7/24/01

Infiltration Maps for Modern, Monsoon, and Glacial Transition Climates

Debra Hughson and Chandrika Manepally requested shallow infiltration results from the CNWRA/NRC code for input into their thermal modeling. They wanted shallow infiltration data for the locations of 1D columns modeled by Tom Buscheck of LLNL and used for the DOE TSPA simulations. Hughson and Manepally wanted me to give them maps of mean case, lower-bound, and upper-bound for each of the three climates: modern, monsoon, and glacial transition.

Since the NRC/CNWRA TPA code does not use stepped climate changes (an external file with a curve is used instead), the first step will be to determine precipitation and temperature values to use for each climate case and uncertainty bounds. The second step will be to extract net infiltration values for the TEF locations from the infiltration maps. To do this, I will modify the script I used to extract repository-average net infiltration values (Scientific Notebook #227, pages 11-17).

I will be running the ITYM preprocessor associated with last documented version of the TPA code (version 4.0) to obtain those maps. Hydraulic parameter uncertainty will be incorporated using the ITYM module of TPA 4.0 in Monte Carlo mode. The ITYM will be run on the SUN machine called vulcan. The source files and external files for ITYM were obtained from Ron Janetzke to ensure that the 4.0 version was used.

vulcan ~/ITYM_July2001/itym40/* original files from Ron Janetzke with original date stamps of ~ March 2000
./itym40-compile /* the compiled version (using the supplied makefile)
./simulations/* the simulations using the precip and temperatures for each climate
./simulations/DS9/* the simulations run on ds9 instead of vulcan
vulcan ~/Itym-Usage/TEF-July2001/* extraction of point/cell data from infiltration map files
bubo: J:\AVData\Repository\repository-tef.apr
bubo: J:\Itym-Usage\TEF-July2000*

The climate data will come from DOE since NRC does not formal reconstruct climate using CRWMS M&O (2000, infiltration AMR).

Modern upper-bound:	Desert Rock data, scaled to repository average of 4 mm/yr
Modern mean:	Desert Rock data, scaled to repository average of 8.5 mm/yr
Modern lower-bound:	Desert Rock data, scaled to repository average of 13 mm/yr
Monsoon upper-bound:	Hobbs, NM scaled by same factor as mean modern
Monsoon mean:	Average of lower- and upper-bound scaled monsoon results
Monsoon lower-bound:	Use scaled mean modern results
Glacial upper-bound:	Rosalia, WA scaled by same factor as mean modern
Glacial mean:	Average of lower- and upper-bound scaled glacial transition results
Glacial lower-bound:	Delta, UT scaled by same factor as mean modern

The Desert Rock precipitation and temperature, as applied to YM (precipitation is scaled for TPA), are taken directly from the TPA 4.0 documentation. The Hobbs, Rosalia, and Delta precipitation and temperature data are taken from tables in CRWMS M&O (2000, infiltration AMR). The scaling is based on the analysis and recommendation in Winterle et al. (1999). The mean and range of repository-averaged infiltration are inputs to the TPA tpa.inp input file. The range is intended to account for uncertainty in general, for the nebulous way vegetation is incorporated, snowmelt is not included, soil depths and textures remain constant over climate change, and the lack of a surface routing/runon module in the model. The monsoon and glacial transition mean cases are averages; this is the same approach used by DOE except that the lower- and upper-bounds are themselves averages of 2 or 3 sites. I chose Hobbs, Rosalia, and Delta to be conservative (expected greater net infiltration).

The steps to develop a table that has net infiltration uncertainty for each TEF location will be:

1. Run ITYM in Monte Carlo mode for each precipitation and temperature pair. 1000 realizations were run for each precip & temperature pair. Output is named with city, see vulcan ~/ITYM-July2001/simulations/*.
   - Desert Rock 162.8 mm/yr 17.4°C
   - Hobbs 417.6 mm/yr 16.8°C
   - Rosalia 459.7 mm/yr 8.4°C
   - Delta 197.9 mm/yr 10.1°C
2. Run maids.f to create scaled net infiltration maps created in step 1 for:
   - mean modern: modern.m
   - upper-bound monsoon: monsoon.ub
   - upper-bound glacial transition: glacial.ub
   - lower-bound glacial transition: glacial.lb
3. Cycle through steps 4,5,6 for each output map from step 2 to avoid writing over the output results (because the output files for scripts are named the same).
4. Run extract.f on output of maid.f to extract infiltration values for the TEF locations (outfile: tef.out)
5. Run extract-repository.f to check on repository-averaged net infiltration values as a consistency check (output file is named: summary-main.dat).
6. Concatenate (cat) tef.out and summary-main.dat (output files from steps 4 and 5, respectively) to get:
   - mean-modern.out
   - upper-monsoon.out
   - upper-glacial.out
   - lower-glacial.out
7. Put data from output files into spreadsheet and calculate remaining entries:
   - lower-bound modern: scale results from scaled mean modern by 1/8.5
   - upper-bound modern: scale results from scaled mean modern by 13/8.5
   - lower-bound monsoon: use scaled mean modern results directly
   - mean monsoon: calculate average of lower- and upper-bound scaled monsoon
   - mean glacial transition: calculate average of lower- and upper-bound scaled glacial transition

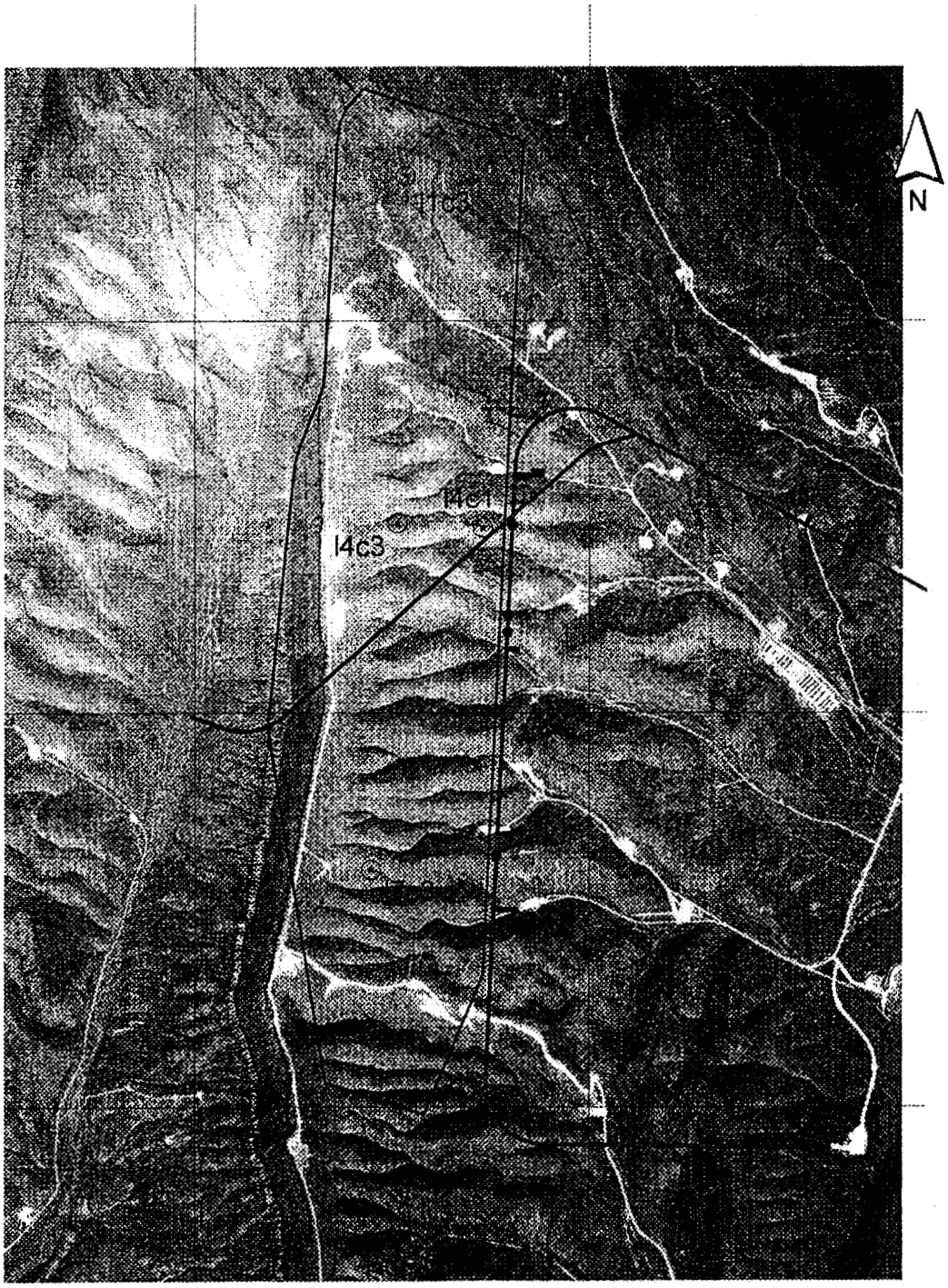
As usual use the UNIX commands dos2unix and unix2dos commands whenever transferring files between UNIX and WinNT.

The code in step 1 is controlled under TOP-018 as part of the TPA 4.0 code. The scripts in steps 2-5 perform simple data transformations; specifically, normalizing (scaling) an infiltration map and extracting an average infiltration from a portion of the infiltration map. All three scripts are included directly below. Visual checks on the output files indicate (tracking first few and last entries) that the scripts performed as designed. The step 5 check on the repository average also suggests that the maids.f and extract.f of steps 2 and 4 are performing as expected; note that extract-repository.f is a direct copy of a previously used (and checked) script, see Sci Ntbk #227, page 11-17. The maids.f and extract.f scripts were started from the extract-repository.f script, and do slightly different tasks. Hence, repository average should be the same between the latter two scripts; and this would also indicate that maids.f is reading and writing the maidtbl.dat format correctly.

The TEF locations are shown in the table below:

LLNL location name	tefd.txt location name	Nevada State Plane Easting, meters	Nevada State Plane Northing, meters	UTM NAD27 Easting, meters	UTM NAD27 Easting, meters
l4c3	1	170717.1	233795.7	548026	4078964
l4c1	2	171150.5	233772.7	548460	4078942
l1c3	3	170761.5	235559.9	548065	4080728
l7c3	4	170574.3	232036.8	547890	4077205

Three of the locations are on ridges and one is on a side slope (l4c3) as shown in the figure below.



Scripts

Two scripts, with many redundant aspects, are included below; maids.f and extract.f. Since extract-repository.f is a copy of a previously used script, it is not included here (see Sci Ntbk #227, page 11-17). The scripts have extensive commenting, hence no further description is included here.

maids.f script:

```
      program extract
c Calculates repository average from Desert Rock results, then scales
c Desert Rock to get mean, lower-bound, and upper bound modern climate.
c The repository average from Desert Rock is used to scale the Hobbs, Delta,
c and Rosalia results. Then the scaled Hobbs and mean modern are averaged to
c get the monsoonal mean, and the scaled Rosalia and Delta are averaged to
c get the glacial transition mean. These files are output for the extract.f
c script to extract TEF location shallow infiltration estimates.
c The EDA-II design is used for the repository outline, main block only.
c RFedors  July 24, 200?
c  vulcan~/Itym-Usage/TEF-July2001/maids.f
c23456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789 12
      integer ioread, iowrit, mx, i, j, k
      integer ndrift, i_max, i_min, ict, lf_rt
      parameter (mx=1000,mxx=100000)
      real*8 ymax, ytop, xbot, xpos, ypos, sum
      real*8 avg, stdev, avg_t, stdev_t, xsegment, ax,ay
      real*8 drift(mx,2), segment(mx,2)
      real*8 array(mxx,3), repository(mxx), reposit(mxx,3)
      character*9 flagside(mx), junk(6)
      character*60 header, headers(7)
      character*24 fdate
      real*8 xllcorner, yllcorner, cellsize
      real*8 tpa_avg, tpa_low, tpa_upp, scale, temp
      real*8 hobbs(mxx), delta(mxx), rosalia(mxx)

c set input and output unit numbers, and tpa mean, upper and lower bounds.
      ioread = 7
      iowrit = 8

      tpa_avg = 8.5d0
      tpa_low = 4.d0
      tpa_upp = 13.d0

c read in drift coords file, 1st line comment line, 2nd line # of points
c account for repeated entry of first point as last entry.
      open(unit = ioread, file = 'drift1.txt', form = 'formatted')
      read(ioread,'(a60)') header
      read(ioread,'(i5)') ndrift
      do i = 1, ndrift
        read(ioread,'(2f10.2)') drift(i,1), drift(i,2)
      enddo
      close(ioread)

c  set up usage of drift coordinates; checking to right or left of segment;
c  find min and max y-coord, then assign left/right to line segments
      ymax = 0.d0
      ymin = 4.d10
      do i = 1, ndrift-1
        if(drift(i,2).gt.ymax) then
          ymax = drift(i,2)
          i_max = i
        endif
        if(drift(i,2).lt.ymin) then
          ymin = drift(i,2)
          i_min = i
        endif
      enddo

      if(i_max.lt.i_min) then
          do i = 1, ndrift-1
            flagside(i) = 'left'
          enddo
          do i = i_max, i_min-1
            flagside(i) = 'right'
          enddo
        else
          do i = 1, ndrift-1
            flagside(i) = 'right'
          enddo
          do i = i_min, i_max-1
            flagside(i) = 'left'
          enddo
      endif

c calculate line segmen: equations going counter-clockwise;
c segment(i,1)=slope; segment(i,2)=intercept; for horizontal lines,
c set flagside to avoid checking either side of the segment and
c then set denominator to any number just to avoid blowout;
c for vertical lines (xbot=0), set numerator of slope to a small number.
      do i = 1, ndrift-1
        ytop = drift(i+1,2) - drift(i,2)
        xbot = drift(i+1,1) - drift(i,1)
        if(dabs(ytop).lt.1.d-9) then
            flagside(i)='neither'
            segment(i,1) = 1.d0
            segment(i,2) = 1.d0
          elseif(dabs(xbot).lt.1.d-10) then
            segment(i,1) = 1.d0
            segment(i,2) = 0.d0
          else
            segment(i,1) = ytop / xbot
            segment(i,2) = drift(i,2) - (segment(i,1)*drift(i,1))
        endif
      enddo
      do i = 1, ndrift-1
        print*, segment(i,1), segment(i,2)
      enddo

c read in Desert Rock DEM of infiltration; note that the coordinates of
c the southwest corner of the domain are given in the header, but the
c ordering of data is row-major starting from the northwest corner.
      open(unit=ioread,file='maidtbl.mean-modern1000',form='formatted')
      do i = 1, 4
        read(ioread,'(a60)') headers(i)
      enddo
      read(ioread,'(a9,i10)') junk(1), ncols
      read(ioread,'(a9,i10)') junk(2), nrows
      read(ioread,'(a9,f16.5)') junk(3), xllcorner
      read(ioread,'(a9,f16.5)') junk(4), yllcorner
      read(ioread,'(a9,f15.5)') junk(5), cellsize
      do i = 5, 7
        read(ioread,'(a60)') headers(i)
      enddo
      print*, ncols, nrows, cellsize, xllcorner, yllcorner

      ypos = yllcorner + cellsize * dfloat(nrows-1)
      xpos = xllcorner
      k = 1
      do i = 1, nrows
        do j = 1, ncols
          read(ioread,'(e15.8)') array(k,3)
          array(k,1) = xpos
          array(k,2) = ypos
          xpos = xpos + cellsize
          k = k + 1
        enddo
        ypos = ypos - cellsize
        xpos = xllcorner
      enddo
      read(ioread,'(a9)') junk(6)
      close(ioread)

c check to see if current position is within repository outline
      lf_rt = 0
      ict = 0
      do i = 1, nrows*ncols
        ay = array(i,2)
        ax = array(i,1)
        do m = 1, ndrift-1
          if(ay.le.drift(m,2).and.ay.gt.drift(m+1,2).or.
     &         ay.ge.drift(m,2).and.ay.lt.drift(m+1,2)) then
           xsegment = (array(i,2)-segment(m,2)) / segment(m,1)
           if(dabs(segment(m,2)).le.1.d-10) xsegment = drift(m,1)
           if(flagside(m).eq.'right'.and.ax.ge.xsegment) lf_rt= lf_rt+ 1
           if(flagside(m).eq.'left'.and.ax.le.xsegment) lf_rt= lf_rt + 1
          endif
          if(lf_rt.eq.2) then
                ict = ict + 1
                repository(ict) = array(i,3)
                reposit(ict,1) = array(i,1)
                reposit(ict,2) = array(i,2)
                reposit(ict,3) = array(i,3)
                lf_rt = 0
          endif
        enddo
        lf_rt = 0
      enddo

c statistics on repository cells
      open(unit=iowrit, file='summary-main.dat', form='formatted')
      sum = 0.d0
      do i = 1, ict
         sum = sum + repository(i)
      enddo
      avg = sum / dfloat(ict)

      sum = 0.d0
      do i = 1, ict
         sum = sum + dabs(repository(i) - avg)
      enddo
      stdev = dsqrt(sum/dfloat(ict-1))

      sum = 0.d0
      do i = 1, ncols*nrows
         sum = sum + array(i,3)
      enddo
      avg_t = sum / dfloat(nrows*ncols)

      sum = 0.d0
      do i = 1, ncols*nrows
        sum = sum + dabs(array(i,3) - avg_t)
      enddo
      stdev_t = dsqrt(sum/dfloat(nrows*ncols-1))

      write(iowrit,*) 'Number in Repository = ', ict
      write(iowrit,*) 'Average = ', avg
      write(iowrit,*) 'Std Dev = ', stdev
      write(iowrit,*) 'Number in Modeling Domain = ', nrows*ncols
      write(iowrit,*) 'Average = ', avg_t
      write(iowrit,*) 'Std Dev = ', stdev_t
      close(iowrit)

c Writing out normalized Desert Rock MAI to maidtbl.dat format.
      scale = tpa_avg / avg
      print*, 'scale = ', scale
      open(unit = iowrit, file = 'modern.m', form = 'formatted')
      do i = 1, 3
        write(iowrit,'(a60)') headers(i)
      enddo
      write(iowrit,'(a44,a11,a24)') headers(4),' Scaled on ',fdate()
      write(iowrit,'(a9,i4)') junk(1), ncols
      write(iowrit,'(a9,i4)') junk(2), nrows
      write(iowrit,'(a9,f15.5)') junk(3), xllcorner
      write(iowrit,'(a9,f15.5)') junk(4), yllcorner
      write(iowrit,'(a9,f14.5)') junk(5), cellsize
      do i = 5, 7
        write(iowrit,'(a60)') headers(i)
      enddo
      do i = 1, nrows*ncols
         temp = array(i,3) * scale
         write(iowrit,'(e15.8)') temp
      enddo
      write(iowrit,'(a9)') junk(6)
      close(iowrit)

c Note that the order of the read and writes retains the date stamp of MAI.
c Read in Hobbs DEM of net infiltration results.
      open(unit=ioread,file='maidtbl.hobbs1000',form='formatted')
      do i = 1, 4
        read(ioread,'(a60)') headers(i)
      enddo
```

```
      read(ioread,'(a9,i10)') junk(1), ncols
      read(ioread,'(a9,i10)') junk(2), nrows
      read(ioread,'(a9,f16.5)') junk(3), xllcorner
      read(ioread,'(a9,f16.5)') junk(4), yllcorner
      read(ioread,'(a9,f15.5)') junk(5), cellsize
      do i = 5, 7
        read(ioread,'(a60)') headers(i)
      enddo
      do i = 1, nrows*ncols
         read(ioread,'(e15.8)') hobbs(i)
      enddo
      read(ioread,'(a9)') junk(6)
      close(ioread)

c Writing out normalized Hobbs MAI to maidtbl.dat format.
      open(unit = iowrit, file = 'monsoon.ub', form = 'formatted')
      do i = 1, 3
        write(iowrit,'(a60)') headers(i)
      enddo
      write(iowrit,'(a44,a11,a24)') headers(4),' Scaled on ',fdate()
      write(iowrit,'(a9,i4)') junk(1), ncols
      write(iowrit,'(a9,i4)') junk(2), nrows
      write(iowrit,'(a9,f15.5)') junk(3), xllcorner
      write(iowrit,'(a9,f15.5)') junk(4), yllcorner
      write(iowrit,'(a9,f14.5)') junk(5), cellsize
      do i = 5, 7
        write(iowrit,'(a60)') headers(i)
      enddo
      do i = 1, nrows*ncols
         temp = hobbs(i) * scale
         write(iowrit,'(e15.8)') temp
      enddo
      write(iowrit,'(a9)') junk(6)
      close(iowrit)

c Read in Delta DEM of net infiltration results.
      open(unit=ioread,file='maidtbl.delta1000',form='formatted')
      do i = 1, 4
        read(ioread,'(a60)') headers(i)
      enddo
      read(ioread,'(a9,i10)') junk(1), ncols
      read(ioread,'(a9,i10)') junk(2), nrows
      read(ioread,'(a9,f16.5)') junk(3), xllcorner
      read(ioread,'(a9,f16.5)') junk(4), yllcorner
      read(ioread,'(a9,f15.5)') junk(5), cellsize
      do i = 5, 7
        read(ioread,'(a60)') headers(i)
      enddo
      do i = 1, nrows*ncols
         read(ioread,'(e15.8)') delta(i)
      enddo
      read(ioread,'(a9)') junk(6)
      close(ioread)

c Writing out normalized Delta MAI to maidtbl.dat format.
      open(unit = iowrit, file = 'glacial.lb', form = 'formatted')
      do i = 1, 3
        write(iowrit,'(a60)') headers(i)
      enddo
      write(iowrit,'(a44,a11,a24)') headers(4),' Scaled on ',fdate()
      write(iowrit,'(a9,i4)') junk(1), ncols
      write(iowrit,'(a9,i4)') junk(2), nrows
      write(iowrit,'(a9,f15.5)') junk(3), xllcorner
      write(iowrit,'(a9,f15.5)') junk(4), yllcorner
      write(iowrit,'(a9,f14.5)') junk(5), cellsize
      do i = 5, 7
        write(iowrit,'(a60)') headers(i)
      enddo
      do i = 1, nrows*ncols
         temp = delta(i) * scale
         write(iowrit,'(e15.8)') temp
      enddo
      write(iowrit,'(a9)') junk(6)
      close(iowrit)

c Read in Rosalia DEM of net infiltration results.
      open(unit=ioread,file='maidtbl.rosalia1000',form='formatted')
      do i = 1, 4
        read(ioread,'(a60)') headers(i)
      enddo
      read(ioread,'(a9,i10)') junk(1), ncols
      read(ioread,'(a9,i10)') junk(2), nrows
      read(ioread,'(a9,f16.5)') junk(3), xllcorner
      read(ioread,'(a9,f16.5)') junk(4), yllcorner
      read(ioread,'(a9,f15.5)') junk(5), cellsize
      do i = 5, 7
        read(ioread,'(a60)') headers(i)
      enddo
      do i = 1, nrows*ncols
         read(ioread,'(e15.8)') rosalia(i)
      enddo
      read(ioread,'(a9)') junk(6)
      close(ioread)

c Writing out normalized Rosalia MAI to maidtbl.dat format.
      open(unit = iowrit, file = 'glacial.ub', form = 'formatted')
      do i = 1, 3
        write(iowrit,'(a60)') headers(i)
      enddo
      write(iowrit,'(a44,a11,a24)') headers(4),' Scaled on ',fdate()
      write(iowrit,'(a9,i4)') junk(1), ncols
      write(iowrit,'(a9,i4)') junk(2), nrows
      write(iowrit,'(a9,f15.5)') junk(3), xllcorner
      write(iowrit,'(a9,f15.5)') junk(4), yllcorner
      write(iowrit,'(a9,f14.5)') junk(5), cellsize
      do i = 5, 7
        write(iowrit,'(a60)') headers(i)
      enddo
      do i = 1, nrows*ncols
         temp = rosalia(i) * scale
         write(iowrit,'(e15.8)') temp
      enddo
      write(iowrit,'(a9)') junk(6)
      close(iowrit)

      stop
      end
```

extract.f script:

```
      program extract
c Script for determining if a point lies within the repository footprint,
c   or any other odd-shaped outline (rectangles still work).
c  Assumptions:
c   1. assumes counter-clockwise order of polygon coordinates
c   2. assumes first coordinate is repeated as last coordinate
c RFedors  June 14, 2000
c Modified July 23, 2001 for TEF,
c To read in points and to extract net infilt from area around those pts.
c Run on a SUN (UNIX) because of problems with the LAHEY compiler on WinNT.
c  vulcan:~/Itym-Usage/TEF-July2001/extract.f
c23456789 123456789 123456789 123456789 123456789 123456789 123456789 12
      integer ioread, iowrit, mx, i, j, k, m, ipts, npts
      integer ndrift, i_max, i_min, ict, lf_rt
      parameter (mx=1000,mxx=100000)
      real*8 ymax, ytop, xbot, xpos, ypos, sum, sum2
      real*8 avg, stdev, avg_t, stdev_t, xsegment, ax,ay
      real*8 drift(5,2), segment(4,2), tef(mx,2)
      real*8 array(mxx,3), repository(mxx), reposit(mxx,3)
      character*9 flagside(mx), junk
      character*60 header
      real*8 xllcorner, yllcorner, cellsize, cellhalf

c set input and output unit numbers
      ioread = 7
      iowrit = 8

c read in DEM of infiltration; note that the coordinates of the
c southwest corner of the domain are given in the header, but the
c ordering of data is row-major starting from the northwest corner.

      open(unit = ioread, file = 'maidtbl.dat', form = 'formatted')

      do i = 1, 4
        read(ioread,'(a60)') header
      enddo
      read(ioread,'(a9,i10)') junk, ncols
      read(ioread,'(a9,i10)') junk, nrows
      read(ioread,'(a9,f16.5)') junk, xllcorner
      read(ioread,'(a9,f16.5)') junk, yllcorner
      read(ioread,'(a9,f15.5)') junk, cellsize
      do i = 1, 3
        read(ioread,'(a60)') header
      enddo
      print*, ncols, nrows, cellsize, xllcorner, yllcorner

      ypos = yllcorner + cellsize * dfloat(nrows-1)
      xpos = xllcorner
      k = 1
      do i = 1, nrows
        do j = 1, ncols
          read(ioread,'(e15.8)') array(k,3)
          array(k,1) = xpos
          array(k,2) = ypos
          xpos = xpos + cellsize
          k = k + 1
        enddo
        ypos = ypos - cellsize
        xpos = xllcorner
      enddo
      close(ioread)

c read in point coords file, 1st line comment line, 2nd line # of points
c account for repeated entry of first point as last entry.
      open(unit = ioread, file = 'tefd.txt', form = 'formatted')
      read(ioread,'(a60)') header
      read(ioread,'(i5)') npts
      do i = 1, npts
        read(ioread,'(2d12.7)') tef(i,1), tef(i,2)
      enddo
      close(ioread)

c Create polygon around each point, go through remainder of program, then
c  do the next point. For now, the shape is a simple 30m pixel surrounding
c  the desired location.  x=easting and y=northing.
      cellhalf = cellsize * 0.5d0
      ndrift = 5
      do ipts=1,npts
        drift(1,1) = tef(ipts,1) - cellhalf - 0.d0
        drift(1,2) = tef(ipts,2) + cellhalf
        drift(2,1) = tef(ipts,1) - cellhalf
        drift(2,2) = tef(ipts,2) - cellhalf - 0.d0
        drift(3,1) = tef(ipts,1) + cellhalf + 0.d0
        drift(3,2) = tef(ipts,2) - cellhalf
        drift(4,1) = tef(ipts,1) + cellhalf
        drift(4,2) = tef(ipts,2) + cellhalf + 0.d0
        drift(5,1) = drift(1,1)
        drift(5,2) = drift(1,2)

c  set up usage of drift coordinates; checking to right or left of segment;
c  find min and max y-coord, then assign left/right to line segments
        ymax = 0.d0
        ymin = 4.d10
        do i = 1, ndrift-1
          if(drift(i,2).ge.ymax) then
            ymax = drift(i,2)
            i_max = i
          endif
          if(drift(i,2).le.ymin) then
            ymin = drift(i,2)
            i_min = i
          endif
        enddo

        if(i_max.lt.i_min) then
            do i = 1, ndrift-1
              flagside(i) = 'left'
            enddo
            do i = i_max, i_min-1
              flagside(i) = 'right'
            enddo
          else
            do i = 1, ndrift-1
              flagside(i) = 'right'
            enddo
            do i = i_min, i_max-1
              flagside(i) = 'left'
            enddo
        endif

c calculate line segment equations going counter-clockwise;
c segment(i,1)=slope; segment(i,2)=intercept; for horizontal lines,
```

```
c set flagside to avoid checking either side of the segment and
c then set denominator to any number just to avoid blowout;
c for vertical lines (xbot=0), set numerator of slope to a small number.

      do i = 1, ndrift-1
        ytop = drift(i+1,2) - drift(i,2)
        xbot = drift(i+1,1) - drift(i,1)
        if(dabs(ytop).lt.1.d-9) then
            flagside(i)='neither'
            segment(i,1) = 1.d0
            segment(i,2) = 1.d0
        elseif(dabs(xbot).lt.1.d-10) then
            segment(i,1) = 1.d0
            segment(i,2) = 0.d0
        else
            segment(i,1) = ytop / xbot
            segment(i,2) = drift(i,2) - (segment(i,1)*drift(i,1))
        endif
      enddo
      do i = 1, ndrift-1
        print*, segment(i,1), segment(i,2)
      enddo

c check to see if current position is within repository outline
      lf_rt = 0
      ict = 0
      do i = 1, nrows*ncols
        ay = array(i,2)
        ax = array(i,1)
        do m = 1, ndrift-1
          if(ay.le.drift(m,2).and.ay.gt.drift(m+1,2).or.
     &         ay.ge.drift(m,2).and.ay.lt.drift(m+1,2)) then
            xsegment = (array(i,2)-segment(m,2)) / segment(m,1)
            if(dabs(segment(m,2)).le.1.d-10) xsegment = drift(m,1)
            if(flagside(m).eq.'right'.and.ax.ge.xsegment) lf_rt= lf_rt+ 1
            if(flagside(m).eq.'left'.and.ax.le.xsegment) lf_rt= lf_rt + 1
          endif
          if(lf_rt.eq.2) then
              ict = ict + 1
              repository(ict) = array(i,3)
              reposit(ict,1) = array(i,1)
              reposit(ict,2) = array(i,2)
              reposit(ict,3) = array(i,3)
              lf_rt = 0
          endif
        enddo
        lf_rt = 0
      enddo

c statistics on repository cells
      open(unit=iowrit, file='tef.out', form='formatted')
      sum = 0.d0
      do i = 1, ict
         sum = sum + repository(i)
      enddo
      avg = sum / dfloat(ict)

      sum = 0.d0
      do i = 1, ict
         sum = sum + dabs(repository(i) - avg)
      enddo
      stdev = dsqrt(sum/dfloat(ict-1))

      write(iowrit,*) 'Number in Polygon = ', ict
      write(iowrit,*) 'Average = ', avg
      write(iowrit,*) 'Std Dev = ', stdev

c Go back and do the next input location
      enddo

      sum = 0.d0
      do i = 1, ncols*nrows
         sum = sum + array(i,3)
      enddo
      avg_t = sum / dfloat(nrows*ncols)

      sum = 0.d0
      do i = 1, ncols*nrows
         sum = sum + dabs(array(i,3) - avg_t)
      enddo
      stdev_t = dsqrt(sum/dfloat(nrows*ncols-1))

      write(iowrit,*) 'Number in Modeling Domain = ', nrows*ncols
      write(iowrit,*) 'Average = ', avg_t
      write(iowrit,*) 'Std Dev = ', stdev_t

      close(iowrit)

      stop
      end
```

Output Files

Output file from extract.f is named tef.out. The output file from extract-repository.f is named summary-main.dat. These two files were cat'd together in UNIX and named for the appropriate climate case. These renamed files are included below.

mean-modern.out

```
Number in Polygon =    1
 Average =      5.9894397000000
 Std Dev =   NaN
 Number in Polygon =    1
 Average =      14.689873000000
 Std Dev =   NaN
 Number in Polygon =    1
 Average =      12.497174000000
 Std Dev =   NaN
 Number in Polygon =    1
 Average =      14.157749000000
 Std Dev =   NaN
 Number in Modeling Domain =    59700
 Average =      6.9561471139145
 Std Dev =      1.5758140752787
 Number in Repository =    5481
 Average =      8.5000000026638
 Std Dev =      1.8781190860498
 Number in Modeling Domain =    59700
 Average =      6.9561471139145
 Std Dev =      1.5758140752787
```

upper-bound monsoon

```
Number in Polygon =    1
 Average =      30.345164000000
 Std Dev =   NaN
 Number in Polygon =    1
 Average =      65.048693000000
 Std Dev =   NaN
 Number in Polygon =    1
 Average =      60.396325000000
 Std Dev =   NaN
 Number in Polygon =    1
 Average =      64.455194000000
 Std Dev =   NaN
 Number in Modeling Domain =    59700
 Average =      34.115790387203
 Std Dev =      3.1442186994920
 Number in Repository =    5481
 Average =      41.564001399927
 Std Dev =      3.8811504562996
 Number in Modeling Domain =    59700
 Average =      34.115790387203
 Std Dev =      3.1442186994920
```

```
      lower-glacial.out
Number in Polygon =    1
 Average =      12.170442000000
 Std Dev =  NaN
 Number in Polygon =    1
 Average =      26.415352000000
 Std Dev =  NaN
 Number in Polygon =    1
 Average =      23.620349000000
 Std Dev =  NaN
 Number in Polygon =    1
 Average =      25.860109000000
 Std Dev =  NaN
 Number in Modeling Domain =    59700
 Average =      13.704020440467
 Std Dev =      2.0115316273985
 Number in Repository =    5481
 Average =      16.536907926473
 Std Dev =      2.4397166581439
 Number in Modeling Domain =    59700
 Average =      13.704020440467
 Std Dev =      2.0115316273985
```

```
      upper-glacial.out
Number in Polygon =    1
Average =      52.206100000000
Std Dev =  NaN
Number in Polygon =    1
Average =      92.073588000000
Std Dev =  NaN
Number in Polygon =    1
Average =      89.514108000000
Std Dev =  NaN
Number in Polygon =    1
Average =      92.874099000000
Std Dev =  NaN
Number in Modeling Domain =    59700
Average =      55.234934394221
Std Dev =      3.4462704904686
Number in Repository =    5481
Average =      65.666701494618
Std Dev =      4.2844428553188
Number in Modeling Domain =    59700
Average =      55.234934394221
Std Dev =      3.4462704904686
```

The results for each TEF location were extracted into a table and the remaining entries were calculated as follows in the bubo: J:\ITYM-Usage\TEF-July2001\maiTEFlocations.xls file:

- lower-bound modern: scale results from scaled mean modern by 1/8.5
- upper-bound modern: scale results from scaled mean modern by 13/8.5
- lower-bound monsoon: use scaled mean modern results directly
- mean monsoon: calculate average of lower- and upper-bound scaled monsoon
- mean glacial transition: calculate average of lower- and upper-bound scaled glacial transition

The scaling factor was 8.50/12.701897= 0.668229 where 8.5 mm/yr was the desired repository average net infiltration and 12.7 mm/yr was obtained from the Desert Rock IYTM output. This scaling factor was used on the Hobbs, Delta, and Rosalia infiltration maps.

The table below contains the results that Chandrika and Hughson wanted for their top boundary condition.

	l4c3	l4c1	l1c3	l7c3
	Point 1	Point 2	Point 3	Point 4
	MAI mm/yr	MAI mm/yr	MAI mm/yr	MAI mm/yr
modern lower-bound	2.8	6.9	5.9	6.7
modern mean	6.0	14.7	12.5	14.2
modern upper-bound	9.2	22.5	19.1	21.7
monsoon lower-bound	6.0	14.7	12.5	14.2
monsoon mean	18.2	39.9	36.4	39.3
monsoon upper-bound	30.3	65.0	60.4	64.5
glacial lower-bound	12.2	26.4	23.6	25.9
glacial mean	32.2	59.2	56.6	59.4
glacial upper-bound	52.2	92.1	89.5	92.9

Note that the l4c3 location has lower values of net infiltration than the values used by LLNL (DOE) for this location. The DOE uses 8.? mm/yr, 21.? mm/yr, and 38.7 mm/yr for the modern, monsoon, and glacial transition mean cases for location l4c3. Shown in the next section, the NRC/CNWRA repository averages are generally twice as large as the DOE values.

Comparison with DOE Averages

Comparison of results with DOE averages from CRWMS M&O (2000, infiltration AMR) is something that we've needed to do for awhile. We haven't put much credence in a comparison because of the NRC/CNWRA lack of a vegetation module, runon module, and snowmelt module. The first two modules should be in place this year; the last will probably not get incorporated. At any rate, the following table is a first-cut comparison.

Mean Annual Infiltration*	DOE/USGS repository average, mm/yr	NRC/CNWRA repository average, mm/yr	NRC/CNWRA repository average, mm/yr (not scaled)
modern lower-bound	1.3	4	-
modern mean	4.6	8.5	12.7
modern upper-bound	11.1	13	-
monsoon lower-bound	4.6	8.5	12.7
monsoon mean	12.2	(25.1)	(37.5)
monsoon upper-bound	19.8	41.6	62.2
glacial lower-bound	2.5	16.6	24.7
glacial mean	17.8	(41.2)	(61.5)
glacial upper-bound	33	65.7	98.2

*Entries in parentheses are calculated as the average from the upper and lower bounds.

References:

CRWMS M&O, 2000, "Simulation of Net Infiltration for Modern and Potential Future Climates." ANL–NBS–GS–000032. Revision 00. Las Vegas, Nevada: Civilian Radioactive Waste Management System Management and Operating Contractor).

CNWRA, 2000. Total-System Performance Assessment (TPA) Version 4.0 Code: Module Descriptions and User's Guide, Predecisional CNWRA Report to NRC, San Antonio, TX: Center for Nuclear Waste Regulatory Analyses, April 2000.

Winterle, J.R., R.W. Fedors, D.L. Hughson, and S.A. Stothoff, 1999, Update of Hydrologic Parameters for Total-System Performance Assessment Code, CNWRA Letter Report, San Antonio, TX: Center for Nuclear Waste Regulatory Analyses, August 1999.