
        # Read one line at a time and split the data up by commas
        for line in f:
            parts = line.split(',')
            # Look only at the columns after the first four (after the date and time)
            for i in range(numColSkip, numCols):
                # If the format is unknown
                if(formats[i-numColSkip]=='U'):
                    # If the column is not equal to zero and we are not sure what the format is
                    if( (float(parts[i])!=0) ):
                        # If there is a period, then it is a float
                        if('.') in parts[i]:
                            # Barometric Pressure no data value = 99.999
                            if ('pressure' in header[i].lower()):
                                formats[i-numColSkip] = 'F2'
                            # Radiation no data value = 9999.99
                        else:
                            formats[i-numColSkip] = 'F4'
                        # Humidity no data value = 999 (humidity data can be expressed as a float, but the no data value
                        # is an integer in GSSHA)
                        if ('humidity' in header[i].lower()):
                            formats[i-numColSkip] = 'I'
                        # If there is not a period, then it is an integer = 999
                    else:
                        formats[i-numColSkip] = 'I'
```

```
# If we have figured out all of the formats, we are done and can break out of the "for line in f"
loop
if(not 'U' in formats):
break
# Create the 999 string based on the column formats
str999 = ""
for format in formats:
    if(format=='F2'):
        str999 += ' 99.999'
    elif(format=='F4'):
        str999 += ' 9999.99'
    elif(format=='I'):
        str999 += ' 999'
return str999, formats
def HandleMissingParts(allParts, formats):
    """ Replaces missing data from the CSV file with the GSSHA string (e.g., 99.999, 999.99, 999).
    We assume that missing data in the CSV file is represented either as a single period '.' or as a
    string that starts with '#', such as '#VALUE'.
    Args:
    allParts: Holds the data that contain a period or a #.
    formats: The format of the no data values associated with each column (i.e., the second output of
    "GetMissingString").
    Returns:
    A list containing strings with the data and any missing components replaced with the appropriate
    '999' string.
    """
    output = []
    assert len(allParts) == len(formats)
    for i in range(len(allParts)):
        # If a data point is represented by either a period or a #
        if((allParts[i]=='.' ) | (allParts[i][0]=='#')):
            # If the no data format of that column = F2, replace the cell that has the period or # with 99.9999
            if(formats[i]=='F2'):
                output.append('99.999')
            # If the no data format of that column = F4, replace the cell that has the period or # with 9999.99
            elif(formats[i]=='F4'):
                output.append('9999.99')
            # If the no data format of that column = I, replace the cell that has the period or # with 999
            elif(formats[i]=='I'):
                output.append('999')
            # A check to make sure there are not any other types of missing data fills aside from a period or
            #TEXT
            else:
                assert False, "Uh-oh, something funky happened"
            else:
                output.append(allParts[i])
```

return output

```
def ConvertHeader(oldHeader):
    """ Replaces a header that has the format "Date, Time, ..." with a header that has the format
    "Year, Month, Day, Hour, ..."
    Args:
    oldHeader: A string containing the header line from a CSV file of raw HMET data.
    Returns:
    A string containing the new header.
    """
    headerParts = oldHeader.split(',')
    # New heading titles 'Year, Month, Day, and Hour' are added to the initial headers (excluding the
    # first two; date and time)
    return ', '.join(['Year', 'Month', 'Day', 'Hour'] + headerParts[2:])
def ConvertRowParts(oldParts, formats):
    """ Returns correctly formatted Year, Month, Day, and Hour columns.
    This function reads through the heading created in the 'ConvertHeader' function and designates
    which part of the Date and Time in the raw data
    are associated with the year, month, day, and hour.
    Args:
    oldParts: A list of strings with the data [Date, Time, other data...].
    Returns:
    A list of strings with the proper date and time formats, [Year, Month, Day, Hour, other data...].
    """
    # Specifies year is the first three (0,1,2,3) digits of the date (Note: Date in the raw file is
    # formatted YYYYMMDD)
    year = oldParts[0][0:4]
    # Specifies month is the 4th and 5th digits of the date
    month = oldParts[0][4:6]
    # Specifies day is the 6th and 7th digits of the date
    day = oldParts[0][6:8]
    # Specifies hour is the first two digits of the time (Note: Time in the raw file is formatted
    # HHMM)
    hour = oldParts[1][0:2]
    # Returns the new year, month, day, and hour columns along with the other original headings in
    # the dataset (pressure, humidity, etc.)
    return [year, month, day, hour] + HandleMissingParts(oldParts[2:], formats)
def ConvertTemperature(oldTemp):
    """ Returns formatted temperature with the correct units.
    This function converts the temperature units from degrees Celsius to degrees Fahrenheit.
    Args:
    oldTemp: A string containing the temperature (in degrees Celsius).
    Returns:
    A string containing the temperature (in degrees Fahrenheit).
    """
    newTemp = float(oldTemp)*1.8 + 32
```

```

return '%d'%int(newTemp)
def ConvertSkyCover(oldCover):
    """ Returns formatted sky cover with the correct units.
    This function converts the sky cover units from 8ths to a percent.
    Args:
    oldCover: A string containing the sky cover (in 8ths).
    Returns:
    A string containing the sky Cover (as a percent).
    """
    newPercent = float (oldCover)/8*100
    return '%d'%int(newPercent)
def PrintData(csvfile, missingString, formats):
    """ Extracts and reformats data from a CSV file.
    This function reads the contents in a file named 'csvfile', fills in missing data, and converts the
    data format into the form required by GSSHA. The converted header and data is printed to the
    screen.
    Typically this function will be called after a call to the GetMissingString function.
    ARGS:
    csvfile: A string holding the name of the CSV file of interest.
    missingsString: The string to print when a row of data is missing from the CSV file (i.e., the
    string returned by the 'GetMissingString' function).
    formats: A list of strings with the column formats (e.g., 'F', 'I').
    Returns:
    None
    """
    f = open(csvfile,'rU')
    # Read the header and the first line
    header= ConvertHeader(f.readline().strip())
    headerParts = header.split(',')
    currLine = f.readline().strip().strip(',')
    parts= ConvertRowParts(currLine.split(','), formats)
    # Get the columns containing the sky cover and temperature
    skyInd = [i for i, item in enumerate(headerParts) if re.search('Sky Cover', item)][0]
    headerParts[skyInd] = 'Sky Cover (%)'
    tempInd = [i for i, item in enumerate(headerParts) if re.search('Temp', item)][0]
    headerParts[tempInd] = 'Temp (F)'
    header = ''.join(headerParts)
    # Conver the units of sky cover and temperature
    parts[skyInd] = ConvertSkyCover(parts[skyInd])
    parts[tempInd] = ConvertTemperature(parts[tempInd])
    # Change the break between lines from a comma to a space
    print(header)
    print(' '.join(parts))
    # Where each part of the date is in the dataset
    prevYear= int(parts[0])
    prevMonth = int(parts[1])

```

```
prevDay = int(parts[2])
prevHour= int(parts[3])
# Combine info into datetime object for processing
prevDate = dt.datetime(year=prevYear, month=prevMonth, day=prevDay, hour=prevHour)
    for tempLine in f:
currLine = tempLine.strip().strip(',')
parts= ConvertRowParts(currLine.split(','), formats)
parts[skyInd] = ConvertSkyCover(parts[skyInd])
parts[tempInd] = ConvertTemperature(parts[tempInd])
newYear= int(parts[0])
newMonth = int(parts[1])
newDay = int(parts[2])
newHour= int(parts[3])
newDate = dt.datetime(year=newYear, month=newMonth, day=newDay, hour=newHour)
# If the date gap exceeds 3,600 seconds (1 hour), then add 1 hour to the previous date. This
ensures each hour of the dataset is present (which GSSHA requires).
while( ((newDate-prevDate).total_seconds() > 3600) ):
prevDate += dt.timedelta(hours = 1)
print('%04d %02d %02d %02d%s'%(prevDate.year, prevDate.month, prevDate.day,
prevDate.hour, str999))
print(' '.join(parts))
# Update for the next loop
prevDate = newDate
```

#####

MAIN CODE BELOW

```
#####
# The name of the CSV file containing the raw HMET data. It should be comma-delimited and
have the Date and Time as the first two columns.
filename = 'ENTER_YOUR_RAW_HMET_DATA_HERE.csv'
# Get all of the column formats
str999, formats = GetMissingString(filename)
# Print the data, filling in missing rows with the str999 string
PrintData(filename, str999, formats)
```

***NOTE:** The contents of this technical note are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official endorsement or approval of the use of such products.*