Assignment 9: Geospatial simulations using path sampling

October 29, 2010

1 Task

Modify the python script to enhance flow simulation and generate dynamic surface visualization

2 Approach

2.1 Generate slope input

- Download the NC sample GRASS Location, nc_spm_08, if you do not have it:
http://grass.osgeo.org/download/data.php

- From nc_spm_08, use r.slope.aspect to compute dx and dy maps from the elevation map, elev_lid792_1m.

- Output the dx and dy maps as ascii grids using r.out.ascii.

- In the Python program, change the paths in lines 68 and 69 to import your dx and dy files.

2.2 Animate particle sampling

- Change path on line 96 to a directory where you want to save xyz walker locations and pictures for your animation. I would recommend making an extra directory that is only meant to hold the pictures since about 40 will be generated.

- Change parameters
c = 0.8
N = 0.01*A
T = 2000*tau
N is the number of particles, decreasing N will decrease the density of walkers and also make simulation run faster.

The parameter T is the total time where tau is the time step.

- Run the simulation.

- Write a script similar to Assignment 8 that will import walker locations as vector maps, display walkers overlayed on elev_lid792_1m, save the images, and finally animate. Output walker locations are comma separated. Your script may look something like:

```
python
import os,commands
import grass.script as grass
grass.run_command('g.region',rast='elev_lid792_1m')
# list files in directory
files = commands.getoutput('ls').split()
os.system('d.mon x0')
for f in files:
    grass.run_command('v.in.ascii', input=f, output=f, format='point', fs=',', overwrite
    grass.run_command('d.rast', map='secref_elev_1m')
    grass.run_command('d.vect', map=f, icon='basic/point')
    grass.run_command('d.out.file', output=f, format='png')
    grass.run_command('d.erase')

os.system('convert -delay 6 -loop 0 *.png overlandFlow.gif')
#for f in files:
#    os.system('rm '+f)
#    os.system('rm '+f+'.png')

exit()
```

- Because an animation cannot be submitted, submit the final frame as a result.

2.3 Run a fuller simulation

Note: This simulation may take a while. It took about 40 minutes for me. There is a little progress bar, but it is *not* linear since constant rain increases the number of walkers with each iteration.

- Comment out line 106 so that walkers are not output throughout the simulation.

- Uncomment line 112 and change the path in line 110 in order to output the final walker locations to a text file.

- Change parameters
c = 0.8
N = 10*A

T = 500*tau

- Rerun the simulation.

- Generate a depth map by importing the walker locations into GRASS using r.in.xyz method=n. This is effectively a histogram which converts the walker locations (particle representation) into a depth map (field representation). Note that the map generated by r.in.xyz method=n does not really show depth; it is only a count of walkers, therefore, it will be off by a factor which is equal to the volume of water represented by each walker.
A color table that can help you view your depth map is:

```
0 255:255:0
100 0:191:255
1000 0:0:255
16000 0:0:128
```

- Generate another depth map by running the GRASS module r.sim.water elevin=elev_lid792_1m dxin=dx dyin=dy, where dx and dy are the partial derivative maps generated from r.slope.aspect. Leave the other parameters to default values.

- Include pictures of the depth maps from the Python program and r.sim.water. Qualitatively compare the depth maps.