DEVELOPMENT OF A CONCEPTUAL MODEL OF THE SOIL-MOISTURE-PLANT SUB-SYSTEM OF THE HYDROLOGICAL CYCLE

VOL 2

(APPENDICES)

DARRYN ANDREW EVANS

Doctor of Philosophy

THE UNIVERSITY OF ASTON IN BIRMINGHAM

October 1990

CONTENTS

Volume 2

APPENDIX 1

The soil particle-size classes described below are those defined by the United States Department of Agriculture's Soil Conservation Service (USDA-SCS).

Coarse Fragments	(CSE FRAG)	>2	mm
Sand	(SAND)	2-0.05	mm
Silt	(SILT)	0.05-0.002	mm
Clay	(CLAY)	<0.002	mm

Sand and Silt are further divided:

Sand:-			
Very Coarse	(V. CO.)	2-1	mm
Coarse	(CO.)	1-0.5	mm
Medium	(MED.)	0.5-0.25	mm
Fine	(FINE)	0.25-0.1	mm
Very Fine	(V. FINE)	0.1-0.05	mm
Silt:			
Coarse	(CO.)	0.05-0.02	mm
Fine	(FINE)	0.02-0.002	mm

* Letters in brackets refer to the abbreviations given in tables 3.1-3.5 and 7.1.

Other abbreviations used in tables 3.1-3.5 and 7.1.

Organic Carbon (ORG CARB)
Carbon/Nitrogen Ratio (CARB/NIT RATIO)
Magnesium (Mg)
Sodium (Na)
Potassium (K)
Cation Exchange Capacity (CEC)
Bulk Density (BULK DENS)
Total Porosity (TOT PORO)

APPENDIX 2

COMPUTER PROGRAM LISTINGS

```
C
C ***************************************************************************
***
C ***************************************************************************
***
C **
 **
C **
 **
C **                          MAIN BODY OF PROGRAM
 **
C **
 **
C **
 **
C ***************************************************************************
***
C ***************************************************************************
***
C
      PROGRAM Soil_Atmosphere_Simulation
C
C The purpose of this program is to simulate the effects of water movement
C within the soil and its effects on crops. The program utilises the
C relationships between the soil, the atmosphere and the plant calculating
C the movement of water between these using mathematical relationships as fou
nd
C in the literature. It does this without the aid of complex numerical method
s
C instead it uses the timestep to control the rate of flow between the variou
s
C components of the model.
C
C Variable Names and Meanings:
C
C INCLUDE FILE: Main.Inc
C COMMON
C moist(n) =moisture content of the nth layer
C nolyrs =number of soil layers
C wetfr =the layer containing the infiltration wetting front
C head(n) =suction value for the nth layer
C conduc(n) =conductivity value for the nth layer
C pond =amount of water lying in a 'free' state on the surface
C d(n) =depth of nth layer
C time =multiplier used to describe length of timestep
C COMMON / param /
C kf(n) =conductivity value of wetting zone in nth layer
C sf(n) =suction across wetting front in nth layer
C method =type of infiltration process taking place
C         1. all water absorbed by profile
C         2. infiltration governed by movement of wetting fronts
C         3. wetting fronts have been diffused
C         4. a coarse boundary has been reached infiltration is governed by
C            the overlying layer
C kfin(n) =dominant conductivity of wetting zone in the nth layer
C COMMON / macro /
C mp(j) =percentage of ground occupied by channels of the jth class
C mrf(j,i) =radii of wetting fronts for jth macropore class in the ith layer
C mr(j) =radius of jth macropore class channels
C mn(j) =number of channels in jth macropore
C nmw(j) =layer in which the top of the water column resides for the jth clas
s
C          of macropores
C mamd(j) =height of water column above the base in jth macropore class
C hd(j) =depth of macropores of jth class
C msd(j) =layer into which macropores of the jth class extend
C f(n) =sum of infiltration into nth layer
C COMMON / perc /
```

```
C nl(n),alpha(n) =parameters used in Genuchten(1980) equation for calculating
C                 suction
C tr(n) =residual moisture content for the nth layer
C ks(n) =saturated conductivity for the nth layer
C ts(n) =saturated moisture content of the nth layer
C COMMON / Hussein /
C d4/coefd =groundwater parameters
C p =bare soil evaporation parameter
C dwt =depth to water table from the top of the layer
C maxran =the rainfall value at which if higher the rainfall is distributed
C         in the day by means of a parabolic relationship, or if lower purely
C         randomly
C COMMON / Redistrib /
C mc(n) =moisture content of the nth layer as in moist, but when a wetting
C        front enters a layer by infiltration, it contains the average moistu
re
C        content below this.
C mci(n) =moisture content of layer outside of the wetting fronts
C ro =water not infiltrated into soil directly to be added to macropores
C ln =length of wetting front in current layer
C irand =initial random number value for generating numbers for rainfall
C        distribution
C COMMON / Evaporation /
C eplant =evaporative demand for transpiration
C esoil =evaporative demand for bare soil evaporation
C evapor =total daily evaporative demand
C COMMOM / Clay /
C sv =amount of shrinkage between 50cm and 15000cm in terms of volume
C tu =moisture content at 50cm
C tt =moisture content at 15000cm
C shrgra =gradient of the shrinkage equation
C shrc =intercept of the shrinkage equation
C COMMON / Growth1 /
C plntyp =indicates plant group to be modelled
C         1. permanent pasture
C         2. annual monocotyledon crops
C maxlai =maximum potential leaf area index
C lc,rc =converts growth blocks into increases in lai and root density
C        respectively
C koef,lam =parameter values used in the equation partitioning growth
C           between the roots and the leaves
C rdens(j) =root density in the jth layer
C COMMON / Growth2 /
C a4,b4 =parameter values used to determine the amount of growth
C c4    =parameter value relating actual to potential leaf area index
C lai =leaf area index defined here as the fractional canopy cover
C rotdep =deepest layer into which roots extend
C minrat,maxrat =parameter values used in the equation partitioning the
C                growth blocks between the roots and the leaves
C COMMON / Growth3 /
C wtruse =amount of water transpired by the crop
C nocrop =number of crops in the simulation period
C ndays =number of days elapsed in the simulation
C COMMON / Growth4 /
C emerg =number of days after the simulation start for emergence of crop
C anth =number of days after the simulation start before crop anthesis
C harv =number of days after the simulation start before crop harvest
C potlai =potential leaf area index
C gu =growth units
C extrat =extension rate of roots per day
C
C INCLUDE FILE: Temp.Inc
C COMMON / infil /
C fp =infiltration capacity
C fs =if rainfall intensity is greater than surface layer conductivity
C     it is the amount of water infiltrated before ponding occurs
C fprem =amount of water left after ponding for infiltration into the
C        profile
```

```
C tmlft =If the wetting front reaches a boundary during a timestep, and
C        thus conditions change. This parameter indicates the length
C        of time remaining for calculation of infiltration under the new
C        conditions
C rain =amount of precipitation
C fpleft =amount of infiltrated water left from fine_over_coarse
C         subroutine if coarse layer becomes saturated
C Common Blocks / templ /, / temp2 / & / temp3 / all contain temporary
C values for equivalent values held in the file Main.Inc
C COMMON / templ /
C wet, wetmc, wetmci =moist, mc, mci respectively
C wfr,len,runoff =wetfr,ln,ro repectively
C COMMON / temp2 /
C mwf,md =mrf,mamd respectively
C COMMON / temp3 /
C meth,pond =method,ponded respectively
C
C Local Variables
C total =total amount of rainfall for the day in question
C idays =length of simulation in days
C decis =decision variable allowing the soil and plant parameters to be
C        altered from one simulation to the next
C mdep =total depth of water in the macropores (irrespective of radii)
C fcap =indicates if any of the soil layers have moisture contents greater
C       than field capacity
C actran =actual transpiration
C stoar =calculated daily store of water within the subsystem
C crops =number of crops grown in a simulation
C
C Subroutines Called:
C Sys_Attrib  =inputs the characteristics of the soil such as bulk density
C              saturated moisture content and particle-size analysis, from
C              this the parameters of the equations are calculated. Allows
C              also parameters for macropore and plant growth routines to
C              be input
C Soil_Cracks =calculates the amount of soil cracking based upon clay and
C              water content
C One_Day, One_Hour, =determine the timestep to be used for the following
C   Infil_Time        processes, which they later call, infiltration, macropo
re
C                     percolation and groundwater flow. The latter also
C                     distributes rainfall into hourly units
C Evap_Cal =determines the evaporative demands for intercepted water, soil an
d
C           plant evaporation
C Bare_Soil_Evap =determines bare soil evaporation
C Root_Abstraction =determines the transpiration losses from each layer in th
e
C                   soil
C Plant_Growth =determines the daily growth of the plant in terms of leaf
C               area index and root density distribution
C Capillary_Rise =calculates the amount of flow upwards from the layer below
C
C Author: Darryn Evans
C Created: 24th November 1988
C Updated: 20th March 1989
C
      REAL total,mdep,mcfc(50),sumin,sumout,minkf
      INTEGER idays,count,k,k2,nin,crops,minlyr
      CHARACTER decis,decis1,decis2,decis3,decis4
      LOGICAL fcap
C
      INCLUDE 'main.inc'
C
      nin =84
      nout =68
      irand =0.6
      rootp =5000.
```

```
      PRINT*,'Do you want to change the programs attributes y or n',
     1                        '[Press Return]'
      READ(*,100)decis
      IF (decis .eq. 'y') THEN
        CALL Sys_Attrib
      END IF
      PRINT*,'How many days do you want the program to run ',
     1                        '[Press Return]'
      READ(22,299)idays
      WRITE(20,299)idays
C
C Input the data generated by Sys_Attrib into the programs COMMON blocks
C
C Input soil data and parameter values
C
      OPEN(52,file='Vale_Params',status='old',form='formatted',
     1                          access='direct',recl=80)
      READ(52,299,rec=1)nolyrs
      WRITE(20,299)nolyrs
      DO 1 j =1,nolyrs
        k=(((j-1)*3)+1)
        READ(52,200,rec=k+1)nl(j),alpha(j),tr(j),ts(j)
        READ(52,201,rec=k+2)sf(j),kf(j)
        READ(52,202,rec=k+3)d(j),mc(j),mci(j),moist(j)
        ks(j) =kf(j)*2.
    1 CONTINUE
      CLOSE (52)
C
C Input the macropore properties
C
      OPEN(53,status='unknown')
      READ(53,203)permcl
      DO j =1,permcl
        READ(53,204)mn(j),mr(j),msd(j)
        READ(53,205)mp(j),hd(j)
      END DO
      READ(53,206)shgra,shrc,sv,tu,tt
C
C initialise non-zero star parameters
C
      wetfr =1
      method =1
C
C Input the empirical values from Hussein needed for the calculation of
C groundwater flow, initialise groundwater depth and calculate the moisture
C content of the basal layer
C
      dwt =100.
      moist(nolyrs) =((dwt*0.14)+((d(nolyrs)-dwt)*
     1                                ts(nolyrs)))/d(nolyrs)
      OPEN(18,file='xhudata',status='old',access='direct',form=
     1                          'formatted',recl=16)
      READ(18,209,rec=1)d4
      READ(18,209,rec=2)coefd
      READ(18,209,rec=3)p
      READ(18,209,rec=4)maxran
      CLOSE (18)
C
C Input the Plant Characteristics
C
      READ(nin,500)plntyp
      IF (plntyp .eq. 1) THEN
        READ(nin,501)a4,b4,c4,lc,rc
        READ(nin,502)maxlai,minrat,maxrat,koef,lam
        READ(nin,503)rotdep,lai,havest
        potlai =lai
        DO j =1,rotdep
          READ(nin,504)rdens(j)
```

```
        END DO
      ELSE IF (plntyp .eq. 2) THEN
        READ(nin,505)crops
        READ(nin,506)nocrop
        READ(nin,507)emerg,anth,harv
      END IF
C
C For calibration purposes the following statements allow certain program
C variables to be altered; these being saturated conductivity, saturated
C moisture content of the surface layer, initial moisture contents,
C initial root density and the parameter Rc
C
      PRINT*,'Do you want to change the calibration values y or n'
      READ(*,100)decis1
      IF (decis1 .eq. 'y') THEN
        PRINT*,'Do you want to change ks values y or n'
        READ(*,100)decis3
        IF (decis3 .eq. 'y') THEN
          DO j =1,nolyrs-1
            PRINT*,'Input ks of layer',j,' which is now',ks(j)
            READ*,ks(j)
            kf(j) =ks(j)/2.
          END DO
          ks(nolyrs) =ks(nolyrs-1)
          kf(nolyrs) =ks(nolyrs)/2.
        END IF
        PRINT*,'Do you want to change surface ts value y or n'
        READ(*,100)decis4
        IF (decis4 .eq. 'y') THEN
          PRINT*,'Input ts which is now',ts(1)
          READ*,ts(1)
        END IF
        PRINT*,'Do you want to change initial moisture',
     1                      ' contents y or n'
        READ(*,100)decis4
        IF (decis4 .eq. 'y') THEN
          DO j =1,nolyrs-1
            PRINT*,'Input moist of layer ',j,' old value'
     1                                              ,moist(j)
            READ*,moist(j)
            mc(j) =moist(j)
            mci(j) =moist(j)
          END DO
        END IF
        IF (plntyp .eq. 1) THEN
          PRINT*,'Do you want to change root densities y or n'
          READ(*,100)decis4
          IF (decis4 .eq. 'y') THEN
            DO j =1,rotdep
              PRINT*,'Old value is',rdens(j),' Input new'
              READ*,rdens(j)
            END DO
          END IF
          PRINT*,'Input value of Rc: Old value is ',rc
          READ*,rc
          PRINT*,'Input Value of Root Resistance'
          READ*,rootp
        END IF
      END IF
C
C Check to see if there are any restricting layers for the infiltration
C routine. If there are equate the conductivity of the lower layers to
C the values calculated for this layer (kfin).
C
      DO j =1,(nolyrs-1)
        IF (j .eq. 1) THEN
          minlyr =1
          minkf =kf(j)
```

```
          kfin(j) =kf(j)
        ELSE
          IF (kf(j) .gt. minkf) THEN
            kfin(j) =kf(minlyr)
          ELSE
            minlyr =j
            minkf =kf(minlyr)
            kfin(j) =kf(minlyr)
          END IF
        END IF
      END DO
C
C Open a file in which to store the daily summary.
C
      OPEN(60,file='Daily_Summary',status='new',form='formatted',
     1                                                    recl=80)
      extrat =2.
      DO 4 j =1,idays
C
C If crops are grown check to see if they have just started to emerge, if so
C read in relevant data for them
C
        ndays =j
        IF (plntyp .eq. 2) THEN
          IF (emerg .eq. j) THEN
            READ(nin,501)a4,b4,c4,lc,rc
            READ(nin,502)maxlai,minrat,maxrat,koef,lam
            READ(nin,509)rotdep,lai
            potlai =lai
            DO  i =1,rotdep
              READ(nin,504)rdens(j)
            END DO
            IF ((j .eq. (harv+1)) .and. (nocrop .lt. crops)) THEN
C
C read in the emergence date of the next plantings
C
              READ(nin,506)nocrop
              READ(nin,507)emerg,anth,harv
            END IF
          END IF
        END IF
C
C equate all the moisture variables as equal and initialise
C
        DO 5 i =1,nolyrs
          mc(i) =moist(i)
          mci(i) =moist(i)
    5   CONTINUE
        actran =0.
C
C Enter Daily Atmospheric Data Information from an external file and
C convert rainfall from mm's to cm's
C
        READ(22,301)total,evapor
        total =total/10.
C
C Calculate the stage of cracking in the surface layer if appropriate
C
        IF (sv .gt. 0.) THEN
          CALL Soil_Cracks
        ELSE
          noclas =permcl
        END IF
C
C determine the appropriate timestep to start of with
C
        mdep =0.
        DO ij =1,noclas
```

```
          mdep =mdep+mamd(ij)
        END DO
        count =nolyrs-1
        fcap =.true.
        sumdep =dwt
        DO WHILE ((fcap .eq. .true.) .and. (count .ge. 1))
          sumdep =sumdep+d(count)
          depth =sumdep-(d(count)/2)
          mcfc(count) =tr(count)+((ts(count)-tr(count))/((1+((alpha
     1                (count)*depth)**nl(count)))**(1-(1/nl(count)))))
          IF (moist(count) .gt. mcfc(count)) THEN
            fcap =.false.
          ELSE
            count =count-1
          END IF
        END DO
C
C Choose appropriate subroutine to start of with
C
        IF (total .le. 0.000001) THEN
          IF ((fcap .eq. .true.) .and. (mdep .le. 0.)) THEN
            CALL One_Day
          ELSE
            CALL One_Hour
          END IF
        ELSE
          CALL Infil_Time(total)
        END IF
C
C Redistribution algorithms have been applied as well as percolation and
C groundwater movement. Now the evapotranspiration, plant growth and
C capillary-rise algorithms are implemented working on a timestep of one
C day. All wetting fronts are assumed to have diffused at the end of the
C day if they still exist, even if water is still held within the
C macropores. Recalculate suction and conductivity.
C
        method =1
        wetfr =1
        ln =0.
        DO 6 i =1,nolyrs
          DO ij =1,noclas
            mrf(ij,i) =0.
          END DO
          mc(i) =moist(i)
          mci(i) =moist(i)
          CALL Suction_Cal(moist(i),ts(i),i)
          CALL Conductivity_Cal(moist(i),ts(i),i)
    6   CONTINUE
C
        CALL Evap_Cal
        IF (esoil .gt. 0.) THEN
          CALL Bare_Soil_Evap(total)
        END IF
        CALL Root_Abstraction
        IF (actran .gt. 0.) THEN
          CALL Plant_Growth
        END IF
C
        CALL Calculate_Rise
        DO 7 i =1,nolyrs
          CALL Suction_Cal(moist(i),ts(i),i)
          CALL Conductivity_Cal(moist(i),ts(i),i)
    7   CONTINUE
C
C If there is still water on or above the ground surface in 'free water' stor
es
C it is removed from the system at the end of the day by runoff
C
```

```
        IF (noclas .gt. permcl) THEN
          pond1 =(mamd(noclas)*(mp(noclas)/100.))
          mamd(noclas) =0.
        END IF
        pond =pond+ro+pond1
        IF ((pond .gt. 0.) .or. (int .gt. 0.)) THEN
          pond =0.
          ro =0.
        END IF
C
C Write out daily summary to external file(60) and graphics file(20), hence
C recalculate suction and conductivity
C
        WRITE(60,211)j
        DO 9 i =1,nolyrs
          CALL Suction_Cal(moist(i),ts(i),i)
          CALL Conductivity_Cal(moist(i),ts(i),i)
          WRITE(60,210)i,moist(i),conduc(i),head(i)
          WRITE(20,300)moist(i),head(i),conduc(i)
    9   CONTINUE
        WRITE(20,303)evapor,actran
        WRITE(20,302)dwt
C
C Outputs for plant growth algorithm
C
        WRITE(61,600)j,rotdep,lai,wtruse
        DO 10 i =1,nolyrs-1
          WRITE(61,601)i,rdens(i)
   10   CONTINUE
C
    4 CONTINUE
C
  100 FORMAT(a1)
  200 FORMAT(1x,4(f10.8,2x))
  201 FORMAT(1x,f10.4,2x,e14.8,2x,f6.4)
  202 FORMAT(1x,4(f14.8,2x))
  203 FORMAT(1x,i3)
  204 FORMAT(1x,i4,2x,f12.8,2x,i3)
  205 FORMAT(1x,2(f12.6,2x))
  206 FORMAT(1x,5(f12.6,2x))
  208 FORMAT(1x,i2)
  209 FORMAT(1x,f16.6)
  210 FORMAT(1x,i4,2x,f10.8,3x,2(e18.10,2x))
  211 FORMAT(1x//1x,'Simulation Day Number ',i3)
  220 FORMAT(1x,3(f10.6,2x))
  221 FORMAT(1x,2(f18.10,2x))
  299 FORMAT(1x,i4)
  300 FORMAT(1x,f10.8,3x,2(e18.10,2x))
  301 FORMAT(1x,f8.2,2x,f8.3)
  302 FORMAT(1x,f6.2)
  303 FORMAT(1x,2(f8.3,2x))
  500 FORMAT(1x,i3)
  501 FORMAT(1x,5(f10.4,3x))
  502 FORMAT(1x,5(f12.4,3x))
  503 FORMAT(1x,i3,3x,2(f10.4,3x))
  504 FORMAT(1x,f10.4)
  505 FORMAT(1x,i3)
  506 FORMAT(1x,i3)
  507 FORMAT(1x,3(i5,3x))
  509 FORMAT(1x,i3,3x,f10.4)
  600 FORMAT(1x,i4,4x,i3,4x,f10.5,4x,f20.4)
  601 FORMAT(1x,i3,4x,f10.5)
C
      STOP
      END

C ***************************************************************************
***
```

```
C *
  *
C *
  *
C *                PARAMETER DETERMINATION CONTROL SUBROUTINES
  *
C *
  *
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Sys_Attrib
C
C The purpose of this routine is to calculate the parameters used in the late
r
C simulation for differing conditions e.g soil type. It does this by directin
g
C the program to further subroutines according to directions chosen by the
C user, the results are then written into files, where the information is
C stored for later retreival by the main program. This routine is only called
C when changes to the existing data are required.
C
C Local Variable Names and Meanings:
C ts(n) =saturated moisture content of the nth layer
C pb(n) =bulk density of the nth layer
C sand(n),silt(n),clay(n) =sand, silt and clay percentages of the nth layer
C vco(n),co(n),med(n) =sand grade sizes of nth layer in terms of the total
C   fine(n),vfi(n)      sand content
C cosi(n),fisi(n) =coarse and fine silt percentage content of the nth layer
C coarsf(n) =coarse fragment content of the nth layer
C d(n) =depth of nth layer
C mn(j) =number of channels per square metre in the jth macropore class
C mr(j) =radii of channels in jth macropore class
C md(j) =layer into which channels extend into, in the jth class
C dec,decis =character variables whose answer determines choice
C
C Called By:
C PROGRAM Soil_Atmosphere_Simulation
C
C Author: Darryn Evans
C Created: February 1988
C Updated: 23rd November 1988
C
      REAL ts(50),pb(50),sand(50),silt(50),clay(50),vco(50),co(50)
      REAL med(50),fine(50),vfi(50),cosi(50),fisi(50),coarsf(50),d(50)
      REAL mr(10)
      INTEGER mn(10),tmn,msd(10),nolyrs,permcl
      CHARACTER dec,decis,decis1
C
      PRINT*,'Do you want to change the soil properties y or n ',
     1                    '[press return]'
      READ(*,100)dec
      IF (dec .eq. 'y') THEN
        PRINT*,'Input the number of soil layers [max =50]'
        READ*,nolyrs
        DO 1 j =1,nolyrs
          PRINT*,'layer is',j
          PRINT*,'Input Bulk Density [press return]'
          READ*,pb(j)
          ts(j) =0.9*(1.0-(pb(j)/2.65))
          PRINT*,'Input Sand, Silt and Clay Content',
     1                                  '[press return]'
          READ*,sand(j),silt(j),clay(j)
          PRINT*,'Input Sand Fraction starting with Very'
     1                      ,' Coarse [press return]'
          READ*,vco(j),co(j),med(j),fine(j),vfi(j)
```

```
            PRINT*,'Input Silt Fractions starting with ',
     1                'Coarse [press return]'
            READ*,cosi(j),fisi(j)
            PRINT*,'Input Coarse Fragments Fraction ',
     1                 '[press return]'
            READ*,coarsf(j)
            PRINT*,'Depth of layer ',j,' [press return]'
            READ*,d(j)
C
C Open the file into which the values are to be input
C
     1    CONTINUE
          OPEN(50,file='Vale_Props',status='new',form='formatted'
     1                ,access='direct',recl=80)
          WRITE(50,299,rec=1)nolyrs
          DO 2 j =1,nolyrs
            k=(((j-1)*4)+1)
            WRITE(50,101,rec=k+1)ts(j),pb(j),sand(j),silt(j),clay(j)
            WRITE(50,102,rec=k+2)vco(j),co(j),med(j),fine(j),vfi(j)
            WRITE(50,103,rec=k+3)cosi(j),fisi(j),coarsf(j)
            WRITE(50,104,rec=k+4)d(j)
     2    CONTINUE
          CLOSE (50)
        END IF
C
C Input values for macropores if required
C
        PRINT*,'Do you want to change the macropore variables y or n'
     1                ,' [press return]'
        READ(*,100)decis
        IF (decis .eq. 'y') THEN
          PRINT*,'Input the Number of Permanent Macropore Classes'
          READ*,permcl
          DO j =1,permcl
            PRINT*,'Input the Number of Channels Within Class ',j
            READ*,mn(j)
            PRINT*,'Input the Radii of thes channels (cm)'
            READ*,mr(j)
            PRINT*,'Which Layer do these Impregnate'
            READ*,msd(j)
          END DO
C
C File 44 contains the macropore variables as input
C
          OPEN(44,file='Macro_Props',status='new',form='formatted'
     1                ,access='direct',recl=80)
          WRITE(44,120,rec=1)permcl
          DO j =1,permcl
            WRITE(44,121,rec=j+1)mn(j),mr(j),msd(j)
          END DO
          CLOSE (44)
        END IF
C
C Calculate the simulation parameters and input the plant parameters if
C required
C
        CALL Calculate_Parameter_Values
        PRINT*,'Input the value of R in cm [ press return ]'
        READ*,r
        CALL Husseins_Constants(r)
C
        PRINT*,'Do you want to change the plant characteristics y or n',
     1                                  ' [ press return ]'
        READ(*,100)decis1
        IF (decis1 .eq. 'y') THEN
          CALL Input_Plant_Characteristics
        END IF
C
```

```
  100 FORMAT(a1)
  101 FORMAT(1x,5(f8.4,2x))
  102 FORMAT(1x,5(f8.4,2x))
  103 FORMAT(1x,3(f8.4,2x))
  104 FORMAT(1x,f12.4)
  120 FORMAT(1x,i3)
  121 FORMAT(1x,i4,2x,f12.8,2x,i3)
  299 FORMAT(1x,i4)
C
      RETURN
      END
C
C ***************************************************************************
**
C *
 *
C ***************************************************************************
**
C
      SUBROUTINE Calculate_Parameter_Values
C
C The purpose of this subroutine is to calculate the various parameters
C needed for the simulation and then to store them in a file where the
C main program can retrieve them.
C
C Local Variable Names and Meanings:
C ts(n) =saturated moisture content of the nth layer
C pb(n) =bulk density of the nth layer
C sand(n),silt(n),clay(n) =sand, silt and clay percentages of the nth layer
C vco(n),co(n),med(n) =sand grade sizes of nth layer in terms of the total
C   fine(n),vfi(n)      sand content
C cosi(n),fisi(n) =coarse and fine silt percentage content of the nth layer
C coarsf(n) =coarse fragment content of the nth layer
C d(n) =depth of nth layer
C mn(j) =number of channels per square metre of the jth macropore class
C mr(j)=radii of macropore channels in jth class
C msd(j) =layer into which macropore channels extend into, in class j
C cl,bl,dl,al,bla,cla,ul =Regression variables determined from soils data
C el,fl,gl,hl,sl,tl        from several sources
C tthird =moisture content at a third of a bar, determined by regression
C bar15 =moisture content at 15 bars, determined by regression
C tr(n) =residual moisture content: in this case equivalenced to bar15
C xsmc =volumetric water content above a third of a bar
C relmc =volumetric moisture content found above 15 bars
C nl(n),alpha(n) =parametres from Genuchtens(1980) equation    -
C hb =head at bubbling Brooks and Corey(1966)
C l =lambda parameter of Brooks and Corey Equation (1966)
C sf(n) =suction across wetting front boundary in nth layer from Rawls et al
84
C kf(n) =conductivity of wetting zone Rawls et al 1984
C dwt,wt =depth to water table from layer middle
C mc(n) =moisture content of layer before infiltrating wetting front includes
C        macropore wetting fronts
C mci(n) =moisture content excluding all wetting fronts
C moist(n) =moisture content of layer averaged
C mp(j) =percentge of area macropores of the jth class
C hd(j) =height from the base of the macropore to the ground surface for
C        macropore channels of the jth class
C sv =amount of shrinkage between 50cm and 15000cm in terms of volume
C tu =moisture content at 50cm
C tt =moisture content at 15000cm
C shrgra =gradient of linear relationship for calculating shrinkage
C shrc =intercept of above on y-axis
C permcl =number of permanent macropore classes
C
C Called By:
C SUBROUTINE Soil_Properties
C
```

```
C Author: Darryn Evans
C Created: 16th November 1988
C Updated: 20th March 1989
C
      REAL c1,b1,d1,a1,b1a,c1a,e1,f1,g1,h1,s1,t1,u1,ts(50),pb(50)
      REAL sand(50),silt(50),clay(50),vco(50),co(50),med(50)
      REAL fine(50),vfi(50),cosi(50),fisi(50),coarsf(50),d(50)
      REAL tthird,bar15,tr(50),xsmc,relmc,n1(50),r,alpha(50),sf(50)
      REAL hb,1,kf(50),dwt(50),wt,mc(50),mci(50),moist(50)
      REAL mr(10),mp(10),hd(10),sum,aa1,a2,a3,n3,n4,n5,n6,n7,tt1
      REAL t2,t3,t4
      INTEGER k,mn(10),msd(10),nolyrs,k2,permcl
C
      DATA c1,b1,d1 / 8.059426E-03,2.598513E-03,-0.06202 /
      DATA a1,b1a,c1a / -1.45906E-03,-1.53743E-03,4.293512E-03 /
      DATA e1,f1,g1 / -3.78574,5.534973E-03,-6.34563E-03 /
      DATA h1,s1,t1 / -1.96274,1.05870,3.94544 /
      DATA u1 / 9.413364E-03 /
      DATA n3,n4,n5 / 3.156677E-05,.06587,4.859883E-06 /
      DATA n6,n7 / -2.64358E-03,-4.28877E-04 /
      DATA aa1,a2,a3 / 1.765137E-03,2.523676E-03,-3.97461E-06 /
      DATA tt1,t3,t4 / 1.578954E-08,4.189656E-03,-3.60168E-04 /
C
C Read in soil properties for each layer and calculate Genuchtens constants
C and wetting front constants
C
      OPEN(50,file='Vale_Props',status='old',form='formatted'
     1                              ,access='direct',recl=80)
      READ(50,299,rec=1)nolyrs
      DO 1 j =1,nolyrs
        k=(((j-1)*4)+1)
        READ(50,101,rec=k+1)ts(j),pb(j),sand(j),silt(j),clay(j)
        READ(50,102,rec=k+2)vco(j),co(j),med(j),fine(j),vfi(j)
        READ(50,103,rec=k+3)cosi(j),fisi(j),coarsf(j)
        READ(50,104,rec=k+4)d(j)
C
C Calculate Genuchtens equation parameters for each layer
C
        tthird =0.13519+(c1*clay(j))+(b1*silt(j))+(d1*pb(j))
        bar15 =0.04367+(tt1*((sand(j)**2.)*(clay(j)**2.)))
     1          +(t3*clay(j))+(t4*sand(j))
        tr(j) =bar15
        n1(j) =1.33295+(n3*(fine(j)**3.))+(n4*fine(j))+
     1          (n5*(sand(j)**3.))+(n6*(fine(j)**2.))+(n7*(sand(j)**2.))
        alpha(j) =.0267+(aa1*co(j))+(a2*vco(j))+(a3*(silt(j)**2.))
C
C Calculate wetting front parameters from alpha parameter taken from Rawls
C et al 1984
C
        relmc =ts(j)-bar15
        hb =1/alpha(j)
        1 =n1(j)-1
        sf(j) =(2+(3*1))/(1+(3*1))*(hb/2)
        kf(j) =(21*((relmc/hb)**2)*((1**2)/((1+1)*(1+2))))/2
    1 CONTINUE
      CLOSE (50)
C
C Read in initial values of moisture content, putting initial water contents
C equivalent to the field capacity, i.e distance of centre of layer above the

C water table
C
      sum =100.
      DO 2 j =(nolyrs-1),1,-1
        sum =sum+d(j)
        dwt(j) =sum
    2 CONTINUE
      DO 3 j =1,nolyrs
```

```
        IF (j .eq. nolyrs) THEN
          mc(j) =ts(j)
        ELSE
          wt =(dwt(j)-d(j))+(d(j)/2)
          mc(j) =tr(j)+((ts(j)-tr(j))/((1+((alpha(j)*wt)**n1(j)))**
     1                              (1-(1/n1(j)))))
        END IF
        moist(j) =mc(j)
        mci(j) =mc(j)
    3 CONTINUE
        DO  j =1,nolyrs
          moist(j) =mc(j)
          mci(j) =mc(j)
        END DO
C
C calculate macropore parameters outstanding
C
      OPEN(44,file='Macro_Props',status='old',form='formatted'
     1                  ,access='direct',recl=80)
      READ(44,203,rec=1)permcl
      DO j =1,permcl
        READ(44,204,rec=j+1)mn(j),mr(j),msd(j)
        mp(j) =(((3.142*(mr(j)**2.))*real(mn(j)))/10000.)*100.
        sumhd =0.
        DO i =1,msd(j)
          sumhd =sumhd+d(i)
        END DO
        hd(j) =sumhd
      END DO
      sv =(0.35*clay(1))-4.2
      IF (sv .gt. 0.) THEN
        m1 =1-(1/n1(1))
        tu =tr(1)+((ts(1)-tr(1))/((1+((alpha(1)*50.)**n1(1)))**m1))
        tt =tr(1)+((ts(1)-tr(1))/((1+((alpha(1)*15000.)**n1(1)))**m1))
        shrgra =-sv/(tu-tt)
        shrc =sv-(shgrad*tt)
      END IF
C
C Now write parameters out to an external file to be input into the main
C program
C
      OPEN(52,file='Vale_Params',status='new',form='formatted',
     1                        access='direct',recl=80)
      WRITE(52,299,rec=1)nolyrs
      DO 7 j =1,nolyrs
        k =(((j-1)*3)+1)
        WRITE(52,200,rec=k+1)n1(j),alpha(j),tr(j),ts(j)
        WRITE(52,201,rec=k+2)sf(j),kf(j)
        WRITE(52,202,rec=k+3)d(j),mc(j),mci(j),moist(j)
    7 CONTINUE
      CLOSE (52)
      WRITE(53,203)permcl
      DO j =1,permcl
        WRITE(53,204)mn(j),mr(j),msd(j)
        WRITE(53,205)mp(j),hd(j)
      END DO
      WRITE(53,206)shgra,shrc,sv,tu,tt
      CLOSE(53)
C
  101 FORMAT(1x,5(f8.4,2x))
  102 FORMAT(1x,5(f8.4,2x))
  103 FORMAT(1x,3(f8.4,2x))
  104 FORMAT(1x,f12.4)
  200 FORMAT(1x,4(f10.8,2x))
  201 FORMAT(1x,f10.4,2x,e14.8)
  202 FORMAT(1x,4(f14.8,2x))
  203 FORMAT(1x,i3)
  204 FORMAT(1x,i4,2x,f12.8,2x,i3)
```

```
  205 FORMAT(1x,2(f12.6,2x))
  206 FORMAT(1x,5(f12.6,2x))
  299 FORMAT(1x,i4)
C
      RETURN
      END
C
C ***************************************************************************
**
C *
 *
C ***************************************************************************
**
C
      SUBROUTINE Husseins_Constants(maxran)
C
C The purpose of this routine is to create a file into which Husseins paramet
ers
C for the groundwater and the evaporation from bare soil.
C
C Variable Names and Meanings
C D4/Cd = Groundwater Parameters
C p = bare soil evaporation parameter
C maxran =maximum amount of rain before rain is distributed according to
C         the parabola method
C
C Called By
C      SUBROUTINE Sys_Attrib
C
C Author: Darryn Evans
C Created: 1st March
C Updated: 20th March 1989
C
      REAL d4,cd,p,maxran
C
      DATA d4,cd,p / 2.5,0.002,2. /
C
      OPEN(18,file='xhudata',status='new',access='direct',
     1          form='formatted',recl=16)
      WRITE(18,100,rec=1) d4
      WRITE(18,100,rec=2) cd
      WRITE(18,100,rec=3) p
      WRITE(18,100,rec=4) maxran
C
  100 FORMAT(f16.6)
      CLOSE(18)
C
      RETURN
      END
C
C ***************************************************************************
***
C *
 *
C ***************************************************************************
***
C
      SUBROUTINE Input_Plant_Characteristics
C
C This subroutine is used to change and alter the plant characteristics
C important within the subroutine plant_growth
C
C Local Variable Names and Meanings
C a4,b4 =parameter values in the Growth units equation
C minrat, maxrat =minimum and maximum partitioning ratios of assimilates betw
een
C                 above and below ground development of the plant
C koef,lam =parameter values of the partitioning equation
```

```
C lc,rc =conversion of growth units into leaf area and root growth respective
ly
C maxlai,c4 =parameter values used to relate actual leaf area index (lai) to
C           potential
C dens(j) =root density in jth layer cm/cubic cm
C index =initial lai
C havest =lai index value at which grass is harvested
C plant =number denoting the appropriate classification of the plant
C dep =deepest soil layer into which the roots extend
C nocrop =number of arable crops per year ( If plant =2)
C emerg =number of days after start of simulation for emergence of crops
C anth =number of days after start of simulation for anthesis to set in
C harv =number of days after start of simulation to harvest
C out =number of the output file
C typ =indicates whether crops grown change over the simulation
C
C Called By:
C SUBROUTINE Sys_Attrib
C
C Author: Darryn Evans
C Created: 19/06/89
C Updated:
C
      REAL a4,b4,minrat,maxrat,koef,lam,lc,maxlai,c4,rc,dens(50)
      REAL index,havest
      INTEGER plant,dep,nocrop,emerg,anth,harv,out
      CHARACTER typ
C
      out =84
      PRINT*,'From the following list indicate by number the plant ',
     1                                          'type [ press return ]'
      PRINT*,'1. Permanent Pasture'
      PRINT*,'2. Monocotyledon Crops'
      READ*,plant
      WRITE(out,100)plant
C
      IF (plant .eq. 1) THEN
C
C input equation parameter values
C
        PRINT*,'Input Parameters a4,b4 & c4 [ press return ]'
        READ*,a4,b4,c4
        PRINT*,'Partitioning parametrs minrat,maxrat [ press return ]'
        READ*,minrat,maxrat
        PRINT*,'Parameters koef,lam,lc,rc,maxlai [ press return ]' -
        READ*,koef,lam,lc,rc,maxlai
C
C input initial state of the crop at the start of the simulation and write
C results to the data file
C
        PRINT*,'Input the layer in which the roots initially extend',
     1                                ' into '
        READ*,dep
        DO 1 j =1,dep
          PRINT*,'What is the initial root density of layer ',j
          READ*,dens(j)
    1   CONTINUE
        PRINT*,'Input the Initial LAI'
        READ*,index
        PRINT*,'Input value of lai at which crop is harvested'
        READ*,havest
C
        WRITE(out,101)a4,b4,c4,lc,rc
        WRITE(out,102)maxlai,minrat,maxrat,koef,lam
        WRITE(out,103)dep,index,havest
        DO 2 j =1,dep
          WRITE(out,104)dens(j)
    2   CONTINUE
```

```
      ELSE IF (plant .eq. 2) THEN
        PRINT*,'Input the Number of Crops per Simulation'
        READ*,nocrop
        WRITE(out,100)nocrop
        PRINT*,'Is it the same crop each time y or n'
        READ(*,105)typ
        IF (typ .eq. 'y') THEN
          PRINT*,'Input Parameters a4,b4 & c4 '
          READ*,a4,b4,c4
          PRINT*,'Partitioning parametrs minrat,maxrat '
          READ*,minrat,maxrat
          PRINT*,'Parameters koef,lam,lc,rc,maxlai'
          READ*,koef,lam,lc,rc,maxlai
          PRINT*,'Parameters max,e4'
          READ*,max,e4
          PRINT*,'Input the layer in which the roots initially extend'
     1                                , ' into '
          READ*,dep
          DO 3 j =1,dep
            PRINT*,'What is the initial root density of layer ',j
            READ*,dens(j)
    3     CONTINUE
          PRINT*,'Input the Initial LAI'
          READ*,index
        END IF
C
        DO 4 j =1,nocrop
          PRINT*,'Input number of days after simulation has started'
          PRINT*,'Emergence of crop ',j
          READ*,emerg
          PRINT*,'Anthesis of crop ',j
          READ*,anth
          PRINT*,'Harvest of crop ',j
          READ*,harv
C
          IF (typ .eq. 'n') THEN
            PRINT*,'Input Parameters a4,b4 & c4 '
            READ*,a4,b4,c4
            PRINT*,'Partitioning parametrs minrat,maxrat '
            READ*,minrat,maxrat
            PRINT*,'Parameters koef,lam,lc,rc,maxlai'
            READ*,koef,lam,lc,rc,maxlai
            PRINT*,'Input the layer in which the roots initially'
     1                                  ,' extend into '
            READ*,dep
            DO 5 i =1,dep
              PRINT*,'What is the initial root density of layer ',i
              READ*,dens(i)
    5       CONTINUE
            PRINT*,'Input the Initial LAI'
            READ*,index
          END IF
          WRITE(out,106)j
          WRITE(out,107)emerg,anth,harv
          WRITE(out,101)a4,b4,c4,lc,rc
          WRITE(out,102)maxlai,minrat,maxrat,koef,lam
          WRITE(out,109)dep,index
          DO 6 i =1,dep
            WRITE(out,104)dens(i)
    6     CONTINUE
    4   CONTINUE
      END IF
C
C Format Statements
C
  100 FORMAT(1x,i3)
  101 FORMAT(1x,5(f10.4,3x))
  102 FORMAT(1x,5(f12.4,3x))
```

```
  103 FORMAT(1x,i3,3x,2(f10.4,3x))
  104 FORMAT(1x,f10.4)
  105 FORMAT(a1)
  106 FORMAT(1x,i3)
  107 FORMAT(1x,3(i5,3x))
  109 FORMAT(1x,i3,3x,f10.4)
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C *
  *
C *             ROUTINES CONTROL THE LENGTH OF THE TIMESTEP FOR
  *
C *                        REDISTRIBUTION ALGORITHMS
  *
C *
  *
C *
  *
C ***************************************************************************
***
C
C The purpose of these subroutine is to determine the length of the timestep
C over which the application of transfers across the boundaries should occur
C
      SUBROUTINE One_Day
C
C Local Variable Names and Meanings:
C short =logical variable to indicate if too much water has been passed
C r =rainfall
C
C Subroutines Called:
C One_Hour =changes timestep to one hour intervals
C Calculate_Exchanges =calculates the exchanges across the boundaries from
C                      infiltration, evapotranspiration, macropores,
C                      groundwater flow and percolation
C
C Called By:
C PROGRAM Soil_Atmosphere_Simulation
C
C Author: Darryn Evans
C Created: 24th November 1988
C Updated: 20th March 1989
C
      REAL r
      LOGICAL short
C
      INCLUDE 'main.inc'
C
      r =0.
      time =86400
      CALL Calculate_Exchanges(short,r)
C
      IF (short .eq. .true.) THEN
        CALL One_Hour
      END IF
C
      RETURN
      END
C
C ***************************************************************************
***
C *
```

```
*
C ********************************************************************************
***
C
      SUBROUTINE Infil_Time(totalr)
C
C Routine enables distribution of the daily rain into hourly blocks the metho
d
C depending on the exact amount.
C
C Local Variable Names and Meanings:
C short =logical variable to indicate if too much water has been passed
C totalr =amount of daily rainfall
C precip(n) =amount of precipitation in the nth hour
C rain =hourly amount of rain passed to the subroutines
C
C Subroutines Called:
C One_Minute =changes timestep to one minute intervals
C Calculate_Exchanges =calculates the exchanges across the boundaries from
C                      infiltration,macropores,groundwater and percolation
C
C Called By:
C PROGRAM Soil_Atmosphere_Simulation
C
C Author: Darryn Evans
C Created: 30th November 1988
C Updated: 20th March 1989
C
      REAL totalr,precip(24),ran
      LOGICAL short
C
      INCLUDE 'main.inc'
C
      IF (totalr .gt. maxran) THEN
        CALL Distribute_Rain(precip,totalr)
      ELSE
        CALL Random_Rain(precip,totalr)
      END IF
C
      DO 1 j =1,24
        time =3600.
        ran =precip(j)
        CALL Calculate_Exchanges(short,ran)
        IF (short .eq. .true.) THEN
          CALL One_Minute(ran)
        END IF
    1 CONTINUE
C
      RETURN
      END
C
C ********************************************************************************
***
C *
  *
C ********************************************************************************
***
C
      SUBROUTINE One_Hour
C
C Local Variable Names and Meanings:
C short =logical variable to indicate if too much water has been passed
C ran =amount of rainfall
C
C Subroutines Called:
C One_Minute =changes timestep to one minute intervals
C Calculate_Exchanges =calculates the exchanges across the boundaries from
C                      infiltration, evapotranspiration, macropores,
```

```
C                       groundwater flow and percolation
C
C Called By:
C PROGRAM Soil_Atmosphere_Simulation
C SUBROUTINE One_Day
C
C Author: Darryn Evans
C Created: 24th November 1988
C Updated: 20th March 1989
C
      REAL ran
      LOGICAL short
C
      INCLUDE 'main.inc'
C
      ran =0.
      DO 1 j =1,24
        time =3600.
        CALL Calculate_Exchanges(short,ran)
        IF (short .eq. .true.) THEN
          CALL One_Minute(ran)
        END IF
    1 CONTINUE
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE One_Minute(r)
C
C Local Variable Names and Meanings:
C r =amount of rainfall for the hour
C ran =amount of rainfall for each minute
C short =logical variable to indicate if too much water has been passed
C
C Subroutines Called:
C One_Second =changes timestep to one second intervals
C Calculate_Exchanges =calculates the exchanges across the boundaries from
C                      infiltration, evapotranspiration, macropores,
C                      groundwater flow and percolation
C
C Called By:
C SUBROUTINE One_Hour
C
C Author: Darryn Evans
C Created: 24th November 1988
C Updated: 20th March 1989
C
      REAL r,ran
      LOGICAL short
C
      INCLUDE 'main.inc'
C
      ran =r/60.
      DO 1 j =1,60
        time =60.
        CALL Calculate_Exchanges(short,ran)
        IF (short .eq. .true.) THEN
          CALL One_Second(ran)
        END IF
    1 CONTINUE
C
```

```
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE One_Second(r)
C
C Local Variable Names and Meanings:
C r =amount of rainfall for the minute
C ran =amount of rainfall for each second
C short =logical variable to indicate if too much water has been passed
C
C Subroutines Called:
C One_Tenth =changes timestep to one tenth of a second intervals
C Calculate_Exchanges =calculates the exchanges across the boundaries from
C                      infiltration, evapotranspiration, macropores,
C                      groundwater flow and percolation
C
C Called By:
C SUBROUTINE One_Minute
C
C Author: Darryn Evans
C Created: 24th November 1988
C Updated: 20th March 1989
C
      REAL r,ran
      LOGICAL short
C
      INCLUDE 'main.inc'
C
      ran =r/60.
      DO 1 j =1,60
        time =1.
        CALL Calculate_Exchanges(short,ran)
        IF (short .eq. .true.) THEN
          CALL One_Tenth(ran)
        END IF
    1 CONTINUE
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE One_Tenth(r)
C
C Local Variable Names and Meanings:
C r =amount of rain in second
C ran =amount of rain in one-tenth of a second
C short =logical variable to indicate if too much water has been passed
C
C Subroutines Called:
C Calculate_Exchanges =calculates the exchanges across the boundaries from
C                      infiltration, evapotranspiration, macropores,
C                      groundwater flow and percolation
C
C Called By:
C SUBROUTINE One_Second
```

```
C
C Author: Darryn Evans
C Created: 24th November 1988
C Updated: 20th March 1989
C
      REAL r,ran
      LOGICAL short
C
      INCLUDE 'main.inc'
C
      ran =r/10.
      DO 1 j =1,10
        time =0.1
        CALL Calculate_Exchanges(short,ran)
    1 CONTINUE
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C *
  *
C *              CALCULATES AND APPLIES REDISTRIBUTION ALGORITHMS
  *
C *
  *
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Calculate_Exchanges(much,ra)
C
C The purpose of this routine is to calculate the exchanges between the layer
s
C for the relevant timestep. Temporary calculations and variables are introdu
ced
C so that if exchanges are found to be too excessive, the timestep can be
C reduced without affecting the original values
C
C Local Variable Names and Meanings
C much =logical variable indicating whether too much water has been exchanged
C ra =amount of precipitation in timestep
C inf =indicates if there are any wetting fronts in the profile
C summmd =sum of height of water in macropore classes
C tmn =total number of macropores
C
C Subroutines Called:
C Infiltration =calculates the amount of infiltration and amount ponded using
C               Mein and Larsons (1973) description of the Green and Ampt
C               Equation
C Macropore =calculates the macropore flow using concepts derived from Beven
C            and Clarke (1986) of the Green and Ampt Equation
C Redist_Percolation =is used to calculate percolation with the prescence of
C                     wetting fronts in the profile
C Percolation =calculates normal percolation
C
C Called By:
C SUBROUTINE One_Day
C SUBROUTINE One_Hour
C SUBROUTINE One_Minute
C SUBROUTINE One_Second
C SUBROUTINE One_Tenth
C
C Author: Darryn Evans
```

```
C Created: 30th November 1988
C Updated: 20th March 1989
C
      REAL summmd,ra
      INTEGER tmn
      LOGICAL much,inf
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
C initialize the temporary values
C
      DO 1 j =1,nolyrs
        wet(j) =moist(j)
        wetmc(j) =mc(j)
        wetmci(j) =mci(j)
    1 CONTINUE
      DO j =1,noclas
        DO i =1,nolyrs
          mwf(j,i) =mrf(j,i)
        END DO
        md(j) =mamd(j)
      END DO
      wfr =wetfr
      len =ln
      meth =method
      runoff =ro
      ponded =pond
      inf =.false.
      perd =1
C
C Start the calculations
C
C See if the infiltration routines are needed i.e is there any rainfall
C or has there been in the previous one.
C
      IF ((ra .gt. 0.) .or. (len .gt. 0.) .or. (wfr .gt. 1)) THEN
        IF (ra .gt. 0.) THEN
          CALL Infiltration(ra)
          inf =.true.
        ELSE
          DO j =1,nolyrs
            IF ((j .eq. wfr) .and. (wetmci(j) .lt. wetmc(j))) THEN
              wetmci(j) =(((d(j)-len)*wetmci(j))+(len*ts(j)))/d(j)
            ELSE
              wetmci(j) =wet(j)
            END IF
            wetmc(j) =wet(j)
          END DO
          wfr =1
          meth =1
          len =0.
        END IF
      END IF
C
C If there is any water on the surface, in the macropores or if there
C are any wetting fronts associated with the macropores call the
C macropore subroutines
C
      summd =0.
      tmn =0
      sumf =0.
      DO j =1,noclas
        DO i =1,nolyrs
          sumf =sumf+mwf(j,i)
        END DO
        summd =summd+md(j)
        tmn =tmn+mn(j)
```

```
      END DO
      IF ((((runoff+ponded) .gt. 0.) .or. (summmd .gt. 0.)) .and.
     1              (tmn .gt. 0)) THEN
        CALL Macropore
        DO j =1,nolyrs
          IF (j .eq. wfr) THEN
            wet(j) =((len*ts(j))+((d(j)-len)*wetmc(j)))/d(j)
          ELSE
            wet(j) =wetmc(j)
          END IF
        END DO
        inf =.true.
      ELSE IF (sumf .gt. 0.) THEN
        inf =.true.
      END IF
C
C If there has been any infiltration and/or there are any wetting fronts
C call redist_percolation else call percolation
C
      IF (inf .eq. .true.) THEN
        CALL Redist_Percolation(much)
      ELSE
        CALL Percolation(much)
      END IF
C
C now if not too much water has been passed the exchanges are implemented
C
      IF (much .eq. .false.) THEN
        DO 3 j =1,nolyrs
          moist(j) =wet(j)
          mc(j) =wetmc(j)
          mci(j) =wetmci(j)
    3   CONTINUE
        DO j =1,noclas
          DO i =1,nolyrs
            mrf(j,i) =mwf(j,i)
          END DO
          mamd(j) =md(j)
        END DO
        wetfr =wfr
        ln =len
        ro =runoff
        pond =ponded
        method =meth
      END IF
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C *
  *
C *                        RAINFALL DISTRIBUTION ROUTINES
  *
C *
  *
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Distribute_Rain(rfall,total)
C
C The purpose of this subroutine is to distribute high daily rainfall totals
C i.e rainfall in excess of maxran, into six-hour blocks thence into hourly
```

```
C units.
C
C Local Variable Names and Meanings
C rorder(i) =rainfall ordered into rank
C rfall(i) =amount of precipitation in Ith hour
C prog(i) =array containing a randomly generated sequence of the numbers 1..2
4
C          which are not repeated, used to distribute maxblk, maxmin, medmin
C          and minmin values randomly into the six-hour blocks
C total =daily rainfall total
C a/b/c =parameters used to determine the rainfall distribution curve
C area =amount of rainfall left after distribution into 24 hour slots,
C       distributed evenly into these slots. Occurs when intercept of x-axis
C       occurs after 24hrs or not at all
C
C Called By:
C      PROGRAM Soil_Atmosphere_Simulation
C
C Subroutines Called:
C Random_Progression =randomly determines the sequence of blocks data is to b
e
C                     input into
C
C Author: Darryn Evans
C Created: 24th March 1988
C Updated: 16th November 1988
C
      REAL a,b,c,total,sum,rfperc(24),rorder(24),rfall(24)
      INTEGER k1,prog(24),kount
C
C initialize array values and convert rain into mms
C
      DO 1 j =1,24
        rfall(j) =0.
    1 CONTINUE
      t1 =total
      total =total*10.
C
C calculate parameter values a,b and c from regression equations
C
      a =0.041482*total+1.040872
      b =-0.08647*total+4.889794
      c =67.6
C
C Calculate the percentage area for each hour of the curve
C
      kount =1
      DO WHILE ((total .gt. 0.) .and. (kount .lt. 25))
        IF (kount .eq. 1) THEN
          rfperc(kount) =(c*log(kount+a)-(b*kount))-(c*log(a))
        ELSE
          rfperc(kount) =(((c*log(kount+a))-(b*kount))-((c*log(
     1                                  (kount-1)+a))-(b*(kount-1))))
        END IF
        rorder(kount) =(rfperc(kount)/100.)*total
        t =total-rorder(kount)
        IF (t .lt. 0.) THEN
          rorder(kount) =rorder(kount)+t
          total =0.
        ELSE
          total =t
        END IF
        kount =kount+1
      END DO
C
C if any rainfall is outstanding due either to the intercept b being greater
C than 24 or negative then distribute this evenly to the hours of the day
C
```

```
      IF (total .gt. 0.) THEN
        DO 2 i =1,24
          rorder(i) =rorder(i)+(total/24.)
    2   CONTINUE
      END IF
C
C now produce a random number progression to determine in to which hour each
C value will go
C
      CALL Random_Progression(prog,24)
C
C convert rainfall from mms to cms for use in the simulation
C
      DO 3 j =1,24
        rfall(prog(j)) =rorder(j)/10.
    3 CONTINUE
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Random_Rain(rfall,total)
C
C This subroutine distributes smaller daily rainfall totals, i.e less than
C maxran randomly into hourly amounts
C
C Variable Names and Meanings
C rfall(i) =the total rainfall in the Ith hour
C total =total daily rainfall
C rain =rainfall left after each distribution
C units =integer value indicating the number of 0.5mm of rainfall in a day
C x =random number between 0. and 1.0 passed back by Subroutine Rand
C ko/kol =converts x into the interval of random numbers required
C tothr =calculated rainfall for the hour
C
C Called By:
C      PROGRAM Soil_Atmosphere_Simulation
C
C Subroutines Called:
C Rand =calculates a random number between 0. and 1.0
C
C Author: Darryn Evans
C Created: 24th March 1988
C Updated: 16th November 1988
C
      REAL total,tothr,x,xx,rain
      INTEGER units,ko,kol
      DIMENSION rfall(24)
C
C Initialize array values and convert rainfall into mms
C
      DO 1 j =1,24
        rfall(j) = 0.
    1 CONTINUE
      t1 =total
      total =total*10.
C
C rainfall is firstly split up into 0.5mm blocks any remainder is dealt with
C later
C
      units =int(total/0.5)
      rain =total
```

```
      xx =irand
      DO WHILE (units .gt. 0.)
        CALL Rand(x,xx)
        ko =int(real(units)*x)+1
        IF (ko .gt. units) THEN
          ko =units
        END IF
C
C ko is used as a multiplier to determine the rainfall during the hour
C
        tothr =real(ko)*0.5
C
C distribute this randomly into an hour block in the day
C
        xx =x
        CALL Rand(x,xx)
        kol =int(24.0*x)+1
        IF (kol .gt. 24) THEN
          kol =24
        END IF
C
C convert to cms for use in the main program
C
        rfall(kol) =rfall(kol)+(tothr/10.)
        units =int(units-ko)
        rain =rain-tothr
        xx =x
      END DO
C
C distribute any remaining rainfall
C
      IF (rain .gt. 0.) THEN
        CALL Rand(x,xx)
        kol =int(24.0*x)+1
        IF (kol .gt. 24) THEN
          kol =24
        END IF
        rfall(kol) =rfall(kol)+(rain/10.)
      END IF
      irand =x
C
      RETURN
      END
C
C ***********************************************************************
***
C *
  *
C ***********************************************************************
***
C
      SUBROUTINE Random_Progression(seq,num)
C
C Routine returns a randomly produced sequence of numbers in the range of 1..
num
C without duplicating a number. The sequence is num numbers long.
C
C Variable Names and Meanings
C seq(i) =array contains the Ith number in the progression
C num =number of maximum random value
C used,found =indicates if number has been processed before
C ko =random number integer found within the interval
C x =real random number in range 0. - 1.0
C count =used to position the numbers in the sequence and to indicate if it i
s
C         complete
C
C Called By:
```

```
C       SUBROUTINE Distribute_Rain
C
C Subroutines Called:
C Rand =calculates a real random number between 0. and 1.0
C
C Author: Darryn Evans
C Created: 24th March 1988
C Updated:
C
      REAL x,xx
      INTEGER seq(num),num,count,ko
      LOGICAL used,found
C
      count =1
C
C initialize array values to 0
C
      DO 4 i =1,num
        seq(i) =0
    4 CONTINUE
C
      x =irand
      DO 1 j =1,num
        count =count+1
        used =.true.
        DO WHILE (used .eq. .true.)
          found =.false.
          xx =x
          CALL Rand(x,xx)
          ko =int(real(num)*x)+1
          IF (ko .gt. num) THEN
            ko =num
          END IF
          IF (j .eq. 1) THEN
            seq(j) =ko
            used =.false.
          ELSE
            kount =1
            DO WHILE ((found .eq. .false.) .and.
     1                                        (kount .le. (count-1)))
              IF (ko .eq. seq(kount)) THEN
                found =.true.
              END IF
              kount =kount+1
            END DO
            IF (found .eq. .false.) THEN
              seq(j) =ko
              used =.false.
            END IF
          END IF
        END DO
    1 CONTINUE
        irand =x
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Rand(y,yy)
C
C This routine produces random numbers between 0.0 and 1.0, and is taken from
C a book by Meisner, L. P., and Organik, E. I., (1980), M represents the peri
```

```
od
C before the pattern starts repeating itself, RM is equal to M so numbers lie
 in
C the range indicated
C
C Called By:
C       SUBROUTINE Random_Progression
C       SUBROUTINE Random_Rain
C
C Typed in: 24th March 1988
C
       DATA k,j,m,rm /5701,3612,566927,566927.0/
C
       y =0.5
       ix =int(yy*rm)
       iran =mod(j*ix+k,m)
       y =(real(iran)+0.5)/rm
C
       RETURN
       END

C
C ***************************************************************************
***
C *
  *
C *
  *
C *                          INFILTRATION ROUTINES
  *
C *
  *
C *
  *
C ***************************************************************************
***
C
       SUBROUTINE Infiltration(precip)
C
C The purpose of this subroutine is to calculate the infiltration of water
C into each layer via the surface. This infiltration maybe calculated from
C ponded or non-ponded conditions.
C
C Local Variable Names and Meanings:-
C layer =a loop conting variable
C kb =harmonic mean of two conductivities across their boundary
C temp =used to store altered values of moisture so can change wetmci
C w =percentage area of layer inhabited by wetmci in relation to wetmc
C
C Subroutines Called:
C Redist =calculates whether a wetting front has encountered a layer boundary
C Infiltration_Cap =calculates the infiltration capacity of the soil
C Dissolve_Wetting_Front =redistributes water from the infiltrating wetting
C                         front within the layer
C Fine_Over_Coarse =calculates infiltration when a wetting front reaches a
C                   fine over coarse soil boundary
C
C Called By:
C SUBROUTINE Calculate_Exchanges
C
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
       REAL kb,temp(50),xcess(50),w,precip
       INTEGER layer,count
C
       INCLUDE 'main.inc'
```

```
      INCLUDE 'temp.inc'
C
      DO 1 j =1,nolyrs
        temp(j) =0.
        xcess(j) =0.
    1 CONTINUE
      perd =1
C
      rain =precip
      IF (rain .le. (kf(wfr)*time)) THEN
C
C wetting fronts, if present, are effectively destroyed
C
        IF ((wfr .gt. 1) .or. (len .gt. 0.)) THEN
          DO j =1,wfr
            wetmc(j) =wet(j)
          END DO
        END IF
        meth =1
        wfr =1
        len =0.
C
C start calculating infiltration
C
        temp(1) =wet(1)+(rain/d(1))
        IF (temp(1) .gt. ts(1)) THEN
          xcess(1) =(temp(1)-ts(1))*d(1)
          temp(1) =ts(1)
          layer =2
          tmlft =time
          DO WHILE ((xcess(layer-1) .gt. 0.) .and.
     1                                       (layer .le. (nolyrs-1)))
            wfr =layer
            len =0.
            tmlft =tmlft-(((precip-xcess(layer-1))/precip)*tmlft)
            CALL Infiltration_Cap
            IF (xcess(layer-1) .lt. fp) THEN
              IF (wetmc(layer) .lt. ts(layer)) THEN
                IF (xcess(layer-1) .gt. (kf(layer)*tmlft)) THEN
                  wfr =layer
                  len =xcess(layer-1)/(ts(wfr)-wetmc(wfr))
                  meth =2
                ELSE
                  wfr =1
                  len =0.
                END IF
                temp(layer) =wet(layer)+(xcess(layer-1)/d(layer))
                IF (temp(layer) .gt. ts(layer)) THEN
                  xcess(layer) =(temp(layer)-ts(layer))*d(layer)
                  temp(layer) =ts(layer)
                  rain =xcess(layer)
                  IF (layer .eq. (nolyrs-1)) THEN
C
C assumption that no water can pass into the layer containing the phreatic
C surface directly through infiltration
C
                    runoff =runoff+xcess(layer)
                  END IF
                END IF
              ELSE
                IF (layer .lt. (nolyrs-1)) THEN
                  xcess(layer) =xcess(layer-1)
                ELSE
                  runoff =runoff+xcess(layer-1)
                END IF
              END IF
              perd =layer
              layer =layer+1
```

```
            ELSE
              rain =xcess(layer-1)
              xcess(layer-1) =0.
              CALL Redist
              meth =2
              DO WHILE (tmlft .gt. 0.)
                CALL Infiltration_Cap
                IF (fp .gt. rain) THEN
                  CALL Dissolve_Wetting_Front
                ELSE
                  IF (meth .eq. 2) THEN
                    CALL Redist
                  ELSE
                    CALL Fine_Over_Coarse
                  END IF
                END IF
              END DO
            END IF
          END DO
        ELSE
          perd =1
        END IF
C
C make temporary values of moisture content equal to permanent where
C additions have been made
C
        DO j =1,(nolyrs-1)
          IF (temp(j) .gt. wet(j)) THEN
            wet(j) =temp(j)
          END IF
        END DO
C
C for greater intensities where ponding has yet to start and there is not
C an existing wetting front
C
      ELSE IF ((rain .gt. (kf(1)*time)) .and. (meth .eq. 1)) THEN
        fs =(sf(1)*(ts(1)-wet(1)))/(((rain/time)/kf(1))-1)
C
C If the amount of water needed before ponding occurs exceeds the rainfall
C in the timestep, all water is assumed to be infiltrated and no ponding,
C hence no infiltration front occurs within the timestep.
C
        IF (fs .gt. rain) THEN
          tmlft =time
          CALL Dissolve_Wetting_Front(rain)
        ELSE
          tmlft =time-((fs/rain)*time)
          rain =rain-fs
          meth =2
          IF ((ts(1)-wet(1)) .le. 0.) THEN
            wfr =2
            len =0.
          ELSE
            len =fs/(ts(wfr)-wetmc(wfr))
          END IF
C
C for the remaining time calculate the amount of infiltration
C
          DO WHILE (tmlft .gt. 0.)
            CALL Infiltration_Cap
            IF (fp .gt. rain) THEN
              fp =rain
            END IF
            IF (meth .eq. 4) THEN
              CALL Fine_Over_Coarse
            ELSE
              CALL Redist
            END IF
```

```
          END DO
        END IF
C
C If an infiltration wetting front already exists
C
      ELSE IF (meth .gt. 1) THEN
        tmlft =time
        CALL Infiltration_Cap
        IF (fp .gt. rain) THEN
          CALL Dissolve_Wetting_Front(rain)
        ELSE
          DO WHILE (tmlft .gt. 0.)
            CALL Infiltration_Cap
            IF (fp .gt. rain) THEN
              CALL Dissolve_Wetting_Front(rain)
            ELSE
              IF (meth .eq. 2) THEN
                CALL Redist
              ELSE IF (meth .eq. 4) THEN
                CALL Fine_Over_Coarse
              END IF
            END IF
          END DO
        END IF
      END IF
      IF (wfr .gt. perd) THEN
        perd =wfr
      END IF
C
C recalculate temporary layer moisture content values
C
      DO 5 j =1,nolyrs
        IF (j .lt. wfr) THEN
          wetmc(j) =ts(j)
          wetmci(j) =ts(j)
        ELSE IF (j .gt. wfr) THEN
          IF (wetmci(j) .lt. wetmc(j)) THEN
            w =(wetmc(j)-ts(j))/(wetmci(j)-ts(j))
            wetmc(j) =wet(j)
            IF (w .gt. 0.) THEN
              wetmci(j) =(wetmc(j)-ts(j)+(w*ts(j)))/w
            ELSE
              wetmci(j) =wetmc(j)
            END IF
          ELSE
            wetmc(j) =wet(j)
            wetmci(j) =wet(j)
          END IF
        ELSE IF ((j .eq. wfr) .and. (len .le. 0.0000000001)) THEN
          IF (wetmci(j) .lt. wetmc(j)) THEN
            w =(wetmc(j)-ts(j))/(wetmci(j)-ts(j))
            wetmc(j) =wet(j)
            IF (w .gt. 0.) THEN
              wetmci(j) =(wetmc(j)-ts(j)+(w*ts(j)))/w
            ELSE
              wetmci(j) =wetmc(j)
            END IF
          ELSE
            wetmc(j) =wet(j)
            wetmci(j) =wet(j)
          END IF
        END IF
    5 CONTINUE
C
      RETURN
      END
C
C ***************************************************************************
```

```
***
C *
  *
C ********************************************************************************
***
C
      SUBROUTINE Redist
C
C This subroutine decides on whether the wetting front is crossing a layer
C boundary or not and calculates output appropriately
C
C Local Variable Names and Meanings
C lenwf =recalculated length of wetting front after infiltration
C
C Called By:
C SUBROUTINE Infiltration
C SUBROUTINE Dissolve_Wetting_Front
C
C Author: Darryn Evans
C Created: 14th November (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL lenwf
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
      j =wfr
      IF ((ts(wfr)-wet(wfr)) .le. 0.0000000001) THEN
        CALL Boundary
      ELSE
        lenwf =fp/(ts(wfr)-wetmc(wfr))+len
        IF (lenwf .gt. d(wfr)) THEN
          CALL Boundary
        ELSE
          len =len+(fp/(ts(wfr)-wetmc(wfr)))
          runoff =runoff+(rain-fp)
          tmlft =0.
          wet(j) =((ts(j)*len)+(wetmc(j)*(d(j)-len)))/d(j)
        END IF
      END IF
C
      RETURN
      END
C
C ********************************************************************************
***
C *
  *
C ********************************************************************************
***
C
      SUBROUTINE Infiltration_Cap
C
C The purpose of this subroutine is to calculate the infiltration capacity
C of the soil.
C
C Local Variable Names and Meanings:
C sumlen =total length of wetting above the present layer which the wetting
C         front occupies
C sumcon =the sum of each layers depth divided by its conductivity
C
C Called By:
C SUBROUTINE Infiltration
C SUBROUTINE Dissolve_Wetting_Front
C
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
```

```
C Updated: 20th March 1989
C
      REAL sumlen,sumcon,cumulf
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
      cumulf =len*(ts(wfr)-wetmc(wfr))
      IF (wfr .eq. 1) THEN
        fp =(kfin(wfr)*((((ponded+sf(wfr))*(ts(wfr)-wetmc(wfr)))
     1                                  /cumulf)+1))*tmlft
      ELSE IF ((wfr .lt. nolyrs) .and. (wfr .gt. 1)) THEN
        sumlen =0.
        sumcon =0.
        DO 1 i =1,wfr-1
          sumcon =sumcon+(d(i)/kfin(i))
          sumlen =sumlen+d(i)
    1   CONTINUE
        IF (meth .lt. 4) THEN
          fp =(len+sumlen+ponded+sf(wfr))
     1                    /(sumcon+(len/kfin(wfr)))*tmlft
        ELSE
          fp =((sumlen+sf(wfr-1)+ponded)/sumcon)*tmlft
        END IF
      ELSE
        fp =0.
      END IF
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Boundary
C
C The purpose of this subroutine is to continue infiltration across layer
C boundaries, determining the amount of time left in the timestep and the
C amount of water available for infiltration under the new conditions
C
C Called By:
C SUBROUTINE Redist
C SUBROUTINE Fine_Over_Coarse
C
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
      IF (meth .eq. 4) THEN
        fprem =fpleft
      ELSE
        fprem =fp-((d(wfr)-len)*(ts(wfr)-wetmc(wfr)))
      END IF
      wetmc(wfr) =ts(wfr)
      wetmci(wfr) =ts(wfr)
      len =0.
      tmlft =tmlft-(((fp-fprem)/fp)*tmlft)
      rain =rain-(fp-fprem)
      IF (kf(wfr+1) .gt. kf(wfr)) THEN
        meth =4
      ELSE
```

```
        meth =2
      END IF
      wfr =wfr+1
C
      RETURN
      END
C
C ***********************************************************************
***
C *
  *
C ***********************************************************************
***
C
      SUBROUTINE Fine_Over_Coarse
C
C Routine deals with situations when infiltration into a layer is governed by
C the layer of soil above
C
C Local Variable Names and Meanings
C count =local accounting variable
C layer =layer in the soil in which calculations are taking place
C wf =indicates where a fine over coarse boundary exists and is at present
C     delaying the movement down of the wetting front
C
C Called By:
C SUBROUTINE Infiltration
C SUBROUTINE Dissolve_Wetting_Front
C
C Author: Darryn Evans
C Created: 28th April 1989
C Updated:
C
      REAL xcess(50),temp(50)
      INTEGER count,layer,wf
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
      DO 1 j =1,nolyrs
        temp(j) =0.
        xcess(j) =0.
    1 CONTINUE
      wf =wfr
      count =0
      xcess(wfr-1) =fp
      layer =wfr
      temp(layer) =wet(layer)+(xcess(layer-1)/d(layer))
      IF (temp(layer) .gt. ts(layer)) THEN
        xcess(layer) =(temp(layer)-ts(layer))*d(layer)
        temp(layer) =ts(layer)
        layer =layer+1
        fpleft =xcess(layer-1)
        IF (layer .eq. nolyrs) THEN
          runoff =runoff+xcess(layer-1)
          perd =nolyrs-1
          tmlft =0.
        ELSE
          CALL Boundary
        END IF
      ELSE
        runoff =runoff+(rain-fp)
        tmlft =0.
      END IF
C
      DO j =1,(nolyrs-1)
        IF (temp(j) .gt. wet(j)) THEN
          wet(j) =temp(j)
```

```
        END IF
      END DO
C
      RETURN
      END
C
C ***********************************************************************************
***
C *
  *
C ***********************************************************************************
***
C
      SUBROUTINE Dissolve_Wetting_Front(ran)
C
C The purpose of this program is to stop the program working on Green and Amp
t
C principles. This is due to the fact that the supply of water to the surface
C is not great enough to maintain the continuity of the wetting front.
C
C Local Variable Names and Meanings:
C kb =harmonic mean of saturated conductivity at boundary
C
C Called By:
C SUBROUTINE Infiltration
C
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL ran,temp(50),xcess(50),kb
      INTEGER layer,count
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
      DO 1 j =1,nolyrs
        temp(j) =0.
        xcess(j) =0.
    1 CONTINUE
C
      temp(wfr) =wet(wfr)+(ran/d(wfr))
      layer =wfr+1
      IF (temp(wfr) .gt. ts(wfr)) THEN
        xcess(wfr) =(temp(wfr)-ts(wfr))*d(wfr)
        temp(wfr) =ts(wfr)
        IF (layer .eq. nolyrs) THEN
          runoff =runoff+xcess(layer-1)
          perd =nolyrs-1
        ELSE
          DO WHILE ((xcess(layer-1) .gt. 0.)
     1                          .and. (layer .le. (nolyrs-1)))
            wfr =layer
            len =0.
            tmlft =tmlft-(((ran-xcess(layer-1))/ran)*tmlft)
            CALL Infiltration_Cap
            IF (xcess(layer-1) .lt. fp) THEN
              IF (wetmc(wfr) .lt. ts(wfr)) THEN
                IF (xcess(layer-1) .gt. (kf(layer)*tmlft)) THEN
                  wfr =layer
                  len =xcess(layer-1)/(ts(wfr)-wetmc(wfr))
                  meth =2
                ELSE
                  wfr =1
                  len =0.
                END IF
                temp(layer) =wet(layer)+(xcess(layer-1)/d(layer))
                IF (temp(layer) .gt. ts(layer)) THEN
```

```
      DO i =1,noclas
        tmn =tmn+mn(i)
      END DO
      DO 3 j =1,noclas
        md(j) =md(j)+(((REAL(mn(j))*xs)/(REAL(tmn)))*(100./mp(j)))
    3 CONTINUE
      runoff =0.
      ponded =0.
C
C Check to see if any macropores are greater than full, if so add the remaini
ng
C to the other macropores.
C
      CALL Check_Macropore_Heights
C
C Now for each macropore group determine what layer the height of water in
C the macropore reaches into
C
      CALL Calculate_Water_Height
C
C now calculate the transfers from the macropores to the layers
C
      DO 4 j =1,noclas
        CALL Macro_Transfers(mf,mt,ht,j)
        md(j) =ht
    4 CONTINUE
      ponded =0.
      DO j =1,noclas
        IF (md(j) .gt. hd(j)) THEN
          ponded =ponded+((md(j)-hd(j))*(mp(j)/100.))
          md(j) =hd(j)
        END IF
      END DO
C
C Now recalculate the average moisture content for the layer, the increase
C in the size of the wetting fronts. If any water is to be passed back.
C
      DO 5 j =wfr,(nolyrs-1)
        sum =0.
        DO 6 i =1,noclas
          sum =sum+mf(i,j)
    6   CONTINUE
        layer =j
        relf1(j) =(wetmc(j)-wetmci(j))*d(j)
        IF (j .eq. wfr) THEN
          wetmc(j) =wetmc(j)+(sum/(d(j)-len))
        ELSE
          wetmc(j) =wetmc(j)+(sum/d(j))
        END IF
C
C initialise the values of ws,cs and rs to zero
C
        CALL Saturation_Check(mxs,layer)
        relf2(j) =(wetmc(j)-wetmci(j))*d(j)
C
C if the layer is more than saturated then the excess water is passed back
C to the macropores and this is added to the height of the water in the hole
C and minused from that infiltrated from the hole. On occassions where one
C macropore rapidly loses water and one is almost ceased, if the water is
C distributed between the relevant macropores, a minus infiltration appears
C to happen, this is clearly not the case, however water is likely to try
C and reach an equilibrium, so infiltration from the macropore slowly
C draining is negated, and any water not returned to the fast draining hole
C is assumed to have infiltrated into the slower draining.
C
        DO 7 i =1,noclas
          IF ((nmw(i) .le. j) .and. (mn(i) .gt. 0.)) THEN
            mt(i,j) =mt(i,j)-mxs(i)
```

```
C *                          MACROPORE SUBROUTINES
  *
C *
  *
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Macropore
C
C The purpose of this subroutine is to predict macropore flow during
C periods when rainfall exceeds surface infiltration.
C
C Local Variable Names and Meanings:
C mf(j,n) =amount of water infiltrating into the nth layer in terms of depth
C          of the jth class
C mt(j,n) =amount infiltrated from one crack of the jth class taken as height
C          taken as height lost from the water column in the macropore.
C ht =height of water in columns, variable is adjusted as it considers each
C     class
C layer =indicates layer in which calculations are taking place
C mxs(j) =amount of water passed back to the macropores from over
C         saturated layers from the jth class
C xs =sum of water available for macropore flow
C temwet(n) =temporary water content of nth layer
C
C Subroutines Called:
C Check_Macropore_Heights =After adding water from runoff to macropores, each
C                          class of macropore is checked to see its not more
C                          than full, if this is so the excess is added to
C                          other macropore classes.
C Calculate_Water_Height =Calculates the layer in which the top of the water
C                         column is for each macropore class
C Macro_Transfers =calculates amount of water infiltrated from each macropore
C                  class
C Saturation_Check =checks no layers are greater than saturated, if so, water
C                   is passed to macropores impregnating the layer
C
C Called By:
C SUBROUTINE Calculate_Exchanges
C
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL ht,mf(10,50),mt(10,50),mxs(10),temwet(50),depth(50),xs
      REAL relf1(50),relf2(50),r1,r2,ar
      INTEGER layer
      PARAMETER (iout =48)
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
C initialise at the start of each run the calculated exchanges between the
C macropores and the soil to zero
C
      DO 1 j =1,noclas
        DO 2 i =1,nolyrs
          mf(j,i) =0.
          mt(j,i) =0.
    2   CONTINUE
    1 CONTINUE
      tmn =0
C
C Calculate the height of the water in the individual macropores
C
      xs =runoff+ponded
```

```
                  xcess(layer) =(temp(layer)-ts(layer))*d(layer)
                  temp(layer) =ts(layer)
                  rain =xcess(layer)
                  IF (layer .eq. (nolyrs-1)) THEN
C
C assumption that no water can pass into the layer containing the phreatic
C surface directly through infiltration
C
                    runoff =runoff+xcess(layer)
                  END IF
                END IF
              ELSE
                IF (layer .lt. (nolyrs-1)) THEN
                  xcess(layer) =xcess(layer-1)
                ELSE
                  runoff =runoff+xcess(layer-1)
                END IF
              END IF
              perd =layer
              layer =layer+1
            ELSE
              rain =xcess(layer-1)
              xcess(layer-1) =0.
              CALL Redist
              DO WHILE (tmlft .gt. 0.)
                CALL Infiltration_Cap
                IF (fp .gt. rain) THEN
                  fp =rain
                ELSE
                  IF (meth .eq. 2) THEN
                    CALL Redist
                  ELSE
                    CALL Fine_Over_Coarse
                  END IF
                END IF
              END DO
            END IF
          END DO
        END IF
      END IF
C
C Update infiltration parameters
C
      perd =wfr
      wet(wfr) =((ts(wfr)*len)+(wetmc(wfr)*(d(wfr)-len)))/d(wfr)
      wfr =1
      meth =1
      tmlft =0.
      len =0.
C
C if there is macropore flow at the same time calculate the area taken up by
C wetmci, and calculate wetmci and wetmc
C
      DO j =1,(nolyrs-1)
        IF (temp(j) .gt. wet(j)) THEN
          wet(j) =temp(j)
        END IF
      END DO
C
      RETURN
      END
C
C ***********************************************************************
***
C *
  *
C *
  *
```

```
            IF (mt(i,j) .lt. 0.) THEN
              mt(i,j) =0.
            END IF
            mf(i,j) =mf(i,j)-(mxs(i)*(mp(i)/100.))
          END IF
    7   CONTINUE
    5 CONTINUE
C
C Calculate the wetting fronts
C
      DO 8 i =1,nolyrs
        IF (i .eq. wfr) THEN
          depth(i) =d(i)-len
        ELSE
          depth(i) =d(i)
        END IF
        IF ((ts(i)-wetmc(i)) .gt. 0.0000000000001)  THEN
          DO 9 j =1,noclas
            IF ((relf1(i) .gt. 0.) .and. (mwf(j,i) .gt. 0.)) THEN
              ar =relf2(i)/relf1(i)
              r1 =mwf(j,i)+mr(j)
              r2 =mr(j)**2.
              mwf(j,i) =((ar*((r1**2.)-r2)+r2)**0.5)-mr(j)
            ELSE
              area1 =mt(j,i)*(mr(j)**2.)*3.142*(1./(ts(i)-wetmci(i)))
              IF (area1 .gt. 0.) THEN
                r1 =mr(j)+mwf(j,i)
                const =area1/(3.142*d(i))
                mwf(j,i) =((const+(r1**2.))**0.5)-mr(j)
              END IF
            END IF
    9     CONTINUE
        ELSE
C
C the layer is saturated so all effective water fronts have merged so their
C influence on future infiltration is only deemed in terms of moisture defici
t
C
          DO 10 j =1,noclas
            mwf(j,i) =0.
   10     CONTINUE
        END IF
    8 CONTINUE
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Soil_Cracks
C
C Routine calculates the amount of cracking in the surface layer of the soil
C based upon the clay content and the amount of moisture. This is then equate
d
C to circular hole in order to calculate the amount of infiltration into them
.
C
C Local Variable Names and Meanings
C dist =length of cracks in a metre squared in cm units
C width =width of cracks
C area =area of 1 metre square in square centimetres occupied by cracks
C cracs =number of cracks in a metre square
C
```

```
C Called By:
C Program Soil_Atmosphere_Simulation
C Author: Darryn Evans
C Created: 13th April 1989
C Updated:
C
      REAL area,width
      PARAMETER (dist =100.,cracs =8.)
C
      INCLUDE 'main.inc'
C
      cp =(shrgra*moist(1))+shrc
      IF (cp .gt. 0.) THEN
        noclas =permcl+1
        hd(noclas) =d(1)+d(2)
        area =(cp/100.)*10000
        width =(area/(cracs*dist))
        mr(noclas) =width/2.
        mn(noclas) =aint(area/(3.142*(mr(noclas)**2.)))
        mp(noclas) =cp
        msd(noclas) =2
      ELSE
        noclas =permcl
      END IF
C
      RETURN
      END
C
C ***********************************************************************
***
C *
  *
C ***********************************************************************
***
C
      SUBROUTINE Check_Macropore_Heights
C
C The purpose of this subroutine is to check that no macropore becomes filled
,
C if this happens the excess water is passed into the macropores yet to be
C filled or if all the macropores are full are passed to the variable pond wh
ich
C at present has no defined purpose.
C
C Local Variable Names and Meanings
C excess =describes the amount of water leftover if the macropore class is fu
ll
C full(n) =indicates if macropore of nth class is full
C tempmn =number of macropores that are still not full
C neg =indicates that all water has been distributed if true
C
C Called By:
C SUBROUTINE Macropore
C
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C Revised: 21st October 1989
C
      REAL excess
      INTEGER full(50),tempmn,count,kount,tmn
      LOGICAL neg
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
      DATA nout / 68 /
C
```

```
      DO j =1,noclas
        full(j) =0
      END DO
      neg =.false.
      DO WHILE (neg .eq. .false.)
        count =1
        kount =0
        tempmn =0
        DO WHILE (count .le. noclas)
          j =count
          IF (md(j) .gt. hd(j)) THEN
            excess =(md(j)-hd(j))*(mp(j)/100.)
            md(j) =hd(j)
            full(j) =1
            count =noclas+1
C
C determine the number of macropores not full
C
            DO i =1,noclas
              IF (full(i) .lt. 1) THEN
                tempmn =tempmn+mn(i)
              END IF
            END DO
C
C add water to empty macropores, if none left add to ponded store
C
            IF (tempmn .gt. 0) THEN
              DO i =1,noclas
                IF (full(i) .lt. 1) THEN
                  md(i) =md(i)+((mn(i)*excess)/tempmn)*
     1                                              (100./mp(i))
                END IF
              END DO
            ELSE
              ponded =ponded+excess
              neg =.true.
            END IF
          ELSE
            kount =kount+1
            count =count+1
            IF (kount .eq. noclas) THEN
C
C all water has been distributed
C
              neg =.true.
            END IF
          END IF
        END DO
      END DO
C
C if there is any ponded water remaining on the surface distribute this evenl
y
C between all macropore classes, however keep knowledge of the height of
C ponding to add to calculations for macropore infiltration.
C
      IF (ponded .gt. 0.) THEN
        tmn =0
        DO i =1,noclas
          tmn =tmn+mn(i)
        END DO
        DO j =1,noclas
          md(j) =md(j)+(((real(mn(j))*ponded)/(real(tmn)))*(100.
     1                                                    /mp(j)))
        END DO
      END IF
C
      RETURN
      END
```

```
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Calculate_Water_Height
C
C The purpose of this subroutine is to calculate which layer the top of the
C water column is in for each class of macropore
C
C Called By:
C SUBROUTINE Macropore
C
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL depth
      INTEGER count
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
      DO j =1,noclas
        IF (md(j) .gt. hd(j)) THEN
          nmw(j) =1
        ELSE
          depth =0.
          count =msd(j)
          IF (md(j) .gt. depth) THEN
            DO WHILE (md(j) .gt. depth)
              depth =depth+d(count)
              IF (depth .ge. md(j)) THEN
                nmw(j) =count
              ELSE
                count =count-1
              END IF
            END DO
          ELSE
            nmw(j) =msd(j)
          END IF
        END IF
      END DO
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Macro_Transfers(mf,t,height,cl)
C
C The purpose of this subroutine is to calculate the amount of infiltration
C into each layer from each macropore class
C
C Local Variable Names and Meanings:
C mf(j,n) =amount of water infiltrating into nth layer in terms of depth and
C          the jth class
C t(j,n) =amount of water infiltrating into nth layer in terms of height lost
C        from macropore of class j
C height =height of water in macropore after calculation
```

```
C z =used to keep account of the height of the water column above the base of

C     the present layer
C sum =keeps account of the of the amount of infiltration for each layer
C suml =keeps account of the drop in height of the water column for each loss
C       of water to each layer
C max =maximum allowable infiltration for each layer i.e the amount of water
C      above the base of the layer at the start of timestep
C empt =length above the top of the water column and the top of the layer
C dist =distance over which there is no infiltration in the column due to
C       prescence of a wetting front
C cl =macropore class
C
C Called By:
C SUBROUTINE Macropore
C
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL sum,suml,max,empt,dist,mf(10,50),t(10,50),height,z,zl,xcess
      INTEGER kount,cl
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
C initialize calculation values
C
      DO i =1,nolyrs
        t(cl,i) =0.
        mf(cl,i) =0.
      END DO
      IF (md(cl) .gt. hd(cl)) THEN
        z =hd(cl)
        xcess =md(cl)-hd(cl)
      ELSE
        z =md(cl)
        xcess =0.
      END IF
      sum =0.
      DO 20 i =msd(cl),nmw(cl),-1
        zl =z+ponded
        IF ((i .eq. wfr) .and. (wet(i) .lt. ts(i))) THEN
          IF (i .eq. nmw(cl)) THEN
            empt =d(i)-z
            IF (empt .gt. len) THEN
              dist =z
            ELSE
              dist =d(i)-len
            END IF
            IF (mwf(cl,i) .gt. 0.) THEN
              t(cl,i) =(((kf(i)*mr(cl)*((zl/2.)+sf(i)))/((mwf(cl,i)**2
     1                    .)+(mr(cl)*mwf(cl,i))))*(dist))*time
            ELSE
              t(cl,i) =z+xcess
            END IF
          ELSE
            IF (mwf(cl,i) .gt. 0.) THEN
              t(cl,i) =(((kf(i)*mr(cl)*((zl-(d(i)/2.))+sf(i)))/(
     1           (mwf(cl,i)**2.)+(mr(cl)*mwf(cl,i))))*(d(i)-len))*time
            ELSE
              t(cl,i) =z+xcess
            END IF
          END IF
        ELSE IF ((wet(i) .lt. ts(i)) .and. (i .gt. wfr)) THEN
          IF (i .eq. nmw(cl)) THEN
            IF (mwf(cl,i) .gt. 0.) THEN
              t(cl,i) =(((kf(i)*mr(cl)*((z/2.+(zl-z))+sf(i)))/
```

```
     1                    ((mwf(cl,i)**2.)+(mr(cl)*mwf(cl,i))))*z)*time
            ELSE
              t(cl,i) =z+xcess
            END IF
          ELSE
            IF (mwf(cl,i) .gt. 0.) THEN
              t(cl,i) =(((kf(i)*mr(cl)*((z1-(d(i)/2.))+sf(i)))/((mwf
     1                  (cl,i)**2.)+(mr(cl)*mwf(cl,i))))*d(i))*time
            ELSE
              t(cl,i) =z+xcess
            END IF
          END IF
        ELSE
          t(cl,i) =0.
        END IF
        mf(cl,i) =t(cl,i)*(mp(cl)/100.)
        sum =sum+t(cl,i)
        IF (i .gt. nmw(cl)) THEN
          z =z-d(i)
        END IF
   20 CONTINUE
      max =md(cl)
C
C This part of the subroutine checks that no one layer can infiltrate more
C water than is already available in the macropore. It starts the check at
C the base layer and proceeds upwards, once a layer is identified as
C producing too much flow, infiltration is recalculated for this layer and th
e
C above layers based on the percentages of the original calculations.
C
      kount =msd(cl)
      DO WHILE (kount .ge. nmw(cl))
        IF (max .gt. 0.) THEN
          IF (t(cl,kount) .gt. max) THEN
            t(cl,kount) =max
            mf(cl,kount) =max*(mp(cl)/100.)
            max =0.
          ELSE
            IF (kount .eq. nmw(cl)) THEN
              max =max-t(cl,kount)
            ELSE
              IF (t(cl,kount) .lt. d(kount)) THEN
                max =max-d(kount)
              ELSE
                max =max-t(cl,kount)
              END IF
              IF (max .lt. 0.) THEN
                max =0.
              END IF
            END IF
          END IF
        ELSE
          t(cl,kount) =0.
          mf(cl,kount) =0.
        END IF
        kount =kount-1
      END DO
      height =md(cl)
      DO j =msd(cl),nmw(cl),-1
        height =height-t(cl,j)
      END DO
C
      RETURN
      END
C
C ***********************************************************************
***
C *
```

```
  *
C ***************************************************************************
***
C
      SUBROUTINE Saturation_Check(ms,lyr)
C
C The purpose of this routine, is if any layer becomes greater than saturated
C by macropore inflow, the resulting excess is added back to the macropores
C that penetrate into it
C
C Local Variable Names and Meanings:
C ms(j) =amount of water passed back to the jth macropore classes from
C        an oversaturated layer
C lyr =layer in which the calculations are taking place
C pond =if all macropores are full this variable holds the excess water
C excess =the amount the layer is oversaturated by
C
C Called By:
C SUBROUTINE Macropore
C
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL ms(10),excess
      INTEGER lyr,extend(10),tempmn,full(50)
      LOGICAL neg
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
      DO j =1,noclas
        ms(j) =0.
        extend(j) =0
      END DO
      tempmn =0
      IF (wetmc(lyr) .gt. ts(lyr)) THEN
        IF (lyr .eq. wfr) THEN
          excess =(wetmc(lyr)-ts(lyr))*(d(lyr)-len)
        ELSE
          excess =(wetmc(lyr)-ts(lyr))*d(lyr)
        END IF
        wetmc(lyr) =ts(lyr)
        DO j =1,noclas
          IF (msd(j) .ge. lyr) THEN
            extend(j) =1
            tempmn =tempmn+mn(j)
          END IF
        END DO
        DO j =1,noclas
          IF (extend(j) .eq. 1) THEN
            ms(j) =(((REAL(mn(j))*excess)/(REAL(tempmn)))*(100./mp(j)))
            md(j) =md(j)+ms(j)
          END IF
        END DO
C
C Check to see if any macropores are greater than full, if so add the remaini
ng
C to the other macropores, and adjust for ws,cs and rs.
C
        DO j =1,noclas
          full(j) =0
        END DO
        neg =.false.
        DO WHILE (neg .eq. .false.)
          count =1
          kount =0
          tempmn =0
```

```
          DO WHILE (count .le. noclas)
            j =count
            IF (md(j) .gt. hd(j)) THEN
              excess =(md(j)-hd(j))*(mp(j)/100.)
              md(j) =hd(j)
              ms(j) =ms(j)-(md(j)-hd(j))
              full(j) =1
              count =noclas+1
C
C determine the number of macropores not full
C
              DO i =1,noclas
                IF (full(i) .lt. 1) THEN
                  tempmn =tempmn+mn(i)
                END IF
              END DO
C
C add water to empty macropores, if none left add to ponded store
C
              IF (tempmn .gt. 0) THEN
                DO i =1,noclas
                  IF (full(i) .lt. 1) THEN
                    md(i) =md(i)+((mn(i)*excess)/tempmn)*
     1                                                (100./mp(i))
                    IF (ms(i) .gt. 0.) THEN
                      ms(i) =ms(i)+((mn(i)*excess)/tempmn)*(100./mp(i))
                    END IF
                  END IF
                END DO
              ELSE
                ponded =ponded+excess
                neg =.true.
              END IF
            ELSE
              kount =kount+1
              count =count+1
              IF (kount .eq. noclas) THEN
C
C all water has been distributed
C
                neg =.true.
              END IF
            END IF
          END DO
        END DO
      END IF
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C *
  *
C *                          PERCOLATION SUBROUTINES
  *
C *
  *
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Redist_Percolation(toomuc)
C
C The purpose of this routine is to calculate percolation whilst wetting fron
```

```
ts
C persist within the profile
C
C Local Variable Names and Meanings:
C headav(n),headin(n) =the suction value for the nth layer given by mc(n) and
C                      mci(n) respectively
C condav(n),condin(n) =as previous but for conductivity
C kbav(n),kbin(n) =conductivtiy across the boundary as calculated by the
C                  harmonic mean for mc(n) and mci(n) respectively
C v12av,v12in =exchange acroos the nth boundary for mc(n) and mci(n)
C exch(n) =actual exchange across the nth boundary
C tc =revised amount of mci(n) after exchange
C macper(n) =percentage contribution of each classes wetting fronts to
C            percolation
C depth(n) =depth of nth layer, used to adjust the length of the layer
C           cdontaining the wetting front
C satflo =amount of water lost from the mmacropores wetting fronts
C excess =amount of water in layer in excess of saturation
C toomuc =logical variable deciding on whether too much water has been passed
C gain =amount of water gained by the layer containing the groundwater layer
C macsum =indicates if any of the percolation from the above layer has come d
own
C          through diminished macropores.
C
C Subroutines Called:
C Suction_Cal =calculates the suction for each layer
C Conductivity_Cal =calculates the conductivity for each layer
C Exchange_Cal =calculates the exchange across the boundary
C Recal_Fronts =calculates the contribution from the macropores wetting front
s
C                to percolation
C Give_Back =if underlying layer becomes too saturated due to too much
C            percolation routine calculates how much of the excess is
C            distributed back to the macropores wetting fronts and the area
C            outside, based on original contribution
C
C
C Called By:
C SUBROUTINE Calculate_Exchanges
C
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL headav(50),headin(50),v12av(50),v12in(50),condav(50)
      REAL condin(50),depth(50),kbav(50),kbin(50),exch(50),tc
      REAL macper(10,50),satflo,excess,gain,macsum,mdsum,head1(50)
      INTEGER count
      LOGICAL toomuc
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
C Calculate suctions and conductivities for each layer
C
      mdsum =0.
      DO 1 j =perd,nolyrs
        DO i =1,noclas
          mdsum =mdsum+mwf(i,j)
        END DO
        IF (mdsum .gt. 0.) THEN
          CALL Suction_Cal(wetmc(j),ts(j),j)
          headav(j) =head(j)
          CALL Suction_Cal(wetmci(j),ts(j),j)
          headin(j) =head(j)
          CALL Conductivity_Cal(wetmc(j),ts(j),j)
          condav(j) =conduc(j)
          CALL Conductivity_Cal(wetmci(j),ts(j),j)
          condin(j) =conduc(j)
```

```
        ELSE
          CALL Suction_Cal(wetmc(j),ts(j),j)
          headav(j) =head(j)
          headin(j) =headav(j)
          CALL Conductivity_Cal(wetmc(j),ts(j),j)
          condav(j) =conduc(j)
          condin(j) =condav(j)
        END IF
C
C Calculate the depth of each layer, in the layer containing the wetting
C front only the depth which is unaffected by it, is considered for drainage
C
        IF (j .eq. wfr) THEN
          depth(j) =d(j)-len
        ELSE
          depth(j) =d(j)
        END IF
    1 CONTINUE
C
C Work out the average conductivity for each layer
C
      DO 2 j =perd,(nolyrs-1)
        IF (j .eq. (nolyrs-1)) THEN
          kbav(j) =condav(j)
          kbin(j) =condin(j)
        ELSE
          kbav(j) =((depth(j)+depth(j+1))/((depth(j)/condav(j))+(depth
     1                    (j+1)/condin(j+1))))
          kbin(j) =((depth(j)+depth(j+1))/((depth(j)/condin(j))+(depth
     1                    (j+1)/condin(j+1))))
        END IF
    2 CONTINUE
C
C Calculate the exchange of water between layers
C
      DO 3 j =perd,(nolyrs-1)
        IF ((wetmc(j) .gt. wetmci(j)) .or. (wetmc(j+1)
     1                                  .gt. wetmci(j+1))) THEN
          head1(j) =headav(j)
          head1(j+1) =headin(j+1)
          CALL Exchange_Cal(v12av(j),kbav,head1,depth,time,j)
          CALL Exchange_Cal(v12in(j),kbin,headin,depth,time,j)
          IF ((v12av(j) .gt. 0.) .and. (v12in(j) .gt. 0.)) THEN
            IF (v12in(j) .gt. v12av(j)) THEN
              exch(j) =v12in(j)
            ELSE
              satflo =v12av(j)-v12in(j)
C
C Recalculate the wetting fronts according to the contribution of them
C to the percolation total
C
              CALL Recal_Fronts(tc,wetmci(j),depth,j,
     1                                                  v12in(j),satflo)
              v12in(j) =tc
              exch(j) =v12av(j)
            END IF
          ELSE IF (v12av(j) .gt. 0.) THEN
            CALL Recal_Fronts(tc,wetmci(j),depth,j,
     1                                                  v12in(j),v12av(j))
            v12in(j) =tc
            exch(j) =v12av(j)
          ELSE IF (v12in(j) .gt. 0.) THEN
            exch(j) =v12in(j)
          ELSE
            v12in(j) =0.
            exch(j) =0.
          END IF
        ELSE
```

```
          CALL Exchange_Cal(v12av(j),kbav,headav,depth,time,j)
          v12in(j) =v12av(j)
          exch(j) =v12av(j)
        END IF
    3 CONTINUE
C
C Test to see if any of the layers are passing too much water in the present
C timestep, if this is so pass back to the calling routine to enable the
C reduction of the timestep, unless this is already the smallest one.
C
      toomuc =.false.
      count =perd
      DO WHILE ((toomuc .eq. .false.) .and. (count .le. (nolyrs-1)))
        j =count
        IF (((exch(j)/depth(j)) .gt. (0.1*wetmc(j))) .and. (time .gt.
     1                                          0.1)) THEN
          toomuc =.true.
        END IF
        count =count+1
      END DO
C
      IF (toomuc .eq. .true.) THEN
        RETURN
      ELSE
C
C Continue with the program. Apply moisture changes to the layers
C
        DO 4 j =perd,nolyrs
          IF (j .eq. perd) THEN
            wetmc(j) =wetmc(j)-(exch(j)/depth(j))
            wetmci(j) =wetmci(j)-(v12in(j)/depth(j))
          ELSE IF (j .eq. nolyrs) THEN
            gain =exch(j-1)
          ELSE
            wetmc(j) =wetmc(j)+((exch(j-1)-exch(j))/depth(j))
            wetmci(j) =wetmci(j)+((exch(j-1)-v12in(j))/depth(j))
          END IF
    4   CONTINUE
C
C estimate the groundwater flow from Husseins assumptions
C
        CALL Groundwater_Discharge(gain)
C
C Check to ensure no layer becomes more than saturated
C
        DO 5 j =nolyrs,(perd+1),-1
          excess =0.
          IF (wetmc(j) .gt. ts(j)) THEN
            excess =wetmc(j)-ts(j)
            wetmc(j) =ts(j)
            wetmci(j) =ts(j)
            wetmci(j-1) =wetmci(j-1)+(excess*(depth(j)/depth(j-1)))
            wetmc(j-1) =wetmc(j-1)+(excess*(depth(j)/depth(j-1)))
          END IF
    5   CONTINUE
      END IF
C
C Update the temporary average moisture content of the layer
C
      DO 6 j =1,nolyrs
        IF (j .lt. perd) THEN
          wet(j) =ts(j)
        ELSE IF (j .eq. perd) THEN
          wet(j) =((ts(j)*len)+(wetmc(j)*depth(j)))/d(j)
        ELSE
          wet(j) =wetmc(j)
        END IF
    6 CONTINUE
```

```
C
      RETURN
      END
C
C ***********************************************************************************
***
C *
  *
C ***********************************************************************************
***
C
      SUBROUTINE Percolation(toomuc)
C
C This routine is used within the program for periods when ther is no macropo
re
C or infiltration wetting fronts
C
C Local Variable Names and Meanings:
C kb(n) =harmonically averaged mean across the nth boundary
C exch(n) =exchange of water across the nth boundary
C v12 =exchange of water across boundary
C excess =water above saturation which is passed back up
C toomuc =logical variable used to decide if too much water has been passed
C          across the boundary
C gain =amount of water gained by the layer containing the groundwater table
C
C Called By:
C SUBROUTINE Calculate_Exchanges
C
C Author: Darryn Evans
C Created: 17th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL kb(50),exch(50),v12,excess
      INTEGER count
      LOGICAL toomuc
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
C Calculate suction and conductivity for each layer
C
      DO 1 j =1,nolyrs
        CALL Suction_Cal(wet(j),ts(j),j)
        CALL Conductivity_Cal(wet(j),ts(j),j)
    1 CONTINUE
C
C work out the average conductivity for each layer
C
      DO 2 j =1,(nolyrs-1)
        IF (j .eq. (nolyrs-1)) THEN
          kb(j) =conduc(j)
        ELSE
          kb(j) =((d(j)+d(j+1))/((d(j)/conduc(j))+(d(j+1)
     1                                        /conduc(j+1))))
        END IF
    2 CONTINUE
C
C Calculate exchanges for each layer
C
      DO 3 j =1,(nolyrs-1)
        CALL Exchange_Cal(v12,kb,head,d,time,j)
        exch(j) =v12
    3 CONTINUE
C
C Test to see if any of the exchanges across boundaries are greater than that
C which is allowed, if this is so pass back to calling routine to reduce the
C timestep, unless the smallest timestep is already in use
```

```
C
      toomuc =.false.
      count =1
      DO WHILE ((toomuc .eq. .false.) .and. (count .le. (nolyrs-1)))
        j =count
        IF (((exch(j)/d(j)) .gt. (0.1*wet(j))) .and. (time .gt. 0.1))
     1                                                              THEN
          toomuc =.true.
        END IF
        count =count+1
      END DO
C
      IF (toomuc .eq. .true.) THEN
        RETURN
      ELSE
C
C Apply moisture changes to the layer
C
        DO 4 j =1,nolyrs
          IF (j .eq. 1) THEN
            wet(j) =wet(j)-(exch(j)/d(j))
          ELSE IF (j .eq. nolyrs) THEN
            gain =exch(j-1)
          ELSE
            wet(j) =wet(j)+((exch(j-1)-exch(j))/d(j))
          END IF
    4   CONTINUE
C
C adjust the groundwater table according to Husseins Predictions
C
        CALL Groundwater_Discharge(gain)
C
C Check to ensure that no layers become greater than saturated
C
        DO 5 j =nolyrs,1,-1
          excess =0.
          IF (wet(j) .gt. ts(j)) THEN
            excess =wet(j)-ts(j)
            wet(j) =ts(j)
            IF (j .eq. 1) THEN
              ponded =ponded+excess
            ELSE
              wet(j-1) =wet(j-1)+(excess*(d(j)/d(j-1)))
            END IF
          END IF
    5   CONTINUE
      END IF
C
      DO 6 j =1,nolyrs
        wetmc(j) =wet(j)
        wetmci(j) =wet(j)
    6 CONTINUE
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Suction_Cal(vmc,sat,lyr)
C
C The purpose of this subroutine is to calculate the suction in each layer
C depending on the moisture content. The equation used is that of Genuchten
C (1980) with the parameter values being determined by statistical means.
```

```
C
C Local Variable Names and Meanings
C lyr =layer of profile for which calculations are taking place
C vmc =moisture content of layer
C sat =saturated moisture content of layer
C m =Genuchten(1980) equation parameter
C relmc =inverse of relative moisture content
C
C Called By:
C PROGRAM Soil_Atmosphere_Simulation
C SUBROUTINE Redist_Percolation
C SUBROUTINE Percolation
C SUBROUTINE Capillary_Rise
C
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL vmc,sat,m,relmc
      INTEGER lyr
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
      IF (vmc .gt. sat) THEN
        head(lyr) =0.
      ELSE
        IF (vmc .le. tr(lyr)) THEN
          head(lyr) =0.3E+06
        ELSE
          relmc =(sat-tr(lyr))/(vmc-tr(lyr))
          m =1-(1/nl(lyr))
          head(lyr) =(((relmc**(1/m))-1)**(1/nl(lyr)))/alpha(lyr)
        END IF
      END IF
C
      RETURN
      END
C
C ***********************************************************************
***
C *
  *
C ***********************************************************************
***
C
      SUBROUTINE Conductivity_Cal(vmc,sat,lyr)
C
C The subroutine calculates the value of conductivity using Genuchtens (1980)
C relationships between the suction and conductivity curves, and the estimate
d
C values of ks.
C
C Local Variable Names and Meanings:
C lyr =layer of profile for which calculations are taking part
C vmc =moisture content of layer
C sat =saturated moisture content of a layer
C m =Genuchten(1980) equation parameter
C relmc =inverse of relative moisture content for layer
C rl =relative moisture content
C kr =relative conductivity for layer
C
C Called By:
C PROGRAM Soil_Atmosphere_Simulation
C SUBROUTINE Redist_Percolation
C SUBROUTINE Percolation
C SUBROUTINE Capillary_Rise
C
```

```
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL vmc,sat,m,relmc,rl,kr
      INTEGER lyr
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
      IF (vmc .ge. sat) THEN
        conduc(lyr) =ks(lyr)
      ELSE
        IF (vmc .le. tr(lyr)) THEN
          kr =0.1E-20
        ELSE
          relmc =(sat-tr(lyr))/(vmc-tr(lyr))
          rl =1./relmc
          m =1-(1/nl(lyr))
          kr =(rl**0.5)*((1-((1-(rl**(1/m)))**m))**2)
          IF (kr .lt. (0.1E-20)) THEN
            kr =0.1E-20
          END IF
        END IF
        conduc(lyr) =kr*ks(lyr)
      END IF
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Exchange_Cal(v12,k,h,deph,t,lyr)
C
C The purpose of this routine is to calculate the exchange of water between
C layers using a much simplified analytical form of the Richards Equation.
C
C Local Variable Names and Meanings:
C lyr =layer of profile for which calculations are taking part
C v12 =exchange of water across boundary
C k =conductivtiy of layer
C h =head of layer
C deph =depth of layer
C t =length of timestep
C
C Called By:
C SUBROUTINE Redist_Percolation
C SUBROUTINE Percolation
C
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL v12,k(50),h(50),deph(50),t
      INTEGER lyr
C
      INCLUDE 'main.inc'
C
      IF (lyr .eq. (nolyrs-1)) THEN
        v12 =(k(lyr)*(((-h(lyr))/((deph(lyr)/2.)+dwt))+1))*t
      ELSE
        v12 =(k(lyr)*(((h(lyr+1)-h(lyr))/((deph(lyr)+deph(lyr+1))
     1                                                    /2))+1))*t
```

```
      END IF
      IF (v12 .lt. 0.) THEN
        v12 =0.
      END IF
C
      RETURN
      END

C
C ***********************************************************************
***
C *
  *
C ***********************************************************************
***
C
      SUBROUTINE Recal_Fronts(tci,tint,deth,lyr,v12i,flo)
C
C The purpose of this routine is to recalculate the wetting fronts according
C to how much water is lost from them by percolation
C
C Local Variable Names and Meanings:
C lyr =layer of profile for which calculations are taken part
C tci =calculated value of mci(n)
C tint  =pre-calculated value of mci(n)
C deth(n) =depth of layer
C v12i =exchange across boundary resulting from mci(n)
C flo =difference between transfers resulting from mc(n) and mci(n). Only
C       considered if mc(n) is greater than mci(n)
C xs =difference between flo and the amount of water within the wetting front
s
C bl =amount of macropore area left after infiltration
C
C Called By:
C SUBROUTINE Redist_Percolation
C
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL tci,tint,deth(50),v12i,flo,xs,md1,b1
      REAL macbac,tmn,r1,const,areal
      INTEGER lyr
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
      md1 =ts(lyr)-tint
      DO j =1,noclas
        f(lyr) =(wetmc(lyr)-wetmci(lyr))*deth(lyr)
      END DO
      IF ((f(lyr) .lt. flo) .or. (f(lyr) .le. 0.)) THEN
        DO j =1,noclas
          mwf(j,lyr) =0.
        END DO
        xs =flo-f(lyr)
        tci =v12i+xs
      ELSE
        ar =(1-(flo/f(lyr)))
        DO j =1,noclas
          IF (mwf(j,lyr) .gt. 0.) THEN
            r1 =mwf(j,lyr)+mr(j)
            r2 =mr(j)**2.
            mwf(j,lyr) =((ar*((r1**2.)-r2)+r2)**0.5)-mr(j)
          END IF
        END DO
        tci =v12i
      END IF
```

```
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Groundwater_Discharge(xtra)
C
C The purpose of this algorithm is to calculate the groundwater discharge and
C then recalculate the height of the water table in the layer
C
C Local Variable Names and Meanings
C xtra =amount of water input into the groundwater layer
C tfc =moisture content at field capacity (for now Husseins value)
C qgw =groundwater drainage
C
C Called By:
C SUBROUTINE Redist_Percolation
C SUBROUTINE Percolation
C
C Author: Darryn Evans
C Created: 4th March 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL xtra,qgw,a
      PARAMETER (tfc =0.14)
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
C calculate groundwater drainage, convert cd for work with cm units
C
      d4 =d(nolyrs)
      qgw =((coefd/86400.)*(d4-temdwt))*time
C
C calculate the new moisture content
C
      wetmc(nolyrs) =wetmc(nolyrs)+(xtra/d4)-(qgw/d4)
      IF (wetmc(nolyrs) .gt. ts(nolyrs)) THEN
        wetmc(nolyrs) =ts(nolyrs)
      END IF
      wetmci(nolyrs) =wetmc(nolyrs)
      wet(nolyrs) =wetmc(nolyrs)
C
C calculate the new height of the water table
C
      a =(xtra-qgw)/(ts(nolyrs)-tfc)
      temdwt =temdwt-a
      IF (temdwt .lt. 0.) THEN
        qgw =qgw+(ABS(temdwt)/(ts(nolyrs)-tfc))
        temdwt =0.
      END IF
      sumqgw =sumqgw+qgw
C
      RETURN
      END
C
C ***************************************************************************
C *
C ***************************************************************************
C
      SUBROUTINE Calculate_Rise
C
```

```
C This routine determines how the capillary rise is calculated by choosing
C the optimum timestep
C
C Local Variable Names and Meanings:
C toomuc =indicates whether too much water has been allowed into the layer
C          in one timestep. If true then timestep is reduced.
C tme(n) =the nth timestep value
C sumcap(j) =sum of capillary-rise for the day across the jth boundary
C smcr(j) =passes amount of capillary-rise for each timestep back
C
C Called By:
C PROGRAM Soil_Atmosphere_Simulation
C
C Author: Darryn Evans
C Created: 15/01/90
C Updated:
C
      REAL tme(4),sumcap(50),smcr(50)
      LOGICAL toomuc
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
      DATA tme(1),tme(2),tme(3),tme(4) / 86400.,1440.,60.,1. /
C
      DO j =1,nolyrs-1
        sumcap(j) =0.
      END DO
      toomuc =.true.
      DO WHILE (toomuc .eq. .true.)
        CALL Capillary_Rise(tme(1),toomuc,smcr)
        IF (toomuc .eq. .true.) THEN
          DO j =1,24
            CALL Capillary_Rise(tme(2),toomuc,smcr)
            IF (toomuc .eq. .true.) THEN
              DO ij =1,60
                CALL Capillary_Rise(tme(3),toomuc,smcr)
                IF (toomuc .eq. .true.) THEN
                  DO i =1,60
                    CALL Capillary_Rise(tme(4),toomuc,smcr)
                    DO ji =1,nolyrs
                      moist(ji) =wet(ji)
                      sumcap(ji) =sumcap(ji)+smcr(ji)
                    END DO
                  END DO
                ELSE
                  DO i =1,nolyrs
                    moist(i) =wet(i)
                    sumcap(i) =sumcap(i)+smcr(i)
                  END DO
                END IF
              END DO
            ELSE
              DO ij =1,nolyrs
                moist(ij) =wet(ij)
                sumcap(ij) =sumcap(ij)+smcr(ij)
              END DO
            END IF
          END DO
        ELSE
          DO j =1,nolyrs
            moist(j) =wet(j)
            sumcap(j) =sumcap(j)+smcr(j)
          END DO
        END IF
      END DO
C
      WRITE(77,120)ndays,sumcap(1),sumcap(2),sumcap(3)
```

```
C
  120 FORMAT(1x,i3,3(4x,e16.8))
C
      RETURN
      END
C
C ***********************************************************************
***
C *
  *
C ***********************************************************************
***
C
      SUBROUTINE Capillary_Rise(tim,muc,smcap)
C
C The routine calculates, on a daily basis, the upward flow of water from one

C layer to the next. One restriction is imposed: if the head is less than 1/3
C bar (c. 330cm) -nominally quoted as field capacity- there is no upward flow
C of water into it.
C
C Local Variables Names and Meanings:
C kb(j) =average conductivity across the jth layer boundary
C exch(j) =exchange of water across the jth boundary
C tfc =nominal value chosen as field capacity for layer 4
C smcap(j) =exchange across the jth boundary
C
C Called By:
C PROGRAM Soil_Atmosphere_Simulation
C
C Author: Darryn Evans
C Created: 12/03/89
C Updated: 20th March 1989
C
      REAL kb(50),exch(50),tfc,tim,smcap(50)
      LOGICAL muc
C
      INCLUDE 'main.inc'
      INCLUDE 'temp.inc'
C
      DO j =1,nolyrs
        wet(j) =moist(j)
        CALL Suction_Cal(wet(j),ts(j),j)
        CALL Conductivity_Cal(wet(j),ts(j),j)
      END DO
      tfc =0.14
C
C Average out the conductivity between layers
C
      DO 1 j =1,(nolyrs-1)
        IF (j .eq. (nolyrs-1)) THEN
          kb(j) =conduc(j)*tim
        ELSE
          kb(j) =((d(j)+d(j+1))/((d(j)/(conduc(j)*tim))+(d(j+1)/
     1          (conduc(j+1)*tim))))
        END IF
    1 CONTINUE
C
C Calculate the exchanges between the layers
C
      DO 2 j =1,(nolyrs-1)
        IF (j .eq. (nolyrs-1)) THEN
          exch(j) =-(kb(j)*(((1-head(j))/((d(j)/2.)+dwt))+1))
        ELSE
          exch(j) =-(kb(j)*(((head(j+1)-head(j))/((d(j)+d(j+1))
     1          /2.))+1.))
        END IF
        IF (exch(j) .lt. 0.) THEN
```

```
          exch(j) =0.
        END IF
    2 CONTINUE
C
C check to ensure changes are not too large in all but the smallest
C timestep
C
      muc =.false.
      jp =nolyrs-1
      DO WHILE ((muc .eq. .false.) .and. (jp .ge. 1))
        IF (((exch(jp)/d(jp)) .gt. (0.1*wet(jp))) .and.
     1            (tim .gt. 1)) THEN
          muc =.true.
        END IF
        jp =jp-1
      END DO
C
      IF (muc .eq. .true.) THEN
        RETURN
      END IF
C
C apply exchanges across the boundaries and recalculate DWT
C
      DO 4 j =1,nolyrs
        IF (j .eq. 1) THEN
          wet(j) =wet(j)+(exch(j)/d(j))
        ELSE IF (j .eq. nolyrs) THEN
          wet(j) =wet(j)-(exch(j-1)/d(j))
          dwt =dwt+(exch(j-1)/(ts(j)-tfc))
        ELSE
          wet(j) =wet(j)+((exch(j)-exch(j-1))/d(j))
        END IF
    4 CONTINUE
C
C Check no layers become greater than saturated, no need to check 4 as only
C limited amount of water excess water already removed from the system
C
      DO 5 j =1,(nolyrs-1)
        IF (wet(j) .gt. ts(j)) THEN
          xcess =wet(j)-ts(j)
          wet(j) =ts(j)
          wet(j+1) =wet(j+1)+(xcess/d(j+1))
          IF (j .eq. (nolyrs-1)) THEN
            dwt =dwt-(exch(j)/(ts(j+1)-tfc))
          END IF
        END IF
    5 CONTINUE
C
  100 FORMAT(1x//)
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C *
  *
C *                      EVAPOTRANSPIRATION SUBROUTINES
  *
C *
  *
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Evap_Cal
```

```
C
C The purpose of this subroutine is to split the evaporative demand between
C transpiration and bare soil evaporation.
C
C Called By:
C PROGRAM Soil_Atmosphere_Simulation
C
C Author: Darryn Evans
C Created: 4th March 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      INCLUDE 'main.inc'
C
      esoil =evapor*(1-lai)-eint
      IF (esoil .lt. 0.) THEN
        esoil =0.
      END IF
      eplant =evapor-esoil
      IF (eplant .lt. 0.) THEN
        eplant = 0.
      END IF
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Bare_Soil_Evap(ra)
C
C The subroutine calculates the evaporation from the bare soil taking into
C account the available moisture in the surface layer
C
C Local Variable Names and Meanings:
C beta =available moisture in the top layer of the soil
C eps =potential bare soil evaporation
C es =actual bare soil evaporation
C
C Called By:
C PROGRAM Soil_Atmosphere_Simulation
C
C Author: Darryn Evans
C Created: 4th March 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL ra,beta,eps,es
C
      INCLUDE 'main.inc'
C
      precip =ra/10.
      IF (precip .gt. evapor) THEN
        es =esoil
      ELSE
        IF (beta .lt. 0.) THEN
          beta =0.
        END IF
        beta =(moist(1)-tr(1))/(ts(1)-tr(1))
        eps =precip+((evapor-precip)*(beta**p))
        es =eps*(1-lai)
      END IF
C
C recalculate moisture content of the layer
C
      moist(1) =moist(1)-(es/d(1))
```

```
      IF (moist(1) .lt. tr(1)) THEN
        xcess =tr(1)-moist(1)
        moist(1) =tr(1)
      END IF
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Root_Abstraction
C
C The purpose of this subroutine is to firstly using potential transpiration
C to calculate the potential leaf suction from this using Rijtema (1965)
C relationship to calculate actual leaf suction, then aroot abstraction
C algorithm modelling the change in potentials from the root to the leaf mode
ls
C the water uptake from each of the layers in the root zone.
C
C Variable Names and Meanings
C sl/sa =potential/actual leaf suction
C trans =potential transpiration
C neg =indicates if any of the abstractions are negative, if the case the lea
f
C       suctions are recalculated
C alpha =parameter used in conversion of potential to actual leaf suction
C q(j) =water uptake from the Jth layer
C a/x/y =arrays used to hold abstraction calculations for each layer
C ya/xa =sum up abstraction calculations form each layer
C rootp =plant resistance
C
C Called By:
C PROGRAM Soil_Atmosphere_Simulation
C
C Author: Darryn Evans
C Created: 3rd March 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL sl,sa,trans,alp,q(50),a(50),x(50),y(50),ya,xa,tempn(50)
      REAL resis(50),test(50),dr(50),sumdr,sumrd,d0
      LOGICAL neg
C
      INCLUDE 'main.inc'
C
      IF (eplant .gt. 0.) THEN
        xa =0.
        ya =0.
        DO 1 j =1,(nolyrs-1)
          tempn(j) =rdens(j)
          q(j) =0.
    1   CONTINUE
C
C using the potential transpiration in the root abstraction equations calcula
te
C the potential leaf suction
C
        DO 2 j =1,rotdep
          x(j) =tempn(j)*(moist(j)/ts(j))*head(j)
          y(j) =tempn(j)*(moist(j)/ts(j))
    2   CONTINUE
        neg =.true.
        DO WHILE (neg .eq. .true.)
          DO 3 j =1,rotdep
```

```
              xa =xa+(x(j)/rootp)
              ya =ya+(y(j)/rootp)
    3       CONTINUE
            IF (ya .gt. 0.) THEN
              sl =(eplant+xa)/ya
              neg =.false.
              DO 5 j =1,rotdep
C
C test to ensure water flow is not downwards i.e flow negative
C
                test(j) =tempn(j)*(sl-head(j))
                IF (test(j) .lt. 0.) THEN
                  neg =.true.
                  x(j) =0.
                  y(j) =0.
                  tempn(j) =0.
                  xa =0.
                  ya =0.
                END IF
    5           CONTINUE
              ELSE
                DO 8 j =1,rotdep
                  q(j) =0.
    8           CONTINUE
                neg =.false.
              END IF
            END DO
C
C Now calculate sa (actual leaf suction)
C
            alp =sl/70000
            IF (alp .gt. 1.0) THEN
              alp =1.0
            END IF
            sa =(70000*(alp-((alp**3)/3.))-0.000001)
C
C substitute actual leaf suction back into the equation to gain the actual
C transpiration and root abstraction
C
             IF (ya .gt. 0.) THEN
               DO 4 j =1,rotdep
                 q(j) =(tempn(j)*(moist(j)/ts(j))*(sa-head(j)))
     1                                                    /rootp
                 IF (q(j) .lt. 0.) THEN
                   q(j) =0.
                 END IF
    4            CONTINUE
             END IF
C
C now recalculate the moisture content of the layer
C
            DO 6 j =1,rotdep
              moist(j) =moist(j)-(q(j)/d(j))
              IF (moist(j) .lt. tr(j)) THEN
                xcess =tr(j)-moist(j)
                moist(j) =tr(j)
                q(j) =q(j)-(xcess*d(j))
              END IF
    6       CONTINUE
C
C if abstraction from one of the layers was negative recalculate actual leaf
C suction
C
            DO 7 j =1,rotdep
              actran =actran+q(j)
    7       CONTINUE
            IF (actran .gt. eplant) THEN
              actran =eplant
```

```
        END IF
      ELSE
        DO j =1,rotdep
          q(j) =0.
        END DO
        actran =0.
        sl =0.
        sa =0.
      END IF
C
      WRITE(82,120)q(1),q(2),q(3),actran
      WRITE(76,121)ndays,sl,sa,resis(1),resis(2),resis(3)
C
  120 FORMAT(1x,4(2x,f10.8))
  121 FORMAT(1x,i3,5(2x,f10.2))
C
      RETURN
      END

C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Plant_Growth
C
C The aim of this routine is to simulate plant growth, this is achieved by
C relating the amount of growth to actual transpiration, this growth is then
C partitioned between the roots and the leaves according to the amount of
C stress the plant is under.
C
C Local Variable Names and Meanings:
C potlai =potential leaf area
C pltfra =ratio of growth distribution between leaves and roots
C growth =amount of growth units for the day
C stress =a measure of the amount of stress a plant is under by comparing
C         actual transpiration to potential
C sg(n) =partitioning variable used for determining the amount of root
C        growth in each layer
C sumprod =used to partition root growth between layers
C minwtt =moisture content of the driest layer in the rootzone
C minwet =driest layer in the rootzone
C xtra =plant biomass not destroyed in previous timestep as all roots in
C       affected layer were already dead
C gromax =maximum potential growth
C reqext =amount of days required of growth for roots to reach the bottom
C         of the layer
C extens =number of days roots having been grown in the deepest layer of
C         penetration
C
C Called By:
C PROGRAM Soil_Atmosphere_Simulation
C
C Author: Darryn Evans
C Created: 19/06/89
C Updated:
C
      REAL pltfra,growth,stress,lam2,den,maxwtt,minwtt,xtra
      REAL gromax,sg(50),sumprod
      INTEGER maxwet,minwet,extens,reqext
C
      INCLUDE 'main.inc'
C
C calculate the number of growth units
C
```

```
      IF (rotdep .eq. 1) THEN
        reqext =0
        extens =0
      ELSE
        rl =d(rotdep)/extrat
        reqext =rl
      END IF
      IF (plntyp .eq. 1) THEN
        gromax =a4
      ELSE
        tem =(real(ndays)-real(emerg))
        teml =tem/(real(harv)-real(emerg))
        tem2 =((teml**f4)*maxgr)
        gromax =maxgr-tem2
      END IF
      growth =gromax*(1-(exp(-(actran*b4))))
      IF (growth .gt. 0.) THEN
C
C from this partition the growth units between the leaves and the roots
C
        stress =1.-(actran/eplant)
        lam2 =1.-(1./lam)
        den =((1.+((koef*stress)**lam))**lam2)
        pltfra =minrat+((maxrat-minrat)/den)
        potlai =potlai+((pltfra*growth)*lc)
        lai =maxlai*(1-(exp(-(c4*potlai))))
        wtruse =wtruse+actran
C
C initialise root growth functions for partitioning root growth between
C layers
C
        DO j =1,nolyrs
          sg(j) =0.
        END DO
        sumprod =0.
C
C calculate root growth within each layer
C
        sg(1) =0.25
        DO j =1,rotdep
          IF (head(j+1) .lt. head(j)) THEN
            sg(j) =sg(j)+0.75
            IF (j .lt. rotdep) THEN
              sg(j+1) =sg(j+1)+0.25
            END IF
          END IF
        END DO
        IF (rotdep .lt. (nolyrs-1)) THEN
          IF ((head(rotdep+1) .lt. head(rotdep)) .and.
     1                                (extens .lt. reqext)) THEN
            extens =extens+1
          END IF
          IF (extens .eq. reqext) THEN
            rotdep =rotdep+1
            extens =0
            sg(rotdep) =0.25
            rdens(rotdep) =0.1*rdens(rotdep-1)
          END IF
        END IF
        DO j =1,rotdep
          sumprod =sumprod+(sg(j)*rdens(j))
        END DO
C
C Apply growth to the appropriate layers. Root density is calculated to
C increase by a certain amount a cubic metre then converted to the layers
C in which they are growing.
C
        rgrow =(((1-pltfra)*growth)*rc)
```

```
        DO j =1,rotdep
          rdens(j) =rdens(j)+(((sg(j)*rdens(j))/sumprod)*
     1                                        (rgrow*(1000./d(j))))
        END DO
C
C Now depending on the plant type senescence is calculated
C
        IF (plntyp .eq. 1) THEN
C
C determine in which layer root death will be in
C
          DO 1 j =1,rotdep
            IF (j .eq. 1) THEN
              minwtt =head(j)
              minwet =j
            ELSE
              IF (head(j) .gt. minwtt) THEN
                minwtt =head(j)
                minwet =j
              END IF
            END IF
     1      CONTINUE
          rdens(minwet) =rdens(minwet)-((rgrow+xtra)*(1000./d(minwet)))
          xtra =0.
          IF (rdens(minwet) .lt. 0.) THEN
            xtra =(0.-rdens(minwet))*(d(minwet)/1000.)
            rdens(minwet) =0.
C
C if this layer is the bottom layer of the rootzone all roots are assumed
C to be gone and the maximum extent of the rootzone decreases by one. If it i
s
C any layer above this it is only the absorbing roots that are dead.
C
            IF (minwet .eq. rotdep) THEN
              rotdep =rotdep-1
            END IF
          END IF
C
C For monocots assumed no loss of root density just weight. Leaves are assume
d
C to stay until harvest
C
        END IF
      END IF
      euse =euse+eplant
      WRITE(58,*)'j,euse,wtruse',ndays,euse,wtruse
      IF (ndays .eq. harv) THEN
        totuse =wtruse/euse
        WRITE(58,*)' '
        WRITE(58,*)'wtruse,euse',wtruse,euse
        WRITE(58,*)' '
        wtruse =0.
        euse =0.
      END IF
      gu =growth
C
      RETURN
      END
```

```
      PROGRAM Calculate_Column_Drainage
C
C The purpose of this program is to simulate the drainage of Pete Hedges's
C (1989) soil columns using appropriate routine from XINFIL.FOR. Columns are
C drained from saturation, there is no input from the surface and no permanen
t
C groundwater function.
C
      REAL pb,grav,sa,si,cl,vco,co,med,fi,vfi,cosi,fsi,clay,sand
      REAL silt,drnd(50),sumdr,dr,n3,n4,n5,n6,n7,aa1,a2,a3,tt1,t2
      REAL t3,t4,wat(50),m12
      INTEGER time1(50),kount,out
      CHARACTER sample
      LOGICAL much
C
      INCLUDE 'main.inc'
C
      DATA n3,n4,n5 / 3.156677E-05,.06587,4.859883E-06 /
      DATA n6,n7 / -2.64358E-03,-4.28877E-04 /
      DATA aa1,a2,a3 / 1.765137E-03,2.523676E-03,-3.97461E-06 /
      DATA tt1,t3,t4 / 1.578954E-08,4.189656E-03,-3.60168E-04 /
      DATA nin,nin1,nin2,nin3 / 30,18,19,16 /
      DATA out / 14 /
C
      nolyrs =21
      READ(nin1,100)nsoils
      READ(nin,100)nosoils
      nos =nosoils-8
      WRITE(out,100)nos
      DO 1 j =1,nosoils
        print*,'Run Number ',j
        READ(nin,100)nores
        DO 2 i =1,nores
          READ(nin,101)time1(i),wat(i)
          IF (i .eq. 1) THEN
            IF (time1(i) .lt. 1) THEN
              nres =nores-1
              IF (wat(1) .gt. 0.) THEN
                temp =wat(1)
              END IF
            END IF
          END IF
    2   CONTINUE
        IF (nres .lt. nores) THEN
          DO i =1,nores
            IF (i .eq. nores) THEN
              time1(i) =0
              wat(i) =0.
            ELSE IF ((i .eq. 1) .and. (temp .gt. 0.)) THEN
              time1(i) =time1(i+1)
              wat(i) =wat(i+1)+temp
              temp =0.
            ELSE
              time1(i) =time1(i+1)
              wat(i) =wat(i+1)
            END IF
          END DO
          nores =nres
        END IF
C
C Read in textural data
C
        READ(nin1,102)sample
        READ(nin1,103)pb,ts
        READ(nin1,104)grav,sa,si,cl
        READ(nin2,105)vco,co,med,fi,vfi
        READ(nin2,106)cosi,fisi,clay
        sand =vco+co+med+fi+vfi
```

```
        silt =cosi+fisi
        READ(nin3,108)ks
        IF ((j .eq. 3) .or. (j .eq. 6) .or. (j .eq. 8) .or. (j .eq. 10)
     1       .or. (j .eq. 11) .or. (j .eq. 12) .or. (j .eq. 15) .or.
     2       (j .eq. 16)) THEN
          PRINT*,'Error in Data'
        ELSE
C
C Calculate parameter values
C
          bar15 =0.04367+(tt1*((sand**2.)*(clay**2.)))
     1          +(t3*clay)+(t4*sand)
          tr =bar15
          n1 =1.33295+(n3*(fi**3.))+(n4*fi)+
     1          (n5*(sand**3.))+(n6*(fi**2.))+(n7*(sand**2.))
          alpha =.0267+(aa1*co)+(a2*vco)+(a3*(silt**2.))
          DO 3 i =1,nolyrs
            d(i) =10.
    3     CONTINUE
          d(20) =5.
          d(21) =5.
          m12 =1.-(1./n1)
          lim =tr+((ts-tr)/((1+((alpha*3.4999999)**n1))**m12))
C
C Start the calculations and initialize
C
          DO 4 i =1,nolyrs
            moist(i) =ts
    4     CONTINUE
          kount =1
          DO 5 i =1,time1(nores)
            CALL Minute(dr)
            IF (i .eq. time1(kount)) THEN
              drnd(kount) =sumdr+dr
              sumdr =0.
              kount =kount+1
            ELSE
              sumdr =sumdr+dr
            END IF
    5     CONTINUE
C
C Write results to file
C
          WRITE(out,100)nores
          DO 6 i =1,nores
            WRITE(out,107)time1(i),wat(i),drnd(i)
    6     CONTINUE
        END IF
    1 CONTINUE
C
  100 FORMAT(1x,i3)
  101 FORMAT(1x,i8,3x,f16.8)
  102 FORMAT(1x,a40)
  103 FORMAT(1x,2(f8.5,2x))
  104 FORMAT(1x,4(f8.5,2x))
  105 FORMAT(1x,5(f8.4,2x))
  106 FORMAT(1x,4(f8.4,2x))
  107 FORMAT(1x,i8,3x,f16.8,3x,f16.8)
  108 FORMAT(1x,f16.8)
C
      STOP
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
```

```
***
C
      SUBROUTINE Minute(rdr)
C
      REAL rdr
      LOGICAL few
C
      INCLUDE 'main.inc'
C
      time =60.
      CALL Cal_Exch(few,rdr)
      IF (few .eq. .true.) THEN
        CALL Sec(rdr)
      END IF
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Sec(rdr1)
C
      REAL rdr1,rdr
      LOGICAL few
C
      INCLUDE 'main.inc'
C
      rdr1 =0.
      time =1.
      DO 1 j =1,60
        CALL Cal_Exch(few,rdr)
        rdr1 =rdr1+rdr
    1 CONTINUE
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Cal_Exch(much,dra)
C
      REAL dra
      LOGICAL much
C
      INCLUDE 'main.inc'
C
      DO 1 ij =1,nolyrs
        wet(ij) =moist(ij)
    1 CONTINUE
      CALL Perc(much,dra)
      IF ((much .eq. .false.) .or. (time .lt. 1.1)) THEN
        DO 2 ij =1,nolyrs
          moist(ij) =wet(ij)
    2   CONTINUE
      ELSE
        dra =0.
      END IF
C
```

```
      RETURN
      END
C
C ***********************************************************************
***
C *
  *
C ***********************************************************************
***
C
      SUBROUTINE Perc(toomuc,rdr)
C
C This routine is used within the program for periods when ther is no macropo
re
C or infiltration wetting fronts
C
C Local Variable Names and Meanings:
C kb(n) =harmonically averaged mean across the nth boundary
C exch(n) =exchange of water across the nth boundary
C v12 =exchange of water across boundary
C excess =water above saturation which is passed back up
C toomuc =logical variable used to decide if too much water has been passed
C          across the boundary
C gain =amount of water gained by the layer containing the groundwater table
C
      REAL kb(50),exch(50),v12,excess
      INTEGER count
      LOGICAL toomuc
C
      INCLUDE 'main.inc'
C
      DO 1 j =1,nolyrs
        CALL Suction_Cal(wet(j),ts,j)
        CALL Conductivity_Cal(wet(j),ts,j)
    1 CONTINUE
C
C work out the average conductivity for each layer
C
      DO 2 j =1,(nolyrs-1)
        IF (j .eq. (nolyrs-1)) THEN
          kb(j) =conduc(j)
        ELSE
          kb(j) =((d(j)+d(j+1))/((d(j)/conduc(j))+(d(j+1)
     1                                    /conduc(j+1))))
        END IF
    2 CONTINUE
C
C Calculate exchanges for each layer
C
      DO 3 j =1,(nolyrs-1)
        CALL Exchange_Cal(v12,kb,head,d,time,j)
        exch(j) =v12
    3 CONTINUE
C
C Test to see if any of the exchanges across boundaries are greater than that
C which is allowed, if this is so pass back to calling routine to reduce the
C timestep, unless the smallest timestep is already in use
C
      toomuc =.false.
      count =1
      DO WHILE ((toomuc .eq. .false.) .and. (count .le. (nolyrs-1)))
        IF (((exch(j)/d(j)) .gt. (0.1*wet(j))) .and. (time .gt. 1.))
                                                          THEN
     1
          toomuc =.true.
        END IF
        count =count+1
      END DO
C
```

```
      IF ((toomuc .eq. .true.) .and. (time .gt. 1.)) THEN
        RETURN
      ELSE
C
C Apply moisture changes to the layer
C
        DO 4 j =1,nolyrs
          IF (j .eq. 1) THEN
            wet(j) =wet(j)-(exch(j)/d(j))
          ELSE IF (j .eq. (nolyrs-1)) THEN
            wet(j) =wet(j)+((exch(j-1)-exch(j))/d(j))
            IF (wet(j) .lt. lim) THEN
              drain =(lim-wet(j))*d(j)
              wet(j) =lim
              exch(j) =exch(j)-drain
            END IF
          ELSE IF (j .eq. nolyrs) THEN
            rdr =exch(j-1)
          ELSE
            wet(j) =wet(j)+((exch(j-1)-exch(j))/d(j))
          END IF
    4   CONTINUE
C
C Check to ensure that no layers become greater than saturated
C
        DO 5 j =(nolyrs-1),1,-1
          excess =0.
          IF (wet(j) .gt. ts) THEN
            excess =wet(j)-ts
            wet(j) =ts
            IF (j .eq. 1) THEN
              PRINT*,'Error in Calculations'
              STOP 77
            ELSE
              wet(j-1) =wet(j-1)+(excess*(d(j)/d(j-1)))
            END IF
          END IF
    5   CONTINUE
      END IF
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Suction_Cal(vmc,sat,lyr)
C
C The purpose of this subroutine is to calculate the suction in each layer
C depending on the moisture content. The equation used is that of Genuchten
C (1980) with the parameter values being determined by statistical means.
C
C Local Variable Names and Meanings
C lyr =layer of profile for which calculations are taking place
C vmc =moisture content of layer
C sat =saturated moisture content of layer
C m =Genuchten(1980) equation parameter
C relmc =inverse of relative moisture content
C
      REAL vmc,sat,m,relmc
      INTEGER lyr
C
      INCLUDE 'main.inc'
C
```

```
      IF (vmc .gt. sat) THEN
        head(lyr) =0.
      ELSE
        IF (vmc .le. tr) THEN
          head(lyr) =0.3E+06
        ELSE
          relmc =(sat-tr)/(vmc-tr)
          m =1-(1/n1)
          head(lyr) =(((relmc**(1/m))-1)**(1/n1))/alpha
        END IF
      END IF
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
***
C
      SUBROUTINE Conductivity_Cal(vmc,sat,lyr)
C
C The subroutine calculates the value of conductivity using Genuchtens (1980)
C relationships between the suction and conductivity curves, and the estimate
d
C values of ks.
C
C Local Variable Names and Meanings:
C lyr =layer of profile for which calculations are taking part
C vmc =moisture content of layer
C sat =saturated moisture content of a layer
C m =Genuchten(1980) equation parameter
C relmc =inverse of relative moisture content for layer
C rl =relative moisture content
C kr =relative conductivity for layer
C
      REAL vmc,sat,m,relmc,rl,kr
      INTEGER lyr
C
      INCLUDE 'main.inc'
C
      IF (vmc .ge. sat) THEN
        conduc(lyr) =ks
      ELSE
        IF (vmc .le. tr) THEN
          kr =0.1E-20
        ELSE
          relmc =(sat-tr)/(vmc-tr)
          rl =1./relmc
          m =1-(1/n1)
          kr =(rl**0.5)*((1-((1-(rl**(1/m)))**m))**2)
          IF (kr .lt. (0.1E-20)) THEN
            kr =0.1E-20
          END IF
        END IF
        conduc(lyr) =kr*ks
      END IF
C
      RETURN
      END
C
C ***************************************************************************
***
C *
  *
C ***************************************************************************
```

```
***
C
      SUBROUTINE Exchange_Cal(v12,k,h,deph,t,lyr)
C
C The purpose of this routine is to calculate the exchange of water between
C layers using a much simplified analytical form of the Richards Equation.
C
C Local Variable Names and Meanings:
C lyr =layer of profile for which calculations are taking part
C v12 =exchange of water across boundary
C k =conductivtiy of layer
C h =head of layer
C deph =depth of layer
C t =length of timestep
C
C Called By:
C SUBROUTINE Redist_Percolation
C SUBROUTINE Percolation
C
C Author: Darryn Evans
C Created: 14th November 1988 (Adapted 30th November 1988)
C Updated: 20th March 1989
C
      REAL v12,k(50),h(50),deph(50),t
      INTEGER lyr
C
      INCLUDE 'main.inc'
C
      IF (lyr .eq. (nolyrs-1)) THEN
        v12 =(k(lyr)*(((1-h(lyr))/((deph(lyr)/2.)))+1))*t
      ELSE
        v12 =(k(lyr)*(((h(lyr+1)-h(lyr))/((deph(lyr)+deph(lyr+1))
     1                                                    /2))+1))*t
      END IF
      IF (v12 .lt. 0.) THEN
        v12 =0.
      END IF
C
      RETURN
      END
```

```
      PROGRAM Genu_Parameter_Find
C
C The purpose of this program is to find the values of the unknown parameters
C in Van Genuchtens (1980) closed form suction equation for different soil ty
pes
C this is achieved with the aid of NAG subroutine e04JAF
C
C Written in: 14/04/88
C Updated: 13/07/88 to include statistical analysis
C
      REAL*8 f,sat1,sat,bl(3),bu(3),w(1000),x(3),y(19),h(19),fmin(19)
      REAL*8 resid(19),m2,pb,xa(3),tr
      INTEGER ibound,ifail,liw,lw,n,nout,nout1,iw(1000),nobs,nom
C
      PARAMETER (nout=88,nout1=89)
      COMMON sat,tr,y,h
C
      n = 3
      lw =200
      liw =200
      READ(18,101)number
      WRITE(nout1,101)number
      DO 1 j =1,number
C
C input initial guesses for starting points and saturated moisture content, d
ata
C read from file with exception of sat is only used in order to advance the
C file to next point of interest, file is also used by related programs
C
        READ(18,102)sample
        READ(18,103)pb,sat
        READ(18,104)gra,sa,si,cl
        x(1) =0.02
        x(2) =2.5
        x(3) =0.05
C
C read in soil moisture data and rescaling
C
        READ(20,110)nobs
        DO 2 i =1,nobs
          READ(20,111)h(i),y(i)
C
C convert the head to cms
C
          h(i) =h(i)/10.
    2   CONTINUE
C
C read in upper and lower boundary limits for calculation
C
        ibound =0
        bl(1) =0.0
        bu(1) =0.3
        bl(2) =1.
        bu(2) =20.
        bl(3) =0.0000000001
        bu(3) =0.9
        ifail =1
C
C call nag subroutine to work out least squares using a minimising function
C
        CALL E04JAF(n,ibound,bl,bu,x,f,iw,liw,w,lw,ifail)
C
C test condition of ifail to see if problems have arisen within the program
C
        write(nout,122) ifail
C
C write results to external file
C
```

```
      m2 =1-(1/x(2))
      DO 10 i =1,nobs
        fmin(i) =x(1)+((sat-x(1))/((1+((x(3)*h(i))**x(2)))**m2))
        resid(i) =y(i)-fmin(i)
10    CONTINUE
      DO 11 i =1,nobs
        WRITE(nout1,120)y(i),fmin(i)
11    CONTINUE
      WRITE(nout,121) j
      WRITE(nout,123) f,sat
      WRITE(nout,124) x(1),x(2),x(3)
      DO 12 i =1,nobs
        WRITE(nout,126)y(i),fmin(i),resid(i)
12    CONTINUE
 1  CONTINUE
C
 100 FORMAT(1x,'RESULTS OF FITTING EQUATION TO DATA'//)
 101 FORMAT(1x,i4)
 102 FORMAT(1x,a40)
 103 FORMAT(1x,2(f8.5,2x))
 104 FORMAT(1x,4(f8.5,2x))
 110 FORMAT(1x,i3)
 111 FORMAT(1x,e18.12,3x,f10.6)
 120 FORMAT(1x,2(f8.5,2x))
 121 FORMAT(1x,'Run Number =',i3)
 122 FORMAT(1x/1x'Failure Number =',i2)
 123 FORMAT(1x,'Minumum Value is',f12.8,2x,'Saturation Value =',f8.6)
 124 FORMAT(1x,'thetaR, n & alpha',3e16.6/)
 126 FORMAT(1x,'obs  ',f8.6,2x,'calc  ',f8.6,2x,'resid  ',f8.6)
 200 FORMAT(1x,i3)
 201 FORMAT(1x,f7.5)
 202 FORMAT(1x,3(f9.5,3x))
 203 FORMAT(1x,f7.5,4x,f16.6)
C
      STOP
      END
C
C routine to evaluate the residuals
C
      SUBROUTINE FUNCT1(n,xc,fc)
C     FUNCTION EVALUATION ROUTINE FOR E04JAF EXAMPLE PROGRAM -
C     THIS ROUTINE MUST BE CALLED FUNCT1
C     .. SCALAR ARGUMENTS ..
      REAL*8 fc,sumsq,yprime(19),xc(n),x1,x2,x3,m,alpha,alphah
     1     ,alp1,sat,y(19),h(19),tr
      INTEGER n,nobs
      COMMON sat,tr,y,h
C
      x1 =xc(3)
      x2 =xc(2)
      x3 =xc(1)
      nobs =19
      DO 1 i =1,nobs
        m =1.-(1./x2)
        alp1 =(x1*h(i))**x2
        alphah =(1.+alp1)**m
        yprime(i) =x3+((sat-x3)/alphah)
    1 CONTINUE
      sumsq =0.
      DO 2 i =1,nobs
        sumsq =sumsq+((y(i)-yprime(i))*(y(i)-yprime(i)))
    2 CONTINUE
      fc =sumsq
      RETURN
      END
```

APPENDIX 3

APPENDIX 3 Weather Data For Preston Vale Farm for Simulation Period: R (rainfall mm/day) and ED (evaporative demand cm/day).

Day No.	R	ED	Day No.	R	ED	Day No.	R	ED
1	2.1	0.01	51	0.0	0.07	101	4.7	0.21
2	3.9	0.00	52	0.0	0.09	102	1.6	0.21
3	0.7	0.00	53	0.0	0.06	103	0.0	0.16
4	3.6	0.04	54	3.8	0.06	104	0.0	0.22
5	1.2	0.02	55	0.3	0.06	105	0.0	0.32
6	2.5	0.02	56	0.0	0.18	106	0.0	0.36
7	5.2	0.06	57	0.0	0.09	107	0.0	0.35
8	2.4	0.05	58	0.3	0.09	108	0.0	0.30
9	15.0	0.02	59	6.1	0.16	109	0.0	0.36
10	16.4	0.00	60	1.1	0.15	110	0.0	0.45
11	1.9	0.09	61	0.2	0.24	111	0.0	0.49
12	4.8	0.02	62	2.7	0.17	112	0.0	0.32
13	0.2	0.02	63	0.2	0.14	113	0.0	0.38
14	0.0	0.02	64	0.5	0.10	114	0.0	0.38
15	11.3	0.02	65	0.3	0.19	115	0.0	0.42
16	0.0	0.02	66	0.0	0.08	116	0.0	0.47
17	3.9	0.01	67	0.0	0.15	117	0.0	0.37
18	11.0	0.10	68	0.0	0.14	118	0.0	0.16
19	6.5	0.03	69	3.4	0.11	119	0.0	0.29
20	4.2	0.02	70	1.7	0.14	120	0.0	0.29
21	2.0	0.01	71	1.8	0.11	121	0.0	0.35
22	7.9	0.03	72	0.1	0.44	122	0.0	0.36
23	2.5	0.03	73	0.0	0.35	123	0.0	0.48
24	6.3	0.03	74	0.0	0.11	124	4.5	0.16
25	0.6	0.02	75	0.0	0.15	125	0.4	0.23
26	0.0	0.03	76	0.0	0.09	126	5.0	0.26
27	0.0	0.01	77	0.3	0.11	127	0.6	0.33
28	7.2	0.04	78	4.6	0.12	128	0.3	0.35
29	2.1	0.04	79	5.2	0.13	129	5.0	0.22
30	1.8	0.10	80	1.3	0.09	130	16.1	0.17
31	0.4	0.15	81	2.8	0.23	131	4.9	0.20
32	0.0	0.19	82	3.4	0.25	132	20.3	0.28
33	0.0	0.12	83	0.3	0.26	133	7.9	0.12
34	0.0	0.06	84	8.5	0.13	134	7.8	0.10
35	0.6	0.07	85	2.3	0.21	135	0.5	0.14
36	0.0	0.09	86	0.7	0.28	136	0.0	0.30
37	0.0	0.16	87	1.9	0.24	137	0.0	0.23
38	1.2	0.09	88	5.1	0.22	138	0.0	0.16
39	1.0	0.19	89	4.5	0.22	139	0.0	0.30
40	0.0	0.08	90	2.2	0.19	140	0.0	0.28
41	0.3	0.08	91	8.9	0.16	141	0.0	0.36
42	2.8	0.16	92	6.2	0.19	142	0.0	0.39
43	8.5	0.04	93	0.5	0.12	143	0.0	0.33
44	2.1	0.06	94	0.0	0.19	144	0.0	0.16
45	0.8	0.16	95	3.7	0.23	145	1.6	0.29
46	2.4	0.10	96	7.0	0.17	146	0.0	0.30
47	1.1	0.11	97	0.5	0.16	147	2.7	0.15
48	1.7	0.08	98	3.1	0.12	148	30.2	0.21
49	0.1	0.11	99	0.0	0.25	149	0.0	0.21
50	2.7	0.05	100	4.8	0.27	150	5.7	0.27

Day No	R	ED	Day No.	R	ED	Day No.	R	ED
151	1.1	0.20	205	0.0	0.15	259	0.0	0.03
152	0.0	0.40	206	17.4	0.19	260	0.0	0.05
153	0.0	0.43	207	0.4	0.27	261	2.5	0.03
154	0.0	0.37	208	0.0	0.06	262	0.8	0.04
155	0.0	0.41	209	0.0	0.13	263	0.9	0.03
156	0.0	0.48	210	2.8	0.19	264	0.0	0.07
157	0.0	0.44	211	0.0	0.14	265	3.6	0.13
158	0.0	0.21	212	0.0	0.19	266	3.1	0.04
159	0.0	0.22	213	2.0	0.12	267	0.0	0.00
160	0.0	0.38	214	0.1	0.24	268	0.0	0.02
161	0.0	0.19	215	0.0	0.19	269	1.0	0.06
162	0.0	0.15	216	3.2	0.11	270	0.4	0.14
163	0.0	0.17	217	0.0	0.29	271	0.0	0.03
164	0.0	0.37	218	1.2	0.11	272	3.1	0.06
165	0.0	0.29	219	0.0	0.15	273	0.4	0.06
166	0.0	0.30	220	2.5	0.13	274	19.5	0.04
167	2.2	0.24	221	0.6	0.17	275	2.4	0.01
168	0.3	0.29	222	0.4	0.11	276	0.0	0.40
169	6.2	0.38	223	0.1	0.16	277	1.7	0.05
170	0.6	0.28	224	0.0	0.22	278	1.5	0.00
171	0.0	0.21	225	0.0	0.10	279	0.5	0.05
172	0.2	0.22	226	0.0	0.13	280	15.0	0.00
173	0.0	0.26	227	0.0	0.16	281	2.9	0.06
174	1.5	0.38	228	0.0	0.21	282	0.8	0.03
175	1.5	0.22	229	0.0	0.11	283	1.5	0.04
176	0.4	0.22	230	0.0	0.09	284	0.0	0.08
177	0.0	0.28	231	0.0	0.14	285	17.7	0.11
178	0.0	0.15	232	0.0	0.10	286	1.8	0.09
179	0.0	0.21	233	0.1	0.09	287	4.8	0.03
180	0.0	0.20	234	0.0	0.08	288	1.3	0.11
181	0.0	0.35	235	0.0	0.09	289	0.0	0.05
182	0.0	0.27	236	2.5	0.13	290	0.2	0.04
183	0.0	0.26	237	3.3	0.15	291	0.0	0.02
184	0.0	0.24	238	1.5	0.06	292	3.9	0.02
185	1.7	0.30	239	0.4	0.10	293	0.0	0.02
186	0.1	0.33	240	0.5	0.17	294	0.0	0.00
187	0.9	0.10	241	0.3	0.13	295	1.5	0.07
188	0.0	0.05	242	4.2	0.08	296	6.8	0.03
189	0.0	0.16	243	0.2	0.14	297	0.0	0.06
190	0.0	0.25	244	0.0	0.16	298	0.0	0.04
191	0.0	0.29	245	4.0	0.09	299	0.0	0.02
192	0.0	0.27	246	2.1	0.10	300	0.0	0.00
193	0.0	0.29	247	0.0	0.05	301	0.0	0.00
194	0.0	0.12	248	4.0	0.08	302	0.0	0.00
195	1.9	0.12	249	0.0	0.08	303	0.0	0.02
196	0.0	0.16	250	5.8	0.05	304	0.8	0.02
197	3.7	0.19	251	4.3	0.03	305	0.0	0.00
198	8.6	0.15	252	0.0	0.11	306	0.0	0.02
199	7.7	0.11	253	3.0	0.04	307	0.0	0.01
200	2.3	0.13	254	0.0	0.12	308	0.8	0.02
201	0.0	0.22	255	0.0	0.09	309	5.6	0.00
202	5.9	0.20	256	0.0	0.05	310	12.3	0.02
203	0.0	0.25	257	0.0	0.05	311	2.6	0.00
204	1.5	0.24	258	0.0	0.03	312	5.3	0.00

Day No.	R	ED	Day No.	R	ED	Day No.	R	ED
313	9.0	0.00	367	1.0	0.00	421	3.3	0.12
314	1.0	0.02	368	9.5	0.10	422	1.0	0.29
315	1.9	0.03	369	4.2	0.01	423	0.3	0.19
316	0.1	0.01	370	2.9	0.00	424	2.5	0.11
317	0.0	0.01	371	0.2	0.02	425	5.0	0.13
318	0.0	0.00	372	0.3	0.06	426	0.0	0.13
319	0.1	0.01	373	0.8	0.03	427	0.0	0.09
320	6.1	0.00	374	0.0	0.00	428	0.0	0.15
321	0.3	0.00	375	0.0	0.00	429	0.0	0.20
322	0.1	0.00	376	0.0	0.00	430	0.0	0.20
323	0.0	0.00	377	0.0	0.02	431	0.0	0.13
324	3.8	0.03	378	0.0	0.03	432	0.0	0.16
325	1.3	0.00	379	0.0	0.01	433	0.7	0.12
326	0.7	0.02	380	0.0	0.03	434	2.3	0.19
327	0.0	0.06	381	0.0	0.03	435	5.9	0.21
328	0.5	0.05	382	0.0	0.01	436	2.3	0.07
329	1.0	0.00	383	0.0	0.02	437	4.2	0.15
330	6.2	0.01	384	0.0	0.05	438	0.4	0.25
331	0.0	0.03	385	0.1	0.02	439	0.0	0.23
332	0.3	0.04	386	1.5	0.06	440	0.0	0.15
333	1.4	0.03	387	3.2	0.01	441	3.1	0.16
334	0.4	0.01	388	2.3	0.05	442	0.0	0.14
335	0.5	0.05	389	0.2	0.04	443	4.3	0.16
336	4.7	0.00	390	5.8	0.01	444	3.0	0.15
337	2.3	0.04	391	0.5	0.05	445	0.8	0.15
338	1.0	0.01	392	6.2	0.06	446	0.0	0.12
339	0.0	0.03	393	2.5	0.04	447	0.0	0.17
340	0.0	0.00	394	1.0	0.03	448	0.0	0.12
341	0.0	0.01	395	1.7	0.08	449	3.8	0.18
342	0.3	0.00	396	0.0	0.03	450	2.1	0.09
343	1.2	0.00	397	0.0	0.03	451	0.0	0.15
344	3.3	0.03	398	0.0	0.10	452	0.0	0.15
345	0.5	0.02	399	0.0	0.06	453	2.1	0.03
346	0.0	0.00	400	0.7	0.10	454	0.2	0.08
347	0.0	0.07	401	0.0	0.03	455	1.4	0.10
348	0.0	0.00	402	0.5	0.11	456	18.4	0.13
349	0.7	0.00	403	0.0	0.06	457	0.1	0.21
350	1.4	0.00	404	0.9	0.10	458	0.9	0.16
351	0.0	0.00	405	2.9	0.07	459	3.6	0.10
352	5.6	0.00	406	0.4	0.10	460	0.4	0.12
353	6.9	0.07	407	1.3	0.09	461	0.5	0.27
354	1.8	0.01	408	0.0	0.15	462	0.0	0.22
355	3.0	0.05	409	0.9	0.15	463	0.0	0.46
356	0.2	0.00	410	0.0	0.19	464	0.0	0.30
357	4.4	0.01	411	0.0	0.18	465	1.0	0.19
358	0.3	0.04	412	4.6	0.08	466	1.5	0.08
359	1.6	0.07	413	7.8	0.13	467	1.4	0.24
360	1.3	0.00	414	0.6	0.32	468	0.4	0.26
361	15.6	0.05	415	3.9	0.12	469	0.0	0.15
362	6.4	0.00	416	1.6	0.18	470	0.0	0.23
363	0.9	0.04	417	1.3	0.22	471	0.0	0.28
364	0.1	0.09	418	3.1	0.15	472	0.0	0.29
365	1.1	0.06	419	0.2	0.27	473	0.0	0.30
366	14.3	0.04	420	0.0	0.08	474	0.0	0.31

Day No.	R	ED	Day No.	R	ED
475	0.0	0.16	511	0.0	0.32
476	0.0	0.31	512	0.5	0.21
477	0.0	0.25	513	1.5	0.19
478	0.0	0.19	514	2.0	0.23
479	0.0	0.34	515	7.0	0.27
480	0.0	0.35	516	1.9	0.12
481	0.0	0.37	517	5.0	0.20
482	0.0	0.41	518	5.9	0.34
483	0.0	0.44	519	1.3	0.14
484	0.0	0.44	520	0.0	0.23
485	2.1	0.41	521	0.0	0.34
486	3.5	0.22	522	2.5	0.34
487	0.6	0.57	523	3.6	0.23
488	0.0	0.54	524	0.0	0.45
489	2.6	0.38	525	0.0	0.33
490	0.8	0.43	526	0.0	0.36
491	0.1	0.43	527	0.0	0.42
492	7.0	0.19	528	0.0	0.42
493	0.0	0.41	529	0.0	0.33
494	0.0	0.37	530	0.0	0.47
495	0.0	0.41	531	0.0	0.38
496	0.0	0.29	532	0.0	0.30
497	0.0	0.24	533	0.0	0.25
498	2.0	0.13	534	1.5	0.20
499	0.0	0.22	535	2.5	0.27
500	0.0	0.19	536	0.0	0.36
501	28.5	0.20	537	0.0	0.26
502	0.0	0.44	538	4.2	0.25
503	0.0	0.57	539	0.0	0.45
504	0.0	0.41	540	0.0	0.29
505	0.0	0.40	541	7.0	0.29
506	0.0	0.00	542	8.6	0.39
507	12.7	0.34	543	0.0	0.54
508	3.0	0.34	544	0.0	0.32
509	0.3	0.26	545	9.0	0.14
510	0.0	0.00	546	3.5	0.18

APPENDIX 3 Observed Moisture Deficits (cm) For Simulation Period Determined Using Hussein's Calibration Curve.

Day No.	Layer 1	Layer 2	Layer 3	Total
40	-0.70	2.10	-2.90	-1.50
65	-1.50	-0.60	-5.00	-7.10
86	-8.20	-2.00	-0.30	-10.50
93	-7.30	-5.40	-10.00	-22.70
107	5.70	0.70	-5.00	1.40
114	20.80	8.80	2.70	32.30
121	27.40	17.50	12.60	57.50
128	27.80	22.70	21.80	72.30
135	-4.40	-5.30	10.90	1.20
142	11.80	5.30	9.40	26.50
149	3.70	-1.50	2.60	4.80
156	16.90	9.10	12.00	38.00
172	25.90	20.40	29.60	75.90
179	28.80	24.50	37.00	90.30
184	30.50	27.90	42.90	101.30
191	32.00	28.10	45.10	105.20
205	13.60	22.10	46.20	81.90
219	20.90	19.70	40.00	80.60
233	28.60	25.10	43.50	97.20
247	24.90	25.80	44.60	95.10
261	19.20	25.00	44.60	88.80
275	-0.80	12.20	28.10	39.50
284	-0.50	3.10	16.30	18.90
289	-5.30	-0.10	4.80	-0.60
303	-1.30	2.90	1.50	3.10
317	-3.60	-0.80	-8.50	-12.90
345	-8.80	0.10	-3.20	-11.90
373	-7.50	-1.00	-9.20	-17.70
387	-5.90	4.00	1.10	-0.80
401	-0.60	1.70	-3.80	-2.70
415	-5.10	0.40	-0.20	-4.90
443	-2.40	1.70	-0.10	-0.80
450	-6.20	2.80	1.50	-1.90
457	-13.60	-5.20	-5.70	-24.50
463	-5.00	0.00	-4.30	-9.30
471	9.20	5.00	1.00	15.20
477	19.30	9.70	5.40	34.40
484	28.30	21.20	15.60	65.10
492	26.60	23.80	25.00	75.40
499	28.40	24.00	27.60	80.00
506	22.20	22.60	29.10	73.90
513	22.50	22.50	30.20	75.20
520	6.90	21.70	31.10	59.70
527	17.90	21.50	32.30	71.70
534	29.00	24.90	39.20	93.10
549	24.70	25.80	42.10	92.60