Session 1: Introduction

Download the tutorial.tar.gz file from the dhsvm ftp site

ftp://ftp.hydro.washington.edu/pub/dhsvm/

Select tutorial.3.1.tar.gz and save it to your working directory.

Unzip and untar the file

```
unix% gunzip tutorial.3.1.tar.gz
unix% tar -xvf tutorial.3.1.tar
```

This should expand into the following directory structure:

tutorial3.1/
- *amlscripts/*
- *input/*
- *configfiles/*
- *metfiles/*
- *output/*
  - *GMTscripts/*
- *programs/*
- *source_codes/*
- *modelstate/*
- *DHSVM3.1_Test_Site_Intro.pdf/*

Run arc/info and view database files

Arc/Info only runs on the PC computer (let's say the machine name is 'plane'), so here are the system-specific commands for running arc/info.

```
plane% cd tutorial3.1/input
plane% arc
```

View database files: soil, veg, dem & mask

Convert NODATA from -9999 to 0 in the mask file:

```
Arc: grid
Grid: mask = con(isnull(mask), 0, mask)
Grid: q
```

Export SOIL, VEGETATION, MASK and DEM files as ascii grids

```
Arc: gridascii soil soil.txt
Arc: q
```

Take the same steps to export the VEG, MASK and DEM files

Convert grids to binary

1) Remove the headers from the ascii files:

Use the **tail** command to output all lines from the 7th line in the input file. This command generates a new output file.

```
unix% cd tutorial3.1/input
unix% tail -n +7 dem.txt > newdem.txt
```

Use the **head** command to keep a copy of the header file.

```
unix% head -6 mask.txt > headerfile.txt
unix% head -6 mask.txt
```

2) Ascii → Binary conversion: compile the program myconvert.c :

```
unix% cd tutorial3.1/programs/
unix% cd ..
```

Use **myconvert** to create dhsvm binary input grids:

usage:

myconvert *source_format target_format source_file target_file number_of_rows number_of_column*)

```
unix% cd ..
unix% programs/myconvert asc float input/newdem.txt input/
      dem.bin 213 164
```

Repeat for each of the map files, changing the target format as follows:
Mask, soil, veg = char dem = float

Session 2: Stream, Soil and Road Inputs

Compile required programs (fixroads.c and AddAat2.java). The JAVA program (*jdk1.7.0_05)* is required to run the following commands.

```
plane% cd /tutorial3.1/programs/
plane% gcc fixroads.c -lm -o fixroads
plane% javac AddAat2.java Aat.java AatError.java
```

*(Upon successful run of this script, 3 class files, including **AddAat2.class**, **Aat.class** and **AatError.class**, will appear in the "program" directory.)*

```
plane% cd ../input/
plane% arc
```

Assume that you copied the tutorial directory in C:/workspace, set the workspace:

Arc: `&workspace C:/workspace/tutorial3.1/input`

Set the path for aml scripts:

Arc: `&amlpath ../amlscripts/`

Note: Open the **createstreamnetwork.aml** in the amscripts folder (with Notepad). Look for the call to the AddAat2 java program. Make sure that the path to the class file is correct. You should have AddAat2.class file in the "program" directory of the workspace, therefore the line specifying the class path in the createstreamnetwork.aml must look like:

&sys `java -classpath program/ AddAat2 %streamnet%`

Create stream and soil inputs

usage:

STREAMNETWORK *<dem> <wshed> <soildepth> <stream network> <MOUTH|MASK> {source area} {min depth} {max depth}*

Arc: `&run createstreamnetwork dem mask soild streams MASK 100000 .76 1.5`

Note: all grids should have same domain and resolution.

dem: a grid of basin elevations. Make sure that dem is in the floating point and the regions outside the watershed are defined as NoData, rather than 0. Otherwise you will get a stream network for the whole raster. [In raster calculator, use 'Set Null' function. The format should look like:

`setnull("mask!=1", "dem")`

or

`SetNull("mask", "dem", "VALUE !=1")`

wshed: EITHER a basin mask file, or a grid designating the location of the basin mouth, as indicated by the keyword MOUTH | MASK, if MOUTH is specified, the mask file will be created. Likewise, the mask must be defined as inbasin=1, outside basin=NoData so the outputs share the same boundary as the basin or watershed.

soild: grid of soil depth, if it doesn't exist it will be created as a function of cumulative drainage area and slope, min and max depth must then be specified.

streams: an arc coverage of stream locations, if it doesn't exist the coverage will be created using the specified constant threshold area (source area)

The last three values represent the minimum contributing area before a channel begins, the minimum soil depth, and the maximum soil depth. The thicker the soil depth is the slower the peak response to rain events tends to be.

You will be prompted to see if you want to continue. Type **y**.

This step creates the files stream.network.dat & stream.map.dat that are needed by DHSVM, and the stream coverage file.

It is **VERY IMPORTANT** to note that, if any error occurs before the aml is done, you **MUST** re-run the aml script with original input files. Always keep a copy of the original arcinfo directory. Within "arcinfo", you should have original input GRID files, "program" folder including 3 class files, and "amlscript" folder. Once any error occurs, read the error statement to find out the potential cause. Then delete the "arcinfo" folder you worked on, copy the original arcinfo folder in the specified workspace, and rerun the script.

Create road inputs

If there are road layers in the study site, take this step to create the files road.network.dat and road.map.dat that are needed by DHSVM.

usage:

ROADNETWORK *<dem> <soildepth> <road network>*

Arc: *&run createroadnetwork dem soild roads*

You will be prompted to see if you want to continue. Type **y**.

Designate stream save indicator

We want to save the output at the final stream segment within the stream network. This is designated within the file /arcinfo/stream.network.dat with a value of -1 in the sixth column. Routing results for this stream segment will be placed in the stream output file if the keyword SAVE appears in the last column.

Within xemacs, open stream.network.dat and make the required changes.

unix% `xemacs &`

Add **SAVE "SPRINGBROOKCREEK"** following the last column of stream segment ID. To properly run the model, the value of -1 in the stream.network.dat file should be replaced with 0. Save the file and exit xemacs.

Export soil depth grid and convert to binary

Arc: `gridascii soild soildepth.txt`

Remove the header

unix% `cd ../arcinfo`
unix% `tail -n +7 soildepth.txt > temp1`
unix% `rm soildepth.txt`
unix% `mv temp1 soildepth.txt`

Convert to binary

unix% `../programs/myconvert asc float soildepth.txt soildepth.bin 52 67`

Create road and stream class files

These are ascii files that contain a look-up table of channel hydraulic properties for each road or stream class. The following columns are necessary for each channel class: <Channel Class> <Hydraulic Width (m)> <Hydraulic Depth (m)> <Manning's friction coefficient> <Max Infiltration (m/s)>

The channel hydraulic width and depth are currently hard-wired in the createroadnetwork and createstreamnetwork scripts, as follows:

Roads:

#Class	Width	Depth
1	0.5	0.5
2	0.5	0.5
3	0.25	0.25
4	0.25	0.25
5	0.25	0.25
6	0.25	0.25
7	0.25	0.25
8	0.25	0.25
9	0.25	0.25

Streams:

#Class	Width	Depth
1	1.5	1.0
2	2.0	1.5
3	3.0	5.0
4	5.0	7.5
5	2.0	1.5
6	5.0	4.0
7	7.5	7.0
8	10.0	10.0
9	15.0	13.0

Edit these tables (road.class.dat and stream.class.dat) in emacs to add the friction and infiltration columns.

The maximum infiltration rate is **NOT** used for the stream class file, but it MUST be specified as a place holder. Manning's roughness can vary from 0.025 to 0.15 for natural channels. Infiltration into the roads may be assumed to be 0 m/s for a starting point (i.e. 100% impervious).

Create surface routing file for the urban module

1) run the program ‘find_nearest_channel.c’

usage:

find_nearest_channel *nrows ncols binary_flowd_file binary_mask_file stream_map_file n_header_map_file*

unix% `cd programs`
unix% `gcc find_nearest_channel.c -o find_nearest_channel`
unix% `cd ..`
unix% `programs/find_nearest_channel 214 164 input/flowdir.bin input/mask.bin stream.map 9`

where:
flowd_file is a char binary flow direction file in the same format as the DHSVM mask file. Make sure that the flowd_file is free of sinks, etc, and flowdirection is assumed to be from ARC-INFO, i.e. 1 to 128.
binary mask_file and ***stream_map file*** are the DHSVM specific input files for mask and stream_map file, respectively.
n_header_map_file are the number of header lines in the stream map file, i.e. lines starting with # . Enter 0 if there are no header lines. Caution: make sure you are referring to the map file not the network file

2) Fill in the path to the flow_routing.txt file under the section of VEGETATION in the configuration file.

Create model states

Compile the MakeModelStateBin.c:

unix% `make -f Makefile.ModelState`

(Makefile.ModelState is under tutorial3.1/programs/)

Create the Interception, Snow and Soil state files for the date specified in InitialState.txt

unix% `./MakeModelState modelstate/ initialstate.txt`

InititalState.txt provides a sample input file set up for Springbrook Creek, assuming no interception storage, no snow for 210 days and 35% volumetric soil moisture in all layers.

The 'initialstate' file MUST contain the following information:
- path for output file (change it if not ../modelstate)
- date for the model state, in mm/dd/yyyy-hh
- number of rows (ny) and number of columns (nx)
- maximum number of vegetation layers
- rain interception in m for each vegetation layer
- snow interception in m for top vegetation layer
- snow cover mask
- number of days since last snow fall
- snow water equivalent in m

- liquid water content in m of bottom layer of snowpack
- temperature in C of bottom layer of snow pack
- liquid water content in m of top layer of snowpack
- temperature in C of top layer of snow pack
- cold content of snow pack
- maximum number of root zone layers
- volumetric soil moisture content for each layer
 (including the layer below the lowest root zone layer)
- temperature in C at soil surface
- soil temperature in C for each root zone layer
- ground heat storage
- runoff

Create initial channel state files

*Also have to change the path to which the output file is stored.

If necessary, make the script MakeChannelState.scr executable:

```
unix% chmod 755 MakeChannelState.scr
```

usage: MakeChannelState.scr <StreamNetworkFile> <InitialDepth> <Output Date String: MM.DD.YYYY.hh.mm.ss>

To run the script (calculates volume of water in each channel segment, given an assumed uniform initial depth in meters):

```
unix% ./MakeChannelState.scr ../input/stream.network 0.25
10.01.1995.00.00.00
```

Create shadow and skyview files

If necessary, make the script MakeChannelState.scr executable:

```
unix% chmod 755 run_solar_programs_daily.scr
```

Edit run_solar_programs_monthly.scr**,** change the parameter settings and the location of DEM map (should be in ../input/dem) and output location (../input).

```
unix% ./run_solar_programs_daily.scr
```

Session 3: Running the model

```
unix% cd ../source_codes/
unix% make
unix% cd ..
unix% source_codes/DHSVM configfiles/INPUT.3.1.SpringBrook
```

Note that the sediment option is turned off in this sample configuration file for the Springbrook creek. If the sediment module or mass wasting option is going to be implemented, please follow the *Tutorial for DHSVM 3.0* (*ftp://ftp.hydro.washington.edu/pub/dhsvm/*)

Use GMT to view output hydrographs/basin average time series

```
unix% output/GMTscripts
```

 # Produces a graph of flow (plot_flow.ps)

```
unix% plot_flow.scr
```

 # Close ghostview window

```
unix% ctrl-c
```

Produces a graph of precipitation over the basin (plot_precip.ps)

```
unix% plot_precip.scr
```

Produces a graph of snow water equivalent (plot_swe.ps)

```
unix% plot_SWE.scr
```

Each of these can also be viewed in ghostscript, e.g.:

```
unix% gs  plot_flow.ps
```

Use GMT or ArcInfo to view output spatial images

The script **plot_modelmap.scr** allows you to plot maps, by changing the script slightly. It works if you designated only one date per map number. For each map you need to designate the map name, the name of the output file, the name of the input file and the name of the color plot table (cpt). Create the Soil Moisture Map first.

```
unix% xemacs &
```

Edit **plot_modelmap.scr** so that the Soil Moisture map name, output file, input file and cpt file do not have a "#" in front of it. Add a "#" in front of all the other map file names. Save the file, but leave it open. The paths to each file may require modification.

```
unix% plot_modelmap.scr
unix% ctrl-c          <to close ghostview window>
```

Repeat these steps for any other maps for which you created files.
Each of these can also be viewed in ghostscript, e.g.:

```
unix% gs Map.1.Soil.Moist.ps
```