Session 1: Creation of Input Files

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

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

To access through netscape, type:
unix% `netscape &`

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

Unzip and untar the file

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

This should expand into the following directory structure:

tutorial/
- arcinfo.tar.gz
- configfiles/
- metfiles/
- output/
  - GMTscripts/
- programs/
- source/
  - mwm/
  - dhsvm_2.0.1.tar.gz

Unzip/untar the files in the arcinfo database

unix% `cd tutorial`
unix% `gunzip arcinfo.tar.gz`
unix% `tar -xvf arcinfo.tar`

This should expand into the following directory structure:
- arcinfo/
  - rainysoil/
  - rainyveg/
  - rainydem/
  - rainymask/
  - roads/
  - rainy10m/
  - amlscripts/
  - info/

Run arc/info and view database files

Arc/Info only runs on one of our computers (plane2), so here are our system-specific commands for running Arc/Info.

unix% `xterm &` <switch to new window>
unix% `ssh plane`
plane% `cd tutorial/arcinfo/`
plane% /usr/local/bin/setup owclient arcinfo81
plane% `arc`

View database files: rainysoil, rainyveg, rainydem & rainymask

The mask file requires data in character format. Because -9999, the standard no data designation, does not fit in the storage space allotted for a character, it must be replaced with zero for no data.

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

Arc: `grid`
Grid: `rainymasks = con ( isnull ( rainymask ) , 0 , rainymask )`
Grid: `quit`

Export soil, vegetation, mask and dem files as ascii grids

Arc: `gridascii rainysoil rainysoil.txt`
`Repeat this command for each of the four files. Note: the mask grid is now named rainymasks`
Arc: `quit`

Convert grids to binary

Since Arc/Info runs on a SUN and we are running on pc's (different byte order), you must convert the grids to binary on your local machine (i.e. go back to your home window).

Remove the headers from the ascii files:

unix% `cd tutorial/arcinfo`

If necessary, make the following script executable:
unix% `chmod 755 ../programs/rmhdr.scr`
unix% `../programs/rmhdr.scr`
Only use this script once. Multiple uses will cause errors, as it removes the first six lines of each file.

Compile the utility program myconvert.c:

unix% `cd tutorial/programs/`

```
unix% gcc myconvert.c -lm -o myconvert
```

```
Find the number of rows and columns in the input files:
```

unix% more rainyhdr.txt

This command opens the header removed from the ArcInfo files. The first two columns are the number of rows and columns in the file.

Use myconvert to create dhsvm binary input grids:

usage: myconvert *source_format target_format source_file target_file number_of_rows number_of_columns*

```
unix% myconvert ascii char ../arcinfo/rainymask.txt
../arcinfo/rainymask.bin 52 67
```

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

You have now created a .bin file corresponding to each input file within the tutorial/arcinfo/ directory.

Create model states

Compile the MakeModelStateBin.c:

```
unix% make -f Makefile.ModelState
```

Usage: MakeModelState <infofile>
The info file MUST contain the following information:
- path for output file
- 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

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

Run the program

unix% `MakeModelStateBin InitialState.txt`

This will create the Interception, Snow and Soil state files for the date specified in InitialState.txt

unix% `mv *State* ../modelstate`

Session 2: Stream, Soil and Road Inputs

Go back to your plane2 window:

Compile required programs (fixroads.c and AddAat2.java)
plane% `cd /tutorial/programs/`
plane% `gcc fixroads.c -lm -o fixroads`
plane% `javac AddAat2.java Aat.java AatError.java`
plane% `cd ../arcinfo/`
plane% `arc`

Set the path for aml scripts:
Arc: `&amlpath amlscripts/`

`Within the amlscripts/ directory are scripts that create stream and soil input files for DHSVM.`

Create stream and soil inputs

The first script we will run is the createstreamnetwork script which creates a soil depth grid and stream network files.

The usage of the script is as follows: STREAMNETWORK <dem> <wshed> <soildepth> <stream network> <MOUTH|MASK> {source area} {min depth} {max depth}

Arc: `&run createstreamnetwork rainydem rainymask soild streams MASK 100000 .76 1.5`

Note: all grids should have same domain and resolution
dem: a grid of basin elevations
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.
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)
{min depth} and {max depth} are the minimum and maximum soil depths for the soil depth file.

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

This step creates the files stream.network.dat and stream.map.dat needed by DHSVM.

Create road inputs

This step creates the files road.network.dat and road.map.dat that are needed by DHSVM.

usage: ROADNETWORK <dem> <soildepth> <road network>

where <dem> is the dem grid, <soildepth> is the soil depth grid created in the previous step, and <road network> is an existing arc coverage within the arcinfo directory.

Arc: `&run createroadnetwork rainydem 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. Add SAVE "RAINY CREEK" following the last column of Rainy Creek, stream segment 1. To properly run the model, the value of -1 in the stream.network.dat file should be replaced with 0.

unix% `xemacs &`

```
Within xemacs, open stream.network.dat and make the required
changes.  Save the file and exit xemacs.
```

Export soildepth grid and convert to binary

The soil depth file is required to run DHSVM. Therefore, we need to output the ArcInfo grid to a text file and convert it to binary. This is done in the same manner as with the dem, soil, veg, and mask files in Session 1.

export the grid to a text file

Arc: `gridascii soild soildepth.txt`

Remove the header

On freebsd machine:

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

Convert to binary

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

Create road and stream class files

These are ascii files that contain look-up tables 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.

unix% `xemacs &`

Open each file within xemacs and add the needed values.

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 initial channel state files

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):

```
MakeChannelState.scr ../arcinfo/stream.network.dat 0.25
10.01.1990.03.00.00
```

Session 3: Running the model

Untar and compile the source code

```
unix% cd source/
unix% gunzip dhsvm_2.0.1.tar.gz
unix% tar -xvf dhsvm_2.0.1.tar
unix% cd dhsvm_2.0.1/
unix% make
```

Complete the configuration file

```
unix% xemacs &
```

```
The input file resides in the /configfiles/ directory and is
named INPUT.rainycr.  Open this with xemacs and edit the
required parameters.
```

Look through the file and fill out:
path names
DHSVM requires path names for all of the input files that were just created. Therefore, edit all of the path names to the binary files created such as rainydem.bin.

Model Start = 10/01/1990-03
Model End = 10/01/1994-03
Number of Model States: 1

Model State Date 1 = 10/01/1994-03

Run the model

From the /source/dhsvm_2.0.1/ directory, type the following command to run DHSVM:
unix% `DHSVM ../../configfiles/INPUT.rainycr`

At this point, if you missed one or two of the proper path names within the INPUT file, DHSVM will inform you that a file was not found and exit. If this happens, simply change the path name in INPUT.rainycr of the file DHSVM is missing and re-run the program.

Session 4: Model Output and Calibration
We will now make another model run. This would normally overwrite the output from the previous run; therefore we will copy the files from the previous run to a new directory, output1.

Copy the model state files and backup previous output

```
unix% cd tutorial/
unix% cp output/*1994* modelstate/
unix% mv output output1
unix% mkdir output
unix% mkdir output/GMTscripts
unix% mv output1/GMTscripts/* output/GMTscripts
```

For the next run, we will output maps of variables of your choosing. We also want to output the model state for 10/01/1995-03.

Edit the configuration file

```
unix% xemacs &
```

Change:
Model Start = 10/01/1994-03
Model End = 10/01/1996-03
Model State Date 1 = 10/01/1995-03

This will output the model state on November 1, 1995 at 3 a.m.

Next, fill out the Model Maps Section. Refer to the list of model variable reference numbers at http://www.hydro.washington.edu/Lettenmaier/Models/DHSVM/output_varid.htm for the Map Variables. (Map 512 is not working.) Choose the number of variables to output, the date that you wish to output, and the variable number. At least one of them should be soil moisture for the following exercise.

Run the model

```
unix% cd source/dhsvm_2.0.1
unix% DHSVM ../../configfiles/INPUT.rainycr
```

Use GMT to view output hydrographs/basin average time series

```
unix% cd ../../output/GMTscripts

unix% plot_flow.scr              # Produces a graph of the Rainy Creek flow
                                 (plot_flow.ps)
unix% ctrl-c <to close ghostview window>
```

```
unix% plot_precip.scr        # Produces a graph of precipitation over the basin (plot_precip.ps)
unix% plot_SWE.scr           # Produces a graph of snow water equivalent (plot_swe.ps)
```

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 slighty. 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
```

Experiment with adjusting calibration parameters and re-run

Open the configuration file in emacs and edit the parameters

```
unix% xemacs &
```

Run alternate road or vegetation scenario

Scenario 1: No Roads

Open configuration file in emacs and edit Road Network Section by commenting out the three path and file names (place a # before each `Road`). Note: you must comment out the entire line containing the path name. Otherwise, DHSVM will not run.

```
unix% xemacs &
```

Scenario 2: New Vegetation Scenario

Open the rainyveg.txt in emacs and change all the vegetation to type 7 (Woodland Grass) or type 8 (Closed Shrub).

```
unix% xemacs &
```

Remove the header:

```
unix% cd /arcinfo
unix% tail +7 rainyveg.txt >! newrainyveg.txt
```

Convert to binary:

```
unix% /programs/myconvert ascii char newrainyveg.txt
rainyveg.bin 52 67
```

Open the configuration file in emacs and edit the file names

```
unix% xemacs &
```

Save the previous output and rerun DHSVM

```
unix% mv output output2
unix% mkdir output
unix% cd ../../source/dhsvm_2.0.1
unix% DHSVM ../../configfiles/INPUT.rainycr
```

Save the output and rerun DHSVM with a different scenario.