Monte Carlo Simulation Model Execution/Control using Unix/Linux

Monte Carlo simulation allows investigating changes in decisions in an engineering design problem resulting from treating a parameter or input in a simulation model as a random variable. If the simulation model can be "packaged" as a subroutine that can be called from a main control program, the required logic and programming is relatively simple. Unfortunately, many times the simulation program is either not available in source code form, or is simply not easily converted to a subroutine form. In this situation, implementing a Monte Carlo simulation exercise is significantly more complicated since the control logic must be shifted to a series of programs and operating system script files.

In the example below, the assumption is that the simulation model is a freestanding program, and can not be converted to a subroutine. It is also assumed that a single input variable will be changed in each iteration, or pass of the simulation model. Specifically, suppose that the variable to be treated as a random variable is the concentration of a contaminant entering at a node in the groundwater model SUTRA. Assume it is desired to execute the simulation model *ntimes*, each time with a different value of the contaminant concentration.

Before the actual simulation exercise is started, a file is created by the user that contains all of the simulation parameter values to be used in the entire Monte Carlo exercise. In this case, the file would have at least *ntimes* lines, with one number (the contaminate concentration) on each line. The numbers in the file would usually be created by a Fortran program that produced random contaminant concentration values based on some specified probability distribution. This file will be denoted here as the `random_value_file`.

It is also necessary to create a base, or template input data file for the simulation model. This file should be a complete input file, and the simulation model should be tested against this file before continuing. The value of the random variable used in this file is not important, but a reasonable choice would be the nominal or average value of the random variable. The line of the input file on which the random variable value appears will be denoted `targetline` and the Fortran format that the simulation model uses to read this line is denoted `targetformat`.

One file used in the process is created automatically by the Unix shell program discussed below. This file, denoted the `iteration` file contains a single number, the current iteration value. For example, on the first iteration, the `iteration` file will contain the number `1`. On iteration 30, the `iteration` file will contain the number 30.

In the approach given here, two additional Fortran programs need to be written. The first program, denoted `makeinput`, prepares a new input file for the simulation model for each iteration. The program uses the template data file as the basis for the new data file, just changing the `targetline`. The basic logic of this program is to read all of the lines up to the `targetline` and write them out to the new simulation model data file. The `targetline` containing (among other things) the base value of the random parameter is read using `targetformat`, but it is not yet written out to the new data file.

The `iteration` file is then read, giving the iteration number. The `makeinput` program then reads the new value of the random variable from the line of the `random_value_file` corresponding to the iteration number. The `targetline` is now written out to the new data file using `targetformat`, with the new random value of the parameter used instead of the existing base or nominal value. Finally, the remaining lines of the template data file are copied to the new data file and the `makeinput` program ends.

The other program that needs to be written is used to process the output from the simulation model. Typically, only a few lines of the simulation model output are to be "saved" from one iteration to the next. Those lines must be read in and appended to the other lines saved from previous iterations. After all iterations of the simulation model have been completed,

the file containing the important results from each iteration can be processed to determine the effect of the random variable on model output and on the decisions made in the design problem.

The entire Monte Carlo simulation exercise is controlled by a Unix shell script (command file). The script is contained in a file that is created using a text editor such as kate or vi. The shell script contains a loop that executes all of the required programs discussed above the desired number of times in the following order:

1. The `makeinput` (Fortran) program that prepares an input file that will be used by the simulation model each iteration. Usually, the input file only differs in a single place from one iteration to the next (i.e., we select a new input data value from some random distribution).
2. The simulation model itself.
3. A program that processes the output from the simulation model. Usually only a few numbers are needed from the simulation output for each iteration. These values are read in and appended into some other file (the `savefile`). Note that the shell script assumes that this program creates a new file containing all of the saved results from previous iterations and with the important results from the current iteration appended to the end. The shell script will delete the old `savefile` and rename the new one prior to starting a new iteration.

A sample of a shell script file is given below. After creating the file, be sure to change the permissions to include execute permission using the command `chmod +x file_name`. To "execute"the script, just type the name of the file at the command line.

```
#!/bin/bash

#Fill in the required numbers and file names below
ntimes=20 #number of times through the loop
iteration="iteration.dat"
makeinput="control"  #name of fortran executable that creates new input file
model="sutra"  #simulation model executable name
processoutput="sutraout" #executable program that processes model output
savefile="abc.monte"
savetemp="abc.tmp"
echo "This script will loop $ntimes times"
echo "Each iteration, the input data file will be created by program $makeinput"
echo "Then the model $model will be executed"
echo "Finally the program $processoutput will be run to process model output data"
echo
i=1 #initalize loop index
while [ $i -le $ntimes ]  #do while i <= ntimes
do
  echo $i > $iteration  # write the iteration number out to a file
  "$makeinput" #execute the fortran program that create the data new file
  "$model"   #execute the simulation model
  "$processoutput" #run ouput processor program
  rm "$savefile"  #delete the old Monte Carlo summary file
  mv "$savetemp" "$savefile" #rename the new summary file created by processoutput
  i=$(($i+1))  #increment the loop index
done
echo "All done!"
```

A sample Fortran program that satisfies the requirements of the `makeinput` program is given below. The program is set up to use the template data file `abc-template.dat` to create the new data file named `abc.dat` by changing line 694 (the value of `targetline`). For the

curious, the file `abc-template.dat` is just a copy of the SUTRA data file `rocky.d5`, and the value being changed is the concentration of the contaminant at the source point.

```
program control
  implicit none
  integer,parameter::targetline=694
  integer::eof,i,iteration,node
  double precision::x1,x2
  character (len=30),parameter::fnin="abc-template.dat",fnout="abc.dat", &
                                fnvalues="randomvalues.dat", &
                                fniteration="iteration.dat"
  character (len=3), parameter::fmt="(a)"
  character (len=30),parameter::targetformat="(i10,2e15.5)"
  character (len=80)::line
  open(11,file=fnin, status="old")  ! open the source template data file
  open(21,file=fnout) ! open the new data file
  ! Copy the lines up to the target line to the new data file
  do i=1,targetline-1
    read(11,fmt)line
    write(21,fmt)line
  end do
  ! Read in the target line, modify the value of x2, and write it out
  read(11,targetformat)node,x1,x2
  ! The iteration file is a single line file with a single number.
  ! The file is created by the Unix shell file running the show.
  open(12,file=fniteration)
  read(12,*)iteration
  close(12)
  ! The random values file is created before this entire process starts.
  ! It contains the values of the random variable to be used for each
  ! iteration of the simulation model.
  open(12,file=fnvalues)
  ! Advance to the correct entry
  do i=1,iteration-1
    read(12,fmt)line
  end do
  read(12,*)x2
  close(12)
  write(21,targetformat)node,x1,x2
  ! Copy the rest of the lines from the template file to the new data file
  do
    read(11,fmt,iostat=eof)line
    if(eof/=0)exit
    write(21,fmt)line
  end do
  close (11)
  close (21)
  !The following write statement is just to keep track of what is happening
  write(*,*)"Interation ",iteration, "Leaving control with x2 = ",x2
  stop
end program control
```

Finally, a sample Fortran program that illustrates how an output process program that saves important results from each iteration might work.

```
program sutraout
  implicit none
  integer, parameter::nsets=3
  integer::eof,i,j,where
```

```
  integer, dimension(nsets)::endline,startline
  character (len=3), parameter::fmt="(a)"
  character (len=30),parameter::fnin="abc.d6",fnout="abc.monte", fntemp="abc.tmp"
  character (len=140)::line

  !nsets is the number of sets (or groups) of lines that we want to append
  !to the end of fnout.  Each set of lines has a specified starting and
  !ending line number.  In some cases, a more sophisticated search scheme
  !is required to find the desired lines.

  !fnin is the file containing the full simulation model output
  !fnout is the file containing just those lines of output needed for each iteration
  !fntemp is a copy of fnout, with the new lines from fnin appended to the end.

  !The following are the assigned start and end lines for each set
  startline(1)=40
  endline(1)=40
  startline(2)=100
  endline(2)=104
  startline(3)=504
  endline(3)=505

  !The basic strategy is to copy the existing lines of the fnout to fntemp.
  !Then the desired lines of fnin are read and added to the end of fntemp.
  !The Unix shell program will delete fnout and rename fntemp to fnout after
  !this program is finished.

  open(11,file=fnin,status="old")
  open(12,file=fnout)
  open(21,file=fntemp)

  !Copy the existing lines of saved output
  do
    read(12,fmt,iostat=eof)line
    if(eof/=0)exit
    write(21,fmt)line
  end do

  !Read in the new sets of simulation model output that we need to save
  do i=1,nsets
    if(i==1)then
     where=1
    else
     where=endline(i-1)+1
    end if
    do j=where,startline(i)-1
      read(11,fmt)line
    end do
    do j=startline(i),endline(i)
      read(11,fmt)line
      write(21,fmt)line
    end do
  end do

  close(11)
  close(12)
  close(21)
  stop
end program sutraout
```