Contract No:

This document was prepared in conjunction with work accomplished under Contract No. DE-AC09-08SR22470 with the U.S. Department of Energy (DOE) Office of Environmental Management (EM).

June 6, 2018

SRNL-L3200-2018-00067
RSM Track #: 10560

TO: B. T. BUTCHER, 773-42A

FROM: J. A. DYER, 773-42A

REVIEWER: T. L. DANIELSON, 773-42A

SUBSIDENCE INFILTRATION MODEL DESIGN CHECK FOR E-AREA LLWF

Ref:

1. SRNL-STI-2017-00729, J. A. Dyer and G. P. Flach to B. T. Butcher, *E-Area LLWF Vadose Zone Model: Probabilistic Model for Estimating Subsided-Area Infiltration Rates*, 12/12/2017

Scope

A design check will be performed on the revised Python-based probabilistic subsidence infiltration model for the proposed E-Area Low-Level Waste Facility (LLWF) final closure cap design.

Background

Dyer and Flach (ref. 1) describe a Python-based probabilistic model employing Monte Carlo sampling that generates statistical distributions of the upslope-intact-area to subsided-area ratio ($Area_{UAi}/Area_{SAi}$) for E-Area LLWF closure-cap subsidence scenarios that vary in percent subsidence and the total number of uniformly sized compartments. The two main input parameters in the first version of the model are the integer number of subsided compartments and the total integer number of compartments (intact plus subsided). As a result, percent subsidence is implicit in the assumed integer-number of subsided compartments, which leads to computational limitations for low-percent-subsidence cases (≤ 1 percent).

The revised version of the model provides more flexibility in case definition by allowing rational numbers to be used for all input parameters except for the total number of compartments. In addition, the revised model calculates a cap-average infiltration rate for each subsidence case using HELP-model-generated infiltration rates for the intact case. Revised input parameters include compartment size (rational, feet), total number of compartments (integer), percent subsidence (rational), average annual rainfall minus average evapotranspiration rate from HELP (inches/year, rational), and intact infiltration rate from HELP (inches/year, rational). The benefit of the revised model is that any percent subsidence case from 0% to 100% can be simulated.

Design Check Objective

Table 1 below gives the average intact infiltration rate as a function of relative time for an E-Area LLWF closure cap design with 2% slope and 585-foot slope length. Also reported in Table 1 are infiltration rates for three subsidence cases (2%, 0.6%, and 0.04%), which are slope-length-weighted cap-averages for slope lengths of 545 feet and 110 feet.

Table 1. Average Infiltration Rates for E-Area LLWF Intact and Subsidence Cases (Intact Case: 2% slope, 585-foot slope length)

Relative Year	Intact Infiltration Rate (in/yr)	Slope-Length-Weighted, Cap-Average Infiltration Rate (in/yr)		
		2.0% Subsidence (ST6, ST15-21)	0.6% Subsidence (ST5, ST7, ST14)	0.04% Subsidence (ET2)
100	0.00088	5.858	2.166	0.157
180	0.00791	5.824	2.188	0.165
290	0.18881	5.938	2.336	0.348
300	0.20409	5.972	2.353	0.361
340	0.32167	6.020	2.462	0.477
380	0.40513	6.092	2.514	0.568
480	1.45728	6.771	3.465	1.600
660	3.23003	7.928	4.982	3.358
1100	7.01494	10.380	8.274	7.102
1900	10.64990	12.719	11.416	10.708
2723	11.47210	13.255	12.129	11.520
3300	11.53200	13.302	12.195	11.582
5700	11.63140	13.346	12.271	11.678
10100	11.67340	13.373	12.304	11.720

Year 0 is the beginning of institutional control
Year 100 is the end of institutional control and the installation date of the closure cap

The objective of the design check is to confirm the accuracy of the reported infiltration rates in inches per year at each time step for the three subsidence cases.

Design Check Steps and Associated Files

All files identified below can be found at \\godzilla-01\hpc_project\projwork50\E-Area\PA_2019\CoverSystem\Subsidence_Infiltration_Design_Check_2018

The probabilistic infiltration model consists of two key files:

- Python program file SubsideAverage_rev5a.py

- Windows batch file runPython_rev5a.bat, which contains the required input parameters and output filenames for each case

The Windows batch file runPython_rev5a.bat is set up to generate six infiltration-rate time profiles, two for each percent-subsidence case of interest (545-foot and 110-foot slope lengths each at 2%, 0.6%, and 0.04% subsidence). Fourteen time steps are included for each infiltration-rate time profile. Figure 1 shows the two sets of batch file inputs for the 2% subsidence case. Note that the model cases designated "545-foot slope length" are based on 550 feet in the Python model simulations, because the total number of compartments must be an integer number (55 compartments times 10.0-foot compartment size). This model constraint introduces only a small error in the slope-length-weighted infiltration rates in Table 1. Figure 2 displays the section of Python script from SubsideAverage_rev5a.py that defines and reads the arguments in columns 2 through 10 on each line of the batch file.

The design check comprises the following steps:

- Copy runPython_rev5a.bat and SubsideAverage_rev5a.py to a new folder on a computer where the Python software source code is installed.
- Confirm that the Python programming script in SubsideAverage_rev5a.py (Attachment 1) correctly implements the subsidence infiltration conceptual model assumptions outlined in Attachment 2. Please consult with Greg Flach or Jim Dyer for more explanation, if needed.
    - ✓ Python code was thoroughly checked for bugs and proper technical implementation of the subsidence infiltration conceptual model. The code is essentially error free. However, it was noticed that for small numbers of realizations, the possibility exists for the value "length" to be undefined, which leads to errors that kill the execution of the code. This occurs for small numbers of realizations because there is a non-zero probability that 0 subsided compartments will be selected. This is not an issue, but I wanted to point it out in case there is a desire to make it so that this does not kill execution.
        - ✓ The author experienced this same model execution failure when the number of realizations was set at 100 for the 0.04% subsidence case. The Python code in Attachment 1 has been modified to correct this limitation, and is included in Attachment 3.
- Execute the runPython_rev5a.bat file by double-clicking on the filename.
- The batch file will generate three output files for each percent-subsidence case: detailed output file (.out), summary file (.sum), and tabulated results file for import into Microsoft Excel (.tab). Figures 3, 4, and 5 display examples of the three output files.
    - ✓ Works as expected

- The results from the .tab files were copied onto the "By Slope Length & % Subsid." worksheet in Average Infiltration Case for Low Percent Subsidence_05-29-2018.xlsx. Confirm that the results from the three .tab files for each percent-subsidence case were correctly transcribed. Note that output values for the new Monte Carlo simulations may not exactly match the values in Average Infiltration Case for Low Percent Subsidence_05-29-2018.xlsx; however, agreement should be very good for 100,000 realizations.

  ✓ Comparison of the data set originally obtained by Dyer and Flach and the data set obtained by Danielson for design checking has relatively good agreement. The largest percent difference in the cap averaged infiltration rate is 3.39% and was for the following case "SubsidedAverage_rev5a_DesignCheck.py Case_0.04Percent 10. 55 0.04 16.5 0.00088 100000 False True w" where the infiltration rates predicted from the Monte Carlo code are 0.181 and 0.187 for the original and current test runs, respectively. This difference is not expected to be significant.

    ✓ A less than 5 percent difference is acceptable.

- The infiltration rates reported in Table 1 are the slope-length-weighted cap-averages for slope lengths of 545 feet and 110 feet. The slope-length weighting occurs on Worksheet "Slope-Length-Weighted Data" in Average Infiltration Case for Low Percent Subsidence_05-29-2018.xlsx. Confirm that the results reported in Table 1 for the three subsidence cases are correctly calculated.

  ✓ Calculation of slope-length-weighted cap-averages has been performed correctly.

```
SubsidedAverage_rev5a.py Case_2Percent 10. 55 2.0 16.5 0.00088 100000 False True w
SubsidedAverage_rev5a.py Case_2Percent 10. 55 2.0 16.5 0.00791 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 55 2.0 16.5 0.18881 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 55 2.0 16.5 0.20409 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 55 2.0 16.5 0.32167 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 55 2.0 16.5 0.40513 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 55 2.0 16.5 1.45728 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 55 2.0 16.5 3.23003 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 55 2.0 16.5 7.01494 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 55 2.0 16.5 10.6499 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 55 2.0 16.5 11.4721 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 55 2.0 16.5 11.5320 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 55 2.0 16.5 11.6314 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 55 2.0 16.5 11.6734 100000 False True a
#
SubsidedAverage_rev5a.py Case_2Percent 10. 11 2.0 16.5 0.00088 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 11 2.0 16.5 0.00791 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 11 2.0 16.5 0.18881 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 11 2.0 16.5 0.20409 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 11 2.0 16.5 0.32167 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 11 2.0 16.5 0.40513 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 11 2.0 16.5 1.45728 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 11 2.0 16.5 3.23003 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 11 2.0 16.5 7.01494 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 11 2.0 16.5 10.6499 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 11 2.0 16.5 11.4721 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 11 2.0 16.5 11.5320 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 11 2.0 16.5 11.6314 100000 False True a
SubsidedAverage_rev5a.py Case_2Percent 10. 11 2.0 16.5 11.6734 100000 False True a
```

Figure 1. Example Windows Batch File Input for Python Probabilistic Model

```
### Read filenames
prefixFile             = sys.argv[1]  # prefix for output files
compartmentsSize = float(sys.argv[2]) # size of compartment (feet)
compartmentsTotal  = int(sys.argv[3]) # total number of compartments
percentSubsided  = float(sys.argv[4]) # percent subsidence
F_notET          = float(sys.argv[5]) # annual average rainfall minus evapotranspiration
F_intact         = float(sys.argv[6]) # intact infiltration rate
realizations       = int(sys.argv[7]) # (number of Monte Carlo realizations)
debugArg               = sys.argv[8]  # debug flag (true or false)
graphicArg             = sys.argv[9]  # graphic flag for .out file (>>>>>>O>>>>>>)
appendFlag            = sys.argv[10]  # append flag for summary file (w for overwrite or a for append)
```

Figure 2. Definition of Arguments in Windows Batch File

```
              Compartment size: 10.000000
    Total number of compartments: 55
                    Slope length: 550.000000
   Percent subsided compartments: 2.000000
            Flux, intact cover: 0.000880
           Flux, subsided cover: 16.500000
                    Realizations: 100000
=======================================================
Subsided compartments
>>O>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>O>>>>>>>>>>>>>>O>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>O>>>>>>>>>>>>>>>>>>>>O>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>O>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>O>>>>>>>>>>>>>O>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>O>>>>>>>>>>>>O>>>>>>>>>>>>>>>>>>>>>>
>>>>O>>>>>>>>>>>>>>>>>>>>>>>>>O>>>>>>>>>>>>>>O>>>>>>>>>
>>>>>>>>>>O>>>>>>>>O>>>>>>>>>>>>>>O>>>>>>>>>>>>>>>>>>>>
>>>>>O>>>>>>>>>>>>>>>>O>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>O>>>>>O>>>>>>>>>>O>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>O>>>>>O
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>O>>>>>>
>>O>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>O>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>O>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>O>>>>>>>>>>>>>>>>>>>>>>>
>>>>>O>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>O>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>O>>>>>>>>O>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>O>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>O>>>>>>>>>O>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

Figure 3. Output File – Portion of First Time Step in Two-Percent Subsidence Case (> represents an intact compartment and O represents a subsided compartment)

```
                Compartment size: 10.000000
     Total number of compartments: 55
                    Slope length: 550.000000
    Percent subsided compartments: 2.000000
              Flux, intact cover: 0.000880
            Flux, subsided cover: 16.500000
                    Realizations: 100000
=================================================================
Percent subsided/Avg upslope ratio
    2.00/19.15
Percent subsided/Avg upslope ratio single hole
    2.00/31.97
Sample subsided
    2.00
Min/avg/max compartments
    0/1.10/8
Min/median/mean/max length
    0.00/16.00/19.15/54.00
Min/median/mean/max single hole length
    0.00/34.00/31.97/54.00
Slices intact/subsided (%)
    32.92/67.08
Subsided slice intact/subsided (%)
    98.18/1.82
Fluxes notET/intact/runoff (in/yr)
    16.500000/0.000880/16.499120
Fluxes intactAvg/subsidedAvg (in/yr)
    0.000880/544.046831
Fluxes intactDownAvg/subsidedDownAvg/coverDownAvg (in/yr)
    0.000880/9.892625/6.636460
Fluxes intactAcrossAvg/subsidedAcrossAvg/coverAcrossAvg (in/yr)
    0.000880/364.957785/6.636460
Fluxes coverAvg (in/yr)
    6.636460
```

Figure 4. Portion of Summary File – First Time Step in Two-Percent Subsidence Case

Hole/ Compartment Size (ft)	Number of Compartments	Slope Length (ft)	Percent Subsidence	Infiltration Rate Less Evapotranspiration (in/yr)	Intact Infiltration Rate (in/yr)	Number of Realizations	Upslope-to-Subsided Area Ratio	Fraction Intact (fIntact)	Fraction Subsided (fSubsided)	fSubsided, Intact	fSubsided, Subsided	Cap-Averaged Infiltration Rate (in/yr)
10.000000	55	550.000000	2.000000	16.500000	0.000880	100000	31.974240	0.329180	0.670820	0.981818	0.018182	6.636460
10.000000	55	550.000000	2.000000	16.500000	0.007910	100000	32.023526	0.327990	0.672010	0.981818	0.018182	6.662360
10.000000	55	550.000000	2.000000	16.500000	0.188810	100000	31.949757	0.333040	0.666960	0.981818	0.018182	6.706219
10.000000	55	550.000000	2.000000	16.500000	0.204090	100000	31.934323	0.330210	0.669790	0.981818	0.018182	6.739959
10.000000	55	550.000000	2.000000	16.500000	0.321670	100000	31.826158	0.331290	0.668710	0.981818	0.018182	6.778642
10.000000	55	550.000000	2.000000	16.500000	0.405130	100000	31.936634	0.329920	0.670080	0.981818	0.018182	6.863615
10.000000	55	550.000000	2.000000	16.500000	1.457280	100000	32.017087	0.326390	0.673610	0.981818	0.018182	7.540184
10.000000	55	550.000000	2.000000	16.500000	3.230030	100000	32.049126	0.330700	0.669300	0.981818	0.018182	8.566918
10.000000	55	550.000000	2.000000	16.500000	7.014940	100000	31.938290	0.330740	0.669260	0.981818	0.018182	10.816600
10.000000	55	550.000000	2.000000	16.500000	10.649900	100000	31.974066	0.327530	0.672470	0.981818	0.018182	13.008455
10.000000	55	550.000000	2.000000	16.500000	11.472100	100000	31.927858	0.329240	0.670760	0.981818	0.018182	13.491185
10.000000	55	550.000000	2.000000	16.500000	11.532000	100000	31.802323	0.330220	0.669780	0.981818	0.018182	13.516521
10.000000	55	550.000000	2.000000	16.500000	11.631400	100000	31.906126	0.327820	0.672180	0.981818	0.018182	13.589360
10.000000	55	550.000000	2.000000	16.500000	11.673400	100000	32.042206	0.327340	0.672660	0.981818	0.018182	13.623888
10.000000	11	110.000000	2.000000	16.500000	0.000880	100000	5.234809	0.802180	0.197820	0.909091	0.090909	1.850836
10.000000	11	110.000000	2.000000	16.500000	0.007910	100000	5.205766	0.800550	0.199450	0.909091	0.090909	1.863630
10.000000	11	110.000000	2.000000	16.500000	0.188810	100000	5.190658	0.800480	0.199520	0.909091	0.090909	2.020349
10.000000	11	110.000000	2.000000	16.500000	0.204090	100000	5.224344	0.799950	0.200050	0.909091	0.090909	2.048757
10.000000	11	110.000000	2.000000	16.500000	0.321670	100000	5.213059	0.798600	0.201400	0.909091	0.090909	2.162043
10.000000	11	110.000000	2.000000	16.500000	0.405130	100000	5.183575	0.800300	0.199700	0.909091	0.090909	2.211940
10.000000	11	110.000000	2.000000	16.500000	1.457280	100000	5.219210	0.802290	0.197710	0.909091	0.090909	3.138783
10.000000	11	110.000000	2.000000	16.500000	3.230030	100000	5.199770	0.799970	0.200030	0.909091	0.090909	4.726086
10.000000	11	110.000000	2.000000	16.500000	7.014940	100000	5.194565	0.802020	0.197980	0.909091	0.090909	8.072438
10.000000	11	110.000000	2.000000	16.500000	10.649900	100000	5.134369	0.801070	0.198930	0.909091	0.090909	11.298894
10.000000	11	110.000000	2.000000	16.500000	11.472100	100000	5.250817	0.801210	0.198790	0.909091	0.090909	12.040070
10.000000	11	110.000000	2.000000	16.500000	11.532000	100000	5.211668	0.802190	0.197810	0.909091	0.090909	12.086939
10.000000	11	110.000000	2.000000	16.500000	11.631400	100000	5.222630	0.798320	0.201680	0.909091	0.090909	12.186854
10.000000	11	110.000000	2.000000	16.500000	11.673400	100000	5.172374	0.801130	0.198870	0.909091	0.090909	12.212005

Figure 5. Complete Tabular Output File – Two-Percent Subsidence Case

Att.

Attachment 1

Design-Check Version (Rev. 5a) of Python Source Code for Revised Probabilistic Model

```
#!/bin/env python

import sys
import random
import numpy

print "Running:", sys.argv[0]

### Read filenames
prefixFile             = sys.argv[1]  # prefix for output files
compartmentsSize = float(sys.argv[2]) # size of compartment (feet)
compartmentsTotal  = int(sys.argv[3]) # total number of compartments
percentSubsided  = float(sys.argv[4]) # percent subsidence
F_notET          = float(sys.argv[5]) # annual average rainfall minus evapotranspiration
F_intact         = float(sys.argv[6]) # intact infiltration rate
realizations       = int(sys.argv[7]) # (number of Monte Carlo realizations)
debugArg               = sys.argv[8]  # debug flag (true or false)
graphicArg             = sys.argv[9]  # graphic flag for .out file (>>>>>>O>>>>>>)
appendFlag            = sys.argv[10]  # append flag for summary file (w for overwrite or a for append)

slopeLength = compartmentsSize*float(compartmentsTotal)

F_netRunoff = F_notET - F_intact

outputFile = prefixFile + ".out"
summaryFile = prefixFile + ".sum"
tabFile = prefixFile + ".tab"

if debugArg == "True":
    debugFlag = 1
else:
    debugFlag = 0

if graphicArg == "True":
    graphicFlag = 1
else:
    graphicFlag = 0

print "    compartmentsSize:", compartmentsSize
print "   compartmentsTotal:", compartmentsTotal
print "         slopeLength:", slopeLength
```

```
print "     percentSubsided:", percentSubsided
print "             F_notET:", F_notET
print "            F_intact:", F_intact
print "        realizations:", realizations
print "          outputFile:", outputFile

output = open(outputFile, appendFlag)
output.write("                 Compartment size: %f\n" % (compartmentsSize))
output.write("     Total number of compartments: %d\n" % (compartmentsTotal))
output.write("                     Slope length: %f\n" % (slopeLength))
output.write("    Percent subsided compartments: %f\n" % (percentSubsided))
output.write("               Flux, intact cover: %f\n" % (F_intact))
output.write("             Flux, subsided cover: %f\n" % (F_notET))
output.write("                     Realizations: %d\n" % (realizations))

summary = open(summaryFile, appendFlag)
summary.write("                 Compartment size: %f\n" % (compartmentsSize))
summary.write("     Total number of compartments: %d\n" % (compartmentsTotal))
summary.write("                     Slope length: %f\n" % (slopeLength))
summary.write("    Percent subsided compartments: %f\n" % (percentSubsided))
summary.write("               Flux, intact cover: %f\n" % (F_intact))
summary.write("             Flux, subsided cover: %f\n" % (F_notET))
summary.write("                     Realizations: %d\n" % (realizations))

tabdelimited = open(tabFile, appendFlag)


### Subsided trench compartments
if debugFlag or graphicFlag: output.write("================================================================================\n")
if debugFlag:
    output.write("Subsided compartments followed by upslope ratios\n")
elif graphicFlag:
    output.write("Subsided compartments\n")

compartments = range(1,compartmentsTotal+1)
if debugFlag: print "compartments:", compartments

stringIntact = []
for key in compartments:
    stringIntact.append(">")
if debugFlag: print stringIntact

realization = 0

countSubsided = 0
countTotal = 0
```

```
maxCompartments = 0
minCompartments = [compartmentsTotal]
avgCompartments = float(0)

slicesIntact = 0
slicesSubsided = 0
slicesTotal = 0

avgLength = float(0)
avgCount = 0

lengths = []

avgLengthSingleHole = float(0)
avgCountSingleHole = 0

lengthsSingleHole = []

infiltration = float(0)

precise = 1000

while realization < realizations:     #Monte Carlo loop
    realization = realization + 1
    if debugFlag: print realization

    ### Randomly place subsided compartments
    subsided = []
    localCount = 0
    for i in compartments:
        draw = float(random.uniform(0,100*precise))/float(precise)
        countTotal = countTotal + 1
        if draw < percentSubsided:
            countSubsided = countSubsided + 1
            subsided.append(i)
            localCount = localCount + 1
    maxCompartments = max(maxCompartments, localCount)
    minCompartments = min(minCompartments, localCount)
    avgCompartments = avgCompartments + float(localCount)

    if debugFlag: print "subsided:", subsided

    sortedSubsided = sorted(subsided, key=float)
    if debugFlag: print "sortedSubsided:", sortedSubsided
```

```
    stringSubsided = list(stringIntact)
    for key in sortedSubsided:
        stringSubsided[int(key)-1] = 'O'
    if debugFlag: print "stringSubsided:", "".join(stringSubsided)

    ### Tally subsided versus intact slices (realizations)
    slicesTotal = slicesTotal + 1
    if len(sortedSubsided) == 0:
        slicesIntact = slicesIntact + 1
    else:
        slicesSubsided = slicesSubsided + 1

    ### Compute upslope length for slices with holes
    lengthSubsided = []

    for i in range(len(sortedSubsided)):
        if i == 0:
            length = sortedSubsided[i] - 1
        else:
            length = sortedSubsided[i] - sortedSubsided[i-1] - 1
        lengths.append(length)

        if debugFlag: print "i, sortedSubsided[i], length:", i, sortedSubsided[i], length

        lengthSubsided.append(length)

    if debugFlag: print "lengthSubsided:", lengthSubsided

    if debugFlag: print >> output, sortedSubsided
    if graphicFlag: print >> output, "".join(stringSubsided)

    for i in range(len(lengthSubsided)):
        avgCount = avgCount + 1
        avgLength = avgLength + lengthSubsided[i]
        if debugFlag: output.write("%s\n" % (lengthSubsided[i]))

    ### Compute upslope length after consolidating to 1 hole
    if len(sortedSubsided) > 0:
        iBottomHole = len(sortedSubsided)-1
        lengthSingleHole = sortedSubsided[iBottomHole] - 1
        lengthsSingleHole.append(lengthSingleHole)

        if debugFlag: print "iBottomHole, sortedSubsided[iBottomHole], lengthSingleHole:", iBottomHole, sortedSubsided[iBottomHole], 
lengthSingleHole

        avgCountSingleHole = avgCountSingleHole + 1
```

```
        avgLengthSingleHole = avgLengthSingleHole + lengthSingleHole

        if debugFlag: output.write("%s single hole\n" % (lengthSingleHole))

    ### Compute infiltration
    infiltration = infiltration + float(compartmentsTotal - localCount)*F_intact

    for i in range(len(sortedSubsided)):
        if i == 0:
            length = sortedSubsided[i] - 1
        else:
            length = sortedSubsided[i] - sortedSubsided[i-1] - 1

        infiltration = infiltration + F_notET + length*F_netRunoff


### Compute statistics / proportions
avgCompartments = avgCompartments/float(realizations)

sampleSubsided = float(countSubsided)/float(countTotal)*100

avgLength = avgLength/float(avgCount)

medianLength = numpy.median(lengths)
meanLength = numpy.mean(lengths)
minLength = numpy.min(lengths)
maxLength = numpy.max(lengths)

avgLengthSingleHole = avgLengthSingleHole/float(avgCountSingleHole)

medianLengthSingleHole = numpy.median(lengthsSingleHole)
meanLengthSingleHole = numpy.mean(lengthsSingleHole)
minLengthSingleHole = numpy.min(lengthsSingleHole)
maxLengthSingleHole = numpy.max(lengthsSingleHole)

fractionSlicesIntact = float(slicesIntact)/float(slicesTotal)
fractionSlicesSubsided = float(slicesSubsided)/float(slicesTotal)

### Compute infiltration, assuming one hole per subsided slice

# Monte Carlo average
F_coverAvg = infiltration/(float(realizations)*float(compartmentsTotal))

# preparation
f_subsidedSliceIntact   = (float(compartmentsTotal) - 1.)/float(compartmentsTotal)
f_subsidedSliceSubsided =                              1./float(compartmentsTotal)
```

```
F_runonAvg = avgLengthSingleHole*F_netRunoff

# local
F_intactAvg = F_intact
F_subsidedAvg = F_notET + F_runonAvg

# aligned with slope
F_intactDownAvg = F_intactAvg
F_subsidedDownAvg = f_subsidedSliceIntact*F_intactAvg + f_subsidedSliceSubsided*F_subsidedAvg

F_coverDownAvg = fractionSlicesIntact*F_intactDownAvg + fractionSlicesSubsided*F_subsidedDownAvg

# transverse to slope
F_intactAcrossAvg = F_intactAvg
F_subsidedAcrossAvg = fractionSlicesIntact*F_intactAvg + fractionSlicesSubsided*F_subsidedAvg

F_coverAcrossAvg = f_subsidedSliceIntact*F_intactAcrossAvg + f_subsidedSliceSubsided*F_subsidedAcrossAvg

###write output to file
output.write("================================================================================\n")
output.write("Percent subsided/Avg upslope ratio\n")
output.write("\t%.2f/%.2f\n" % (percentSubsided, avgLength))

output.write("Percent subsided/Avg upslope ratio single hole\n")
output.write("\t%.2f/%.2f\n" % (percentSubsided, avgLengthSingleHole))

output.write("Sample subsided\n")
output.write("\t%.2f\n" % (sampleSubsided))

output.write("Min/avg/max compartments\n")
output.write("\t%d/%.2f/%d\n" % (minCompartments, avgCompartments, maxCompartments))

output.write("Min/median/mean/max length\n")
output.write("\t%.2f/%.2f/%.2f/%.2f\n" % (minLength, medianLength, meanLength, maxLength))

output.write("Min/median/mean/max single hole length\n")
output.write("\t%.2f/%.2f/%.2f/%.2f\n" % (minLengthSingleHole, medianLengthSingleHole, meanLengthSingleHole, maxLengthSingleHole))

output.write("Slices intact/subsided (%)\n")
output.write("\t%.2f/%.2f\n" % (fractionSlicesIntact*100, fractionSlicesSubsided*100))

output.write("Subsided slice intact/subsided (%)\n")
output.write("\t%.2f/%.2f\n" % (f_subsidedSliceIntact*100, f_subsidedSliceSubsided*100))

output.write("Fluxes notET/intact/runoff (in/yr)\n")
```

```
output.write("\t%f/%f/%f\n" % (F_notET,F_intact,F_netRunoff))

output.write("Fluxes intactAvg/subsidedAvg (in/yr)\n")
output.write("\t%f/%f\n" % (F_intactAvg,F_subsidedAvg))

output.write("Fluxes intactDownAvg/subsidedDownAvg/coverDownAvg (in/yr)\n")
output.write("\t%f/%f/%f\n" % (F_intactDownAvg,F_subsidedDownAvg,F_coverDownAvg))

output.write("Fluxes intactAcrossAvg/subsidedAcrossAvg/coverAcrossAvg (in/yr)\n")
output.write("\t%f/%f/%f\n" % (F_intactAcrossAvg,F_subsidedAcrossAvg,F_coverAcrossAvg))

output.write("Fluxes coverAvg (in/yr)\n")
output.write("\t%f\n" % (F_coverAvg))
output.write("\n\n================================================================================\n")

summary.write("================================================================================\n")
summary.write("Percent subsided/Avg upslope ratio\n")
summary.write("\t%.2f/%.2f\n" % (percentSubsided, avgLength))

summary.write("Percent subsided/Avg upslope ratio single hole\n")
summary.write("\t%.2f/%.2f\n" % (percentSubsided, avgLengthSingleHole))

summary.write("Sample subsided\n")
summary.write("\t%.2f\n" % (sampleSubsided))

summary.write("Min/avg/max compartments\n")
summary.write("\t%d/%.2f/%d\n" % (minCompartments, avgCompartments, maxCompartments))

summary.write("Min/median/mean/max length\n")
summary.write("\t%.2f/%.2f/%.2f/%.2f\n" % (minLength, medianLength, meanLength, maxLength))

summary.write("Min/median/mean/max single hole length\n")
summary.write("\t%.2f/%.2f/%.2f/%.2f\n" % (minLengthSingleHole, medianLengthSingleHole, meanLengthSingleHole, maxLengthSingleHole))

summary.write("Slices intact/subsided (%)\n")
summary.write("\t%.2f/%.2f\n" % (fractionSlicesIntact*100, fractionSlicesSubsided*100))

summary.write("Subsided slice intact/subsided (%)\n")
summary.write("\t%.2f/%.2f\n" % (f_subsidedSliceIntact*100, f_subsidedSliceSubsided*100))

summary.write("Fluxes notET/intact/runoff (in/yr)\n")
summary.write("\t%f/%f/%f\n" % (F_notET,F_intact,F_netRunoff))

summary.write("Fluxes intactAvg/subsidedAvg (in/yr)\n")
summary.write("\t%f/%f\n" % (F_intactAvg,F_subsidedAvg))
```

```
summary.write("Fluxes intactDownAvg/subsidedDownAvg/coverDownAvg (in/yr)\n")
summary.write("\t%f/%f/%f\n" % (F_intactDownAvg,F_subsidedDownAvg,F_coverDownAvg))

summary.write("Fluxes intactAcrossAvg/subsidedAcrossAvg/coverAcrossAvg (in/yr)\n")
summary.write("\t%f/%f/%f\n" % (F_intactAcrossAvg,F_subsidedAcrossAvg,F_coverAcrossAvg))

summary.write("Fluxes coverAvg (in/yr)\n")
summary.write("\t%f\n" % (F_coverAvg))
summary.write("\n\n=================================================================================\n")

###write output to screen
print ("percentSubsided/avgLength/avgLengthSingleHole: \n\t%.2f/%.2f/%.2f" % (percentSubsided, avgLength, avgLengthSingleHole))

print ("sampleSubsided: %.2f" % (sampleSubsided))

print ("Slices intact/subsided (%)")
print ("\t%.2f/%.2f" % (fractionSlicesIntact*100, fractionSlicesSubsided*100))

print ("Subsided slice intact/subsided (%)")
print ("\t%.2f/%.2f" % (f_subsidedSliceIntact*100, f_subsidedSliceSubsided*100))

print ("Fluxes notET/intact/runoff (in/yr)")
print ("\t%f/%f/%f" % (F_notET,F_intact,F_netRunoff))

print ("Fluxes intactAvg/subsidedAvg (in/yr)")
print ("\t%f/%f" % (F_intactAvg,F_subsidedAvg))

print ("Fluxes intactDownAvg/subsidedDownAvg/coverDownAvg (in/yr)")
print ("\t%f/%f/%f" % (F_intactDownAvg,F_subsidedDownAvg,F_coverDownAvg))

print ("Fluxes intactAcrossAvg/subsidedAcrossAvg/coverAcrossAvg (in/yr)")
print ("\t%f/%f/%f" % (F_intactAcrossAvg,F_subsidedAcrossAvg,F_coverAcrossAvg))

print ("Fluxes coverAvg (in/yr)")
print ("\t%f" % (F_coverAvg))

### write one-line, tab-delimited, results summary
tabdelimited.write("%f\t%d\t%f\t%f\t%f\t%f\t%d\t%f\t%f\t%f\t%f\t%f\t%f\n" %
(compartmentsSize,compartmentsTotal,slopeLength,percentSubsided,F_notET,F_intact,realizations,avgLengthSingleHole,fractionSlicesIntact,fr
actionSlicesSubsided,f_subsidedSliceIntact,f_subsidedSliceSubsided,F_coverAvg))
```

Attachment 2

Assumptions Used to Calculate Infiltration Rates for E-Area Intact and Subsidence Cases

General

- Reported infiltration rates at each time step on the infiltration-rate degradation curve will be average values based on the portion of the total cap area that overlies the waste footprint (i.e., 40-foot overhangs excluded). This is true for both the intact and subsidence cases, and is different from the original PORFLOW simulations where both an intact infiltration rate and a subsided-area-only (or hole-only) infiltration rate were provided. An intact infiltration rate will be applied for the 40-foot overhangs.

- Intact infiltration rates will not be adjusted for location along the sloped length of the cap (i.e., increasing rate from crest to base).

- For the low-percent subsidence cases (typically < 2%), hole (or compartment) size will be fixed at 10 feet in the infiltration model simulations to comply with the assumption of a minimum 10-foot hole size.

- For PORFLOW simulations, the cap-averaged infiltration rate for the subsided case of interest will be used to back-calculate the effective infiltration rate into the subsided "hole(s)" in PORFLOW, which in some cases may differ in number and size compared to the infiltration model.

- F-Area Tank Farm cap design bases, cap degradation assumptions, and material property assumptions will be used in the E-Area HELP model simulations, except where impacted by the 100-year shift in the timeline (i.e., changes in number of geomembrane defects due to differences in age of geomembrane when pine tree intrusion occurs).

- Vegetative cover is Bahia grass (no bamboo).

Intact Case

- A single intact infiltration case based on 2% slope and 585-foot slope length will be used for all E-Area disposal units. This case represents an upper bound for the proposed E-Area final closure cap when considering variability in percent slope and slope length and other HELP model parameter uncertainties.

Subsided Cases

- New/future trench units: Assume 2% subsidence based on input from E-Area operations.

- Closed trench units: Percent subsidence will be based on reported non-crushable content (area).

- Partially filled trench units: Percent subsidence will be based on the reported non-crushable area linearly extrapolated to 100% full.

- For closed and partially filled open trench units, unit-by-unit total footprint area as reported in Table 2 below will be used to calculate percent subsidence.

- For closed and partially filled open trench units, non-crushable content (area) and percent-filled values are reported in Table 3 below.

- Percent subsidence for closed units will be calculated by (non-crush area) / (total footprint area) x 100%

- Percent subsidence for partially filled units will be calculated by (non-crush area) / (total footprint area) / (fraction filled) x 100%

- New/future units: ST15, ST16, ST17, ST18, ST19, ST20, ST21 (all will be set at 2% subsidence)

- Closed units: ST5 and ET1

- Open units: ST6, ST7, ET2, and ST14

- Calculated percent subsidence

  ST5: 0.54%
  ET1: 0.00%
  ST6: 2.00%
  ST7: 0.64%
  ET2: 0.04%
  ST14: 0.56%

- Propose reducing number of cases to:

ST6, ST15-ST21:	2% subsidence
ET2:	0.04% subsidence
ST5, ST7, ST14:	0.6% subsidence
ET1:	Intact case

- Run Python probabilistic model two times for each percent-subsidence case above assuming slope lengths of 545 feet and 110 feet (represents the two sides of the PORFLOW cap transect). Calculate a slope-length-weighted cap-average infiltration rate curve for each percent subsidence case from the two probabilistic simulations.

- Four (4) sets of infiltration vs. time data will be provided for the vadose zone PORFLOW simulations: intact, 2% subsidence, 0.6% subsidence, and 0.04% subsidence.

Table 2. Calculated Areas for E-Area LLWF Disposal Units (from Rev4-E-Area_LLWF_Coordinates_and_Areas_27-Feb-2017.xlxs)

				corner #1		corner #2		corner #3		corner #4		corner #5	
Low-Level Waste Facility	Area (m^2)	Calculated Area (m^2)	Calculated Area (ft^2)	SRS N	SRS E	SRS N	SRS E	SRS N	SRS E	SRS N	SRS E	SRS N	SRS E
Slit Trench 1	9,568	9,581	103,126	N77434.5	E58157.1	N77318.4	E58263.6	N77757.8	E58750.0	N77874.2	E58644.9		
Slit Trench 2	9,568	9,586	103,184	N77307.1	E58273.7	N77190.1	E58379.2	N77630.4	E58865.8	N77746.9	E58760.6		
Slit Trench 3	9,568	9,569	102,995	N77179.9	E58389.5	N77063.7	E58495.1	N77503.4	E58981.9	N77619.6	E58876.3		
Slit Trench 4	9,568	9,566	102,969	N77052.6	E58505.1	N76936.4	E58610.7	N77375.8	E59097.2	N77492.3	E58991.9		
Slit Trench 5	9,568	9,564	102,947	N76674.9	E58856.7	N76558.4	E58961.9	N76998.1	E59448.8	N77114.5	E59343.6		
Slit Trench 6	9,568	9,568	102,984	N76548.4	E58971.0	N76431.8	E59076.2	N76871.5	E59563.0	N76988.0	E59457.8		
Slit Trench 7	9,568	9,570	103,012	N76419.6	E59087.3	N76303.1	E59192.5	N76742.7	E59679.4	N76859.3	E59574.1		
Slit Trench 8		9,567	102,973	N77804.0	E57809.6	N77877.7	E57671.0	N78456.9	E57979.0	N78383.2	E58117.6		
Slit Trench 9		9,567	102,973	N77797.4	E57822.0	N77723.7	E57960.6	N78376.6	E58130.0	N78302.9	E58268.6		
Slit Trench 10		8,692	93,557	N78278.6	E58271.6	N78204.9	E58410.2	N77752.4	E57991.7	N77678.7	E58130.3		
Slit Trench 11		7,455	80,242	N77622.1	E58085.5	N77503.1	E58188.0	N77955.7	E58472.6	N77836.8	E58575.1		
Slit Trench 14		9,568	102,989	N75830.9	E58944.8	N75673.9	E58944.8	N75673.9	E59600.8	N75830.9	E59600.8		
Slit Trench 15		9,564	102,949	N75659.6	E58945.4	N75502.6	E58945.9	N75502.6	E59601.4	N75659.6	E59601.4		
Slit Trench 16		8,726	93,928	N75315.0	E58941.5	N75170.0	E58941.5	N75170.0	E59589.3	N75315.0	E59589.3		
Slit Trench 17		8,726	93,928	N75155.0	E58941.5	N75010.0	E58941.5	N75010.0	E59589.3	N75155.0	E59589.3		
Slit Trench 18		8,726	93,928	N74995.0	E58941.5	N74850.0	E58941.5	N74850.0	E59589.3	N74995.0	E59589.3		
Slit Trench 19		8,726	93,928	N74835.0	E58941.5	N74690.0	E58941.5	N74690.0	E59589.3	N74835.0	E59589.3		
Slit Trench 20		8,726	93,928	N74675.0	E58941.5	N74530.0	E58941.5	N74530.0	E59589.3	N74675.0	E59589.3		
Slit Trench 21		15,668	168,651	N74515.0	E58941.5	N74140.0	E58941.5	N74140.0	E59391.5	N74515.0	E59391.0		
CIG Trench 1	9,568	9,866	106,192	N77362.3	E59110.0	N77246.1	E59215.6	N76805.0	E58730.0	N76921.2	E58624.4		
CIG Trench 2	9,568	9,566	102,965	N77125.7	E59333.5	N77242.2	E59228.3	N76686.0	E58846.7	N76802.5	E58741.5		
Engineered Trench #1	9,568	8,913	95,936	N75995.2	E58944.7	N75845.2	E58943.8	N75845.1	E59590.3	N75992.1	E59590.3		
Engineered Trench #2	9,568	9,559	102,895	N76286.7	E58946.3	N76127.6	E58946.0	N76129.0	E59601.9	N76283.3	E59603.7		
Engineered Trench #3		7,880	84,819	N78727.8	E57528.3	N78522.9	E57934.5	N78443.2	E57899.4	N78378.9	E57817.6	N78564.7	E57448.6
Engineered Trench #4		9,638	103,740	N77943.5	E57147.3	N77865.0	E57283.8	N78465.6	E57574.4	N78533.8	E57433.5		
LAW Vault	8,662	8,666	93,275	N75475.0	E59589.3	N75330.0	E59589.3	N75330.0	E58946.0	N75475.0	E58946.0		
IL Vault	1,256	1,262	13,582	N77790.4	E57679.7	N77748.1	E57657.0	N77658.1	E57928.4	N77615.4	E57905.6		
643-26E NRCDA	4,435	4,430	47,686	N78152.4	E57675.1	N78241.6	E57504.9	N78434.5	E57598.2	N78327.4	E57819.6		
643-7E NRCDA	1,226	546	5,878	N74311.6	E58333.3	N74310.5	E58425.0	N74418.9	E58426.6	N74424.0	E58396.4	N74347.9	E58370.6

Table 3. Non-Crushable Content and Percent-Filled Values for Closed and Partially Filled Open E-Area LLWF Trench Units

Trench ID	Capacity Volume[1] (m^3)	Volume Status[1] (m^3) (1/31/18)	Percent Filled[1] (1/31/18)	Trench Area[2] (m^2)	Trench Area[2] (ft^2)	No. of Waste Packages[3]	Non-Crush Area[3] (ft^2)	Non-Crush Area[3] (%)
ET #1	35,660	35,660	100.0%	8,913	95,939	0	0	0.00%
ET #2	35,500	27,252	76.8%	9,559	102,892	3	34	0.03%
ET #3[5]	27,000	14,417	53.4%	7,511	80,848	0	0	0.00%
SLIT1	14,264	14,264	100.0%	9,581	65,000	0	0	0.00%
SLIT2	15,560	15,560	100.0%	9,586	65,000	9	2,046	3.15%
SLIT3	16,953	16,953	100.0%	9,569	65,000	28	5,019	7.72%
SLIT4	19,193	19,193	100.0%	9,566	65,000	33	3,710	5.71%
SLIT5	28,125	28,125	100.0%	9,564	65,000	18	557	0.86%
SLIT6	23,000	20,848	90.6%	9,568	65,000	49	1,866	2.87%
SLIT7	15,900	10,555	66.4%	9,570	65,000	12	435	0.67%
SLIT8	16,275	15,461	95.0%	9,567	65,000	1	13	0.02%
SLIT9	21,000	18,409	87.7%	9,567	65,000	0	0	0.00%
SLIT14	19,500	12,852	65.9%	9,568	65,000	1	383	0.59%
SLIT15	new							
SLIT16	new							
SLIT20	new							

Page Intentionally Blank

Attachment 3

Corrected Version (Rev. 6) of Python Source Code for Revised Probabilistic Model

```
#!/bin/env python

import sys
import random
import numpy

print "Running:", sys.argv[0]

### Read filenames
prefixFile             = sys.argv[1]  # prefix for output files
compartmentsSize = float(sys.argv[2]) # size of compartment (feet)
compartmentsTotal  = int(sys.argv[3]) # total number of compartments
percentSubsided  = float(sys.argv[4]) # percent subsidence
F_notET          = float(sys.argv[5]) # annual average rainfall minus evapotranspiration
F_intact         = float(sys.argv[6]) # intact infiltration rate
realizations       = int(sys.argv[7]) # (number of Monte Carlo realizations)
debugArg               = sys.argv[8]  # debug flag (true or false)
graphicArg             = sys.argv[9]  # graphic flag for .out file (>>>>>>O>>>>>>)
appendFlag            = sys.argv[10]  # append flag for summary file (w for overwrite or a for append)

slopeLength = compartmentsSize*float(compartmentsTotal)

F_netRunoff = F_notET - F_intact

outputFile = prefixFile + ".out"
summaryFile = prefixFile + ".sum"
tabFile = prefixFile + ".tab"

if debugArg == "True":
    debugFlag = 1
else:
    debugFlag = 0

if graphicArg == "True":
    graphicFlag = 1
else:
    graphicFlag = 0
```

```
print "    compartmentsSize:", compartmentsSize
print "   compartmentsTotal:", compartmentsTotal
print "         slopeLength:", slopeLength
print "     percentSubsided:", percentSubsided
print "             F_notET:", F_notET
print "            F_intact:", F_intact
print "        realizations:", realizations
print "          outputFile:", outputFile

output = open(outputFile, appendFlag)
output.write("                 Compartment size: %f\n" % (compartmentsSize))
output.write("     Total number of compartments: %d\n" % (compartmentsTotal))
output.write("                     Slope length: %f\n" % (slopeLength))
output.write("    Percent subsided compartments: %f\n" % (percentSubsided))
output.write("               Flux, intact cover: %f\n" % (F_intact))
output.write("             Flux, subsided cover: %f\n" % (F_notET))
output.write("                     Realizations: %d\n" % (realizations))

summary = open(summaryFile, appendFlag)
summary.write("                 Compartment size: %f\n" % (compartmentsSize))
summary.write("     Total number of compartments: %d\n" % (compartmentsTotal))
summary.write("                     Slope length: %f\n" % (slopeLength))
summary.write("    Percent subsided compartments: %f\n" % (percentSubsided))
summary.write("               Flux, intact cover: %f\n" % (F_intact))
summary.write("             Flux, subsided cover: %f\n" % (F_notET))
summary.write("                     Realizations: %d\n" % (realizations))

tabdelimited = open(tabFile, appendFlag)


### Subsided trench compartments
if debugFlag or graphicFlag: output.write("================================================================================\n")
if debugFlag:
    output.write("Subsided compartments followed by upslope ratios\n")
elif graphicFlag:
    output.write("Subsided compartments\n")

compartments = range(1,compartmentsTotal+1)
if debugFlag: print "compartments:", compartments

stringIntact = []
for key in compartments:
    stringIntact.append(">")
if debugFlag: print stringIntact

realization = 0
```

```
countSubsided = 0
countTotal = 0

maxCompartments = 0
minCompartments = [compartmentsTotal]
avgCompartments = float(0)

slicesIntact = 0
slicesSubsided = 0
slicesTotal = 0

avgLength = float(0)
avgCount = 0

lengths = []

avgLengthSingleHole = float(0)
avgCountSingleHole = 0

lengthsSingleHole = []

infiltration = float(0)

precise = 1000

while realization < realizations:     #Monte Carlo loop
    realization += 1
    if debugFlag: print realization

    ### Randomly place subsided compartments
    subsided = []
    localCount = 0
    for i in compartments:
        draw = float(random.uniform(0,100*precise))/float(precise)
        countTotal += 1
        if draw < percentSubsided:
            countSubsided += 1
            subsided.append(i)
            localCount += 1
    maxCompartments = max(maxCompartments, localCount)
    minCompartments = min(minCompartments, localCount)
    avgCompartments = avgCompartments + float(localCount)

    if debugFlag: print "subsided:", subsided
```

```
    sortedSubsided = sorted(subsided, key=float)
    if debugFlag: print "sortedSubsided:", sortedSubsided

    stringSubsided = list(stringIntact)
    for key in sortedSubsided:
        stringSubsided[int(key)-1] = 'O'
    if debugFlag: print "stringSubsided:", "".join(stringSubsided)

    ### Tally subsided versus intact slices (realizations)
    slicesTotal += 1
    if len(sortedSubsided) == 0:
        slicesIntact += 1
    else:
        slicesSubsided += 1

    ### Compute upslope length for slices with holes
    lengthSubsided = []

    for i in range(len(sortedSubsided)):
        if i == 0:
            length = sortedSubsided[i] - 1
        else:
            length = sortedSubsided[i] - sortedSubsided[i-1] - 1
        lengths.append(length)

        if debugFlag: print "i, sortedSubsided[i], length:", i, sortedSubsided[i], length

        lengthSubsided.append(length)

    if debugFlag: print "lengthSubsided:", lengthSubsided

    if debugFlag: print >> output, sortedSubsided
    if graphicFlag: print >> output, "".join(stringSubsided)

    for i in range(len(lengthSubsided)):
        avgCount += 1
        avgLength += lengthSubsided[i]
        if debugFlag: output.write("%s\n" % (lengthSubsided[i]))

    ### Compute upslope length after consolidating to 1 hole
    if len(sortedSubsided) > 0:
        iBottomHole = len(sortedSubsided)-1
        lengthSingleHole = sortedSubsided[iBottomHole] - 1
        lengthsSingleHole.append(lengthSingleHole)
```

```
        if  debugFlag:  print  "iBottomHole,  sortedSubsided[iBottomHole],  lengthSingleHole:",  iBottomHole,  sortedSubsided[iBottomHole],
lengthSingleHole

        avgCountSingleHole += 1
        avgLengthSingleHole += lengthSingleHole

        if debugFlag: output.write("%s single hole\n" % (lengthSingleHole))

    ### Compute infiltration
    infiltration = infiltration + float(compartmentsTotal - localCount)*F_intact

    for i in range(len(sortedSubsided)):
        if i == 0:
            length = sortedSubsided[i] - 1
        else:
            length = sortedSubsided[i] - sortedSubsided[i-1] - 1

        infiltration += F_notET + length*F_netRunoff


### Compute statistics / proportions
avgCompartments /= float(realizations)

sampleSubsided = float(countSubsided)/float(countTotal)*100

if avgCount > 0:
    avgLength /= float(avgCount)
else:
    avgLength = -999

if len(lengths) > 0:
    medianLength = numpy.median(lengths)
    meanLength = numpy.mean(lengths)
    minLength = numpy.min(lengths)
    maxLength = numpy.max(lengths)
else:
    medianLength = -999
    meanLength = -999
    minLength = -999
    maxLength = -999

if avgCountSingleHole > 0:
    avgLengthSingleHole /= float(avgCountSingleHole)
else:
    avgLengthSingleHole = -999
```

```
if len(lengthsSingleHole) > 0:
    medianLengthSingleHole = numpy.median(lengthsSingleHole)
    meanLengthSingleHole = numpy.mean(lengthsSingleHole)
    minLengthSingleHole = numpy.min(lengthsSingleHole)
    maxLengthSingleHole = numpy.max(lengthsSingleHole)
else:
    medianLengthSingleHole = -999
    meanLengthSingleHole = -999
    minLengthSingleHole = -999
    maxLengthSingleHole = -999

fractionSlicesIntact = float(slicesIntact)/float(slicesTotal)
fractionSlicesSubsided = float(slicesSubsided)/float(slicesTotal)

### Compute infiltration, assuming one hole per subsided slice

# Monte Carlo average
F_coverAvg = infiltration/(float(realizations)*float(compartmentsTotal))

# preparation
f_subsidedSliceIntact   = (float(compartmentsTotal) - 1.)/float(compartmentsTotal)
f_subsidedSliceSubsided =                              1./float(compartmentsTotal)

F_runonAvg = avgLengthSingleHole*F_netRunoff

# local
F_intactAvg = F_intact
F_subsidedAvg = F_notET + F_runonAvg

# aligned with slope
F_intactDownAvg = F_intactAvg
F_subsidedDownAvg = f_subsidedSliceIntact*F_intactAvg + f_subsidedSliceSubsided*F_subsidedAvg

F_coverDownAvg = fractionSlicesIntact*F_intactDownAvg + fractionSlicesSubsided*F_subsidedDownAvg

# transverse to slope
F_intactAcrossAvg = F_intactAvg
F_subsidedAcrossAvg = fractionSlicesIntact*F_intactAvg + fractionSlicesSubsided*F_subsidedAvg

F_coverAcrossAvg = f_subsidedSliceIntact*F_intactAcrossAvg + f_subsidedSliceSubsided*F_subsidedAcrossAvg

###write output to file
output.write("================================================================================\n")
output.write("Percent subsided/Avg upslope ratio\n")
output.write("\t%.2f/%.2f\n" % (percentSubsided, avgLength))
```

```
output.write("Percent subsided/Avg upslope ratio single hole\n")
output.write("\t%.2f/%.2f\n" % (percentSubsided, avgLengthSingleHole))

output.write("Sample subsided\n")
output.write("\t%.2f\n" % (sampleSubsided))

output.write("Min/avg/max compartments\n")
output.write("\t%d/%.2f/%d\n" % (minCompartments, avgCompartments, maxCompartments))

output.write("Min/median/mean/max length\n")
output.write("\t%.2f/%.2f/%.2f/%.2f\n" % (minLength, medianLength, meanLength, maxLength))

output.write("Min/median/mean/max single hole length\n")
output.write("\t%.2f/%.2f/%.2f/%.2f\n" % (minLengthSingleHole, medianLengthSingleHole, meanLengthSingleHole, maxLengthSingleHole))

output.write("Slices intact/subsided (%)\n")
output.write("\t%.2f/%.2f\n" % (fractionSlicesIntact*100, fractionSlicesSubsided*100))

output.write("Subsided slice intact/subsided (%)\n")
output.write("\t%.2f/%.2f\n" % (f_subsidedSliceIntact*100, f_subsidedSliceSubsided*100))

output.write("Fluxes notET/intact/runoff (in/yr)\n")
output.write("\t%f/%f/%f\n" % (F_notET,F_intact,F_netRunoff))

output.write("Fluxes intactAvg/subsidedAvg (in/yr)\n")
output.write("\t%f/%f\n" % (F_intactAvg,F_subsidedAvg))

output.write("Fluxes intactDownAvg/subsidedDownAvg/coverDownAvg (in/yr)\n")
output.write("\t%f/%f/%f\n" % (F_intactDownAvg,F_subsidedDownAvg,F_coverDownAvg))

output.write("Fluxes intactAcrossAvg/subsidedAcrossAvg/coverAcrossAvg (in/yr)\n")
output.write("\t%f/%f/%f\n" % (F_intactAcrossAvg,F_subsidedAcrossAvg,F_coverAcrossAvg))

output.write("Fluxes coverAvg (in/yr)\n")
output.write("\t%f\n" % (F_coverAvg))
output.write("\n\n===============================================================================\n")

summary.write("================================================================================\n")
summary.write("Percent subsided/Avg upslope ratio\n")
summary.write("\t%.2f/%.2f\n" % (percentSubsided, avgLength))

summary.write("Percent subsided/Avg upslope ratio single hole\n")
summary.write("\t%.2f/%.2f\n" % (percentSubsided, avgLengthSingleHole))

summary.write("Sample subsided\n")
summary.write("\t%.2f\n" % (sampleSubsided))
```

```
summary.write("Min/avg/max compartments\n")
summary.write("\t%d/%.2f/%d\n" % (minCompartments, avgCompartments, maxCompartments))

summary.write("Min/median/mean/max length\n")
summary.write("\t%.2f/%.2f/%.2f/%.2f\n" % (minLength, medianLength, meanLength, maxLength))

summary.write("Min/median/mean/max single hole length\n")
summary.write("\t%.2f/%.2f/%.2f/%.2f\n" % (minLengthSingleHole, medianLengthSingleHole, meanLengthSingleHole, maxLengthSingleHole))

summary.write("Slices intact/subsided (%)\n")
summary.write("\t%.2f/%.2f\n" % (fractionSlicesIntact*100, fractionSlicesSubsided*100))

summary.write("Subsided slice intact/subsided (%)\n")
summary.write("\t%.2f/%.2f\n" % (f_subsidedSliceIntact*100, f_subsidedSliceSubsided*100))

summary.write("Fluxes notET/intact/runoff (in/yr)\n")
summary.write("\t%f/%f/%f\n" % (F_notET,F_intact,F_netRunoff))

summary.write("Fluxes intactAvg/subsidedAvg (in/yr)\n")
summary.write("\t%f/%f\n" % (F_intactAvg,F_subsidedAvg))

summary.write("Fluxes intactDownAvg/subsidedDownAvg/coverDownAvg (in/yr)\n")
summary.write("\t%f/%f/%f\n" % (F_intactDownAvg,F_subsidedDownAvg,F_coverDownAvg))

summary.write("Fluxes intactAcrossAvg/subsidedAcrossAvg/coverAcrossAvg (in/yr)\n")
summary.write("\t%f/%f/%f\n" % (F_intactAcrossAvg,F_subsidedAcrossAvg,F_coverAcrossAvg))

summary.write("Fluxes coverAvg (in/yr)\n")
summary.write("\t%f\n" % (F_coverAvg))
summary.write("\n\n================================================================================\n")

###write output to screen
print ("percentSubsided/avgLength/avgLengthSingleHole: \n\t%.2f/%.2f/%.2f" % (percentSubsided, avgLength, avgLengthSingleHole))

print ("sampleSubsided: %.2f" % (sampleSubsided))

print ("Slices intact/subsided (%)")
print ("\t%.2f/%.2f" % (fractionSlicesIntact*100, fractionSlicesSubsided*100))

print ("Subsided slice intact/subsided (%)")
print ("\t%.2f/%.2f" % (f_subsidedSliceIntact*100, f_subsidedSliceSubsided*100))

print ("Fluxes notET/intact/runoff (in/yr)")
print ("\t%f/%f/%f" % (F_notET,F_intact,F_netRunoff))
```

```
print ("Fluxes intactAvg/subsidedAvg (in/yr)")
print ("\t%f/%f" % (F_intactAvg,F_subsidedAvg))

print ("Fluxes intactDownAvg/subsidedDownAvg/coverDownAvg (in/yr)")
print ("\t%f/%f/%f" % (F_intactDownAvg,F_subsidedDownAvg,F_coverDownAvg))

print ("Fluxes intactAcrossAvg/subsidedAcrossAvg/coverAcrossAvg (in/yr)")
print ("\t%f/%f/%f" % (F_intactAcrossAvg,F_subsidedAcrossAvg,F_coverAcrossAvg))

print ("Fluxes coverAvg (in/yr)")
print ("\t%f" % (F_coverAvg))

### write one-line, tab-delimited, results summary
tabdelimited.write("%f\t%d\t%f\t%f\t%f\t%f\t%d\t%f\t%f\t%f\t%f\t%f\t%f\n" %
(compartmentsSize,compartmentsTotal,slopeLength,percentSubsided,F_notET,F_intact,realizations,avgLengthSingleHole,fractionSlicesIntact,fr
actionSlicesSubsided,f_subsidedSliceIntact,f_subsidedSliceSubsided,F_coverAvg))
```

c:

S. E. Aleman, 735-A
B. T. Butcher, 773-42A
D. A. Crowley, 773-42A
T. L. Danielson, 703-41A
K. L. Dixon, 773-42A
J. A. Dyer, 773-42A
G. P. Flach, 773-42A
L. L. Hamm, 735-A
T. Hang, 773-42A
L. T. Reid, 773-A
T. Whiteside, 773-42A
J. L. Wohlwend, 703-41A
EM File, 773-42A – Rm. 243