Computer source code for one-dimensional simulation of preferential solute transport in the vadose zone

```
(*************************************************************)
(* Program  : PrefFlow                                       *)
(* Purpose  : Simulation of preferential flow of water and so- *)
(*            lutes through the vadose zone using a piecewise  *)
(*            linear approximation of the hydraulic conducti-  *)
(*            vity as a function of water content to identify  *)
(*            pore groups in which water moves at distinct     *)
(*            velocities.                                      *)
(*                                                             *)
(* Interface: input                                            *)
(*            output                                           *)
(*            outfile1                                         *)
(*            infile1                                          *)
(*            outfile2                                         *)
(* Date     : 05/27/1991,  Last modification  07/04/1996       *)
(* Version  : 1.00                                             *)
(* Authors  : Bart Nijssen, Frank Stagnitti and                *)
(*            Tammo S. Steenhuis                               *)
(* Source   : Turbo Pascal                                     *)
(*                                                             *)
(*************************************************************)


PROGRAM PrefFlow (input,output,infile1,outfile1,outfile2);


USES
    Dos, Crt,Graph;

CONST
    MaxPoreGroups  = 8;       { maximum number of poregroups    }
    MaxNodes       = 650;     { maximum number of nodes         }

TYPE
    CutOffPoints   = ARRAY[0..MaxPoreGroups] OF REAL;
    PoreGroups     = ARRAY[1..MaxPoreGroups] OF REAL;
    TravelDistance = ARRAY[1..MaxPoreGroups] OF LONGINT;
    Mass           = RECORD
                         Water  : REAL;
                         Solute : REAL;
                     END;
    TransportMatrix
                   = ARRAY[1..MaxPoreGroups,1..Maxnodes] OF Mass;
    ClimatePtr     = ^Climate;
    Climate        = RECORD
                         RealTime : REAL;
                         Rainfall : REAL;
                         Concent  : REAL;
                         RealEvap : REAL;
                         Nxt      : ClimatePtr;
                     END;
```

```
VAR
    alpha                   { mixing coefficient              }
                    : PoreGroups;
    travelnodes             { number of nodes that water and  }
                            { solutes in a poregroup move du- }
                            { ring one big timestep           }
                    : TravelDistance;
    alphakappa,             { parameter for power or exponen- }
                            { tial                            }
    fraction,               { fraction that the gradient of the}
                            { conductivity curve is increased }
                            { from one poregroup to the next  }
    n1,                     { counters for writing to outfile1 }
    n2,
    np,                     { number of poregroups            }
    numberofnodes           { total number of nodes in column }
                    : LONGINT;
    deltax,                 { distance between two nodes      }
    initconcentration,      { initial concentration of soil   }
                            { moisture                        }
    initmoisture,           { initial moisture content        }
    lengthofexperiment,     { total length of experiment      }
    timestep,               { length of one timestep          }
    time                    { time elapsed since beginning of }
                            { experiment                      }
                    : REAL;
    m                       { reduced moisture contents       }
                    : CutOffPoints;
    porematrix              { matrix containing records with  }
                            { water and solute content for    }
                            { each node for each poregroup    }
                    : TransportMatrix;
    infile1,                { file containing weather data    }
    outfile1,               { file containing moisture profile }
                            { data at different times         }
    outfile2                { file containg outflow volume per }
                            { timestep                        }
                    : TEXT;
    weatheranchor           { pointer to first element of the }
                            { linked list of climate events   }
                    : ClimatePtr;
    graphdriver,
    graphmode       : INTEGER;
    key                     { dummy for KeyPressed            }
                    : CHAR;
```

```
(*************************************************************)
(* Procedure: OpenFiles                                      *)
(* Purpose  : Open different files containing the input and  *)
(*            output                                         *)
(* Interface: infile1  - var                                 *)
(*            outfile1 - var                                 *)
(*            outfile2 - var                                 *)
(*************************************************************)

PROCEDURE OpenFiles(VAR infile1,
                        outfile1,
                        outfile2 : TEXT);

VAR
    filename                { name of out- and infiles        }
                    : string[12];

BEGIN {openfiles}
    ClrScr;
    WRITELN ('Name of inputfile containing rainfall data, ',
             'evaporation data and');
    WRITE   ('concentration data: ');
    READLN  (filename);
    ASSIGN  (infile1,filename);
    WRITELN;
    WRITELN;
    WRITE   ('Name of outputfile containing moisture profile: ');
    READLN  (filename);
    ASSIGN  (outfile1,filename);
    WRITELN;
    WRITELN;
    WRITE   ('Name of outputfile containing outflow volume  : ');
    READLN  (filename);
    ASSIGN  (outfile2,filename);
    REWRITE (outfile2);
    WRITELN;
END {openfiles};
```

```
(*************************************************************)
(* Procedure: FindPoreGroups                                 *)
(* Purpose  : Calculates the moisture contents (mp) and the  *)
(*            accompanying hydraulic conductivities (kp) de- *)
(*            limiting the different poregroups (p). This is *)
(*            done by way of a piece wise linear approximation *)
(*            of the hydraulic conductivity curve.           *)
(* Interface: alpha              - var                       *)
(*            travelnodes        - var                       *)
(*            alphakappa         - var                       *)
(*            fraction           - var                       *)
(*            np                 - var                       *)
(*            numberofnodes      - var                       *)
(*            deltax             - var                       *)
(*            initconcentration  - var                       *)
(*            inimoisture        - var                       *)
(*            lengthofexperiment - var                       *)
(*            timestep           - var                       *)
(*            m                  - var                       *)
(*            outfile1           - var                       *)
(*************************************************************)


PROCEDURE FindPoreGroups(VAR alpha             : PoreGroups;
                         VAR travelnodes       : TravelDistance;
                         VAR alphakappa,
                             fraction,
                             np,
                             numberofnodes     : LONGINT;
                         VAR deltax,
                             initconcentration,
                             initmoisture,
                             lengthofexperiment,
                             timestep          : REAL;
                         VAR m                 : CutOffPoints;
                         VAR outfile1          : TEXT);


TYPE
    CutOffPointsP  = ARRAY[0..MaxPoreGroups] OF REAL;
    PoreGroupsP    = ARRAY[1..MaxPoreGroups] OF REAL;

VAR
    macromethod              { indicator for method to be used  }
                             { in dealing with macropores       }
                   : LONGINT;
    columnlength,            { length of column or profile      }
    kf,                      { satuated hydraulic conductivity  }
    ks,                      { saturated hydraulic conductivity }
    thetaf,                  { satuated moisture content        }
    thetam,                  { moisture content including extra }
                             { macropore                        }
    thetar,                  { residual moisture content        }
    thetas                   { saturated moisture content       }
                   : REAL;
```

```
functiontype             { indicator for power or exponen-  }
                         { tial, true in case of power      }
                : BOOLEAN;
changeinput              { indicates whether user wants to  }
                         { change input                     }
                : CHAR;
k                        { normalized hydraulic conductivity}
                : CutOffPointsP;
v                        { velocities of poregroups         }
                : PoreGroupsP;


(*********************************************************)
(* Procedure: ScreenIn1                                  *)
(* Purpose  : Asks the user to decide whether a power    *)
(*            or an exponential function needs to be used *)
(*            as approximation for the hydraulic conducti- *)
(*            vity. Prompts the user for the governing pa- *)
(*            rameters.                                  *)
(* Interface: alpha             - var                    *)
(*            alphakappa        - var                    *)
(*            fraction          - var                    *)
(*            macromethod       - var                    *)
(*            np                - var                    *)
(*            columnlength      - var                    *)
(*            initconcentration - var                    *)
(*            initmoisture      - var                    *)
(*            kf                - var                    *)
(*            ks                - var                    *)
(*            lengthofexperiment - var                   *)
(*            thetaf            - var                    *)
(*            thetar            - var                    *)
(*            thetar            - var                    *)
(*            thetas            - var                    *)
(*            realtimestep      - var                    *)
(*            functiontype      - var                    *)
(*********************************************************)


PROCEDURE ScreenIn1(VAR alpha        : PoreGroups;
                    VAR alphakappa,
                        fraction,
                        macromethod,
                        np           : LONGINT;
                    VAR columnlength,
                        initconcentration,
                        initmoisture,
                        kf,
                        ks,
                        lengthofexperiment,
                        thetaf,
                        thetam,
                        thetar,
                        thetas,
                        realtimestep : REAL;
                    VAR functiontype : BOOLEAN);
```

```
 VAR
    mixdecision,             { indicator for type of mixing }
    typeoffunction           { indicator for function type  }
                   : CHAR;
    p                        { counter for poregroups       }
                   : LONGINT;

BEGIN {screenin1}
    ClrScr;
    WRITELN ('In the Preferential Flow Model there are ',
             'three different ways to deal with');
    WRITELN ('macro-pores:');
    WRITELN ('  - Method 1: No macro-pores, the ',
             'conductivity curve is approximated by a');
    WRITELN ('              piecewise linear conductivity ',
             'curve, tangent to the original');
    WRITELN ('              conductivity curve');
    WRITELN ('  - Method 2: A macro-pore included within ',
             'the measured conductivity.');
    WRITELN ('              In this case the satuated ',
             'moisture content and conductivity');
    WRITELN ('              need to be known');
    WRITELN ('  - Method 3: Add a macro-pore on top of the ',
             'range of the measured');
    WRITELN ('              hydraulic conductivity.  In ',
             'this case the moisture');
    WRITELN ('              content including the macro-',
             'pore needs to be known.');
    WRITELN;
    WRITE   ('Method to be used (1, 2 or 3): ');
    READLN  (macromethod);
    WRITELN;
    REPEAT
    BEGIN {repeat}
        ClrScr;
        WRITE   ('Power function (P) or Exponential ',
                 'function (E): ');
        READLN  (typeoffunction);
        WRITELN;
    END {repeat};
    UNTIL ((typeoffunction = 'P') OR
           (typeoffunction = 'p') OR
           (typeoffunction = 'E') OR
           (typeoffunction = 'e'));
    IF ((typeoffunction = 'P') OR
        (typeoffunction = 'p')) THEN
    BEGIN {if}
        WRITE   ('Constant (1/n) for power function: ');
        functiontype := TRUE;
    END {if}
    ELSE
    BEGIN {else}
        WRITE   ('Constant (kappa) for exponential ',
                 'function: ');
        functiontype := FALSE;
    END {else};
    READLN  (alphakappa);
    WRITELN;
    WRITE   ('Fraction that gradient changes from one ',
             'poregroup to the next: ');
    READLN  (fraction);
    WRITELN;
    WRITE   ('Number of poregroups: ');
    READLN  (np);
```

```
WRITELN;
WRITE   ('Initial concentration of water resident in ',
         'the soil: ');
READLN  (initconcentration);
WRITELN;
WRITE   ('Initial moisture content: ');
READLN  (initmoisture);
WRITELN;
WRITE   ('Saturated hydraulic conductivity (ks): ');
READLN  (ks);
WRITELN;
WRITE   ('Residual moisture content: ');
READLN  (thetar);
WRITELN;
WRITE   ('Saturated moisture content: ');
READLN  (thetas);
WRITELN;
IF macromethod = 2 THEN
BEGIN {if}
    WRITE   ('Satuated hydraulic conductivity: ');
    READLN  (kf);
    WRITELN;
    WRITE   ('Satuated moisture content: ');
    READLN  (thetaf);
    WRITELN;
END {if}
ELSE IF macromethod = 3 THEN
BEGIN {else if}
    WRITE   ('Extra moisture content due to macro-',
             'pore: ');
    READLN  (thetam);
    WRITELN;
END {else if};
WRITE   ('Length of column or profile: ');
READLN  (columnlength);
WRITELN;
WRITE   ('Length of timestep: ');
READLN  (realtimestep);
WRITELN;
WRITE   ('Total length of experiment: ');
READLN  (lengthofexperiment);
WRITELN;
REPEAT
BEGIN {repeat}
    ClrScr;
    WRITE   ('Equal amount of mixing per poregroup (Y/N): ');
    READLN  (mixdecision);
    WRITELN;
END {repeat};
UNTIL ((mixdecision = 'Y') OR
       (mixdecision = 'y') OR
       (mixdecision = 'N') OR
       (mixdecision = 'n'));
IF ((mixdecision = 'Y') OR
    (mixdecision = 'y')) THEN
BEGIN {if}
    WRITE   ('Amount of mixing per poregroup (0 - 1): ');
    READLN  (alpha[1]);
    WRITELN;
    FOR p := 1 TO np DO
        alpha[p] := alpha[1];
END {if}
ELSE
BEGIN {else}
```

```
        FOR p := 1 TO np DO
        BEGIN {for}
            WRITE   ('Amount of mixing for poregroup ',p:1,
                     ' (0 - 1): ');
            READLN  (alpha[p]);
            WRITELN;
        END {for};
    END {else};
END {screenin1};


(*********************************************************)
(* Procedure: PowerFunction1                             *)
(* Purpose  : Piece-wise linear approximation of the hy-   *)
(*            draulic conductivity curve as represented by *)
(*            a power function                           *)
(* Interface: k          - var                           *)
(*            m          - var                           *)
(*            ks                                         *)
(*            thetar                                     *)
(*            thetas                                     *)
(*            alphakappa                                 *)
(*            fraction                                   *)
(*            np                                         *)
(*********************************************************)


PROCEDURE PowerFunction1(VAR k      : CutOffPointsP;
                         VAR m      : CutOffPoints;
                             ks,
                             thetar,
                             thetas : REAL;
                             alphakappa,
                             fraction,
                             np     : LONGINT);

VAR
    alpha,                   { power (1/n)                 }
    imd1,                    { intermediate result in cal-  }
    imd2,                    { culations                   }
    imd3
                  : REAL;
    p                        { counter for poregroups      }
                  : LONGINT;

BEGIN {powerfunction1}
    alpha := alphakappa;
    m[np] := 1;
    k[np] := 1;
    imd1  := LN(fraction);
    imd2  := (1 - alpha)/(alpha * (1 - fraction)) *
             (EXP(alpha/(alpha - 1) * imd1) - 1);
    imd3  := (1 - alpha)/(1 - fraction) *
             (EXP(alpha/(alpha - 1) * imd1) - fraction);
    FOR p := 2 TO np-1 DO
    BEGIN (*for*)
        m[p] := imd2 * EXP((p - np)/(alpha - 1) * imd1);
        k[p] := imd3 * EXP((p - np) * alpha/(alpha - 1) *
                imd1);
    END (*for*);
    m[1] := (alpha - 1)/alpha *
            EXP((2 - np)/(alpha - 1) * imd1);
    k[1] := 0;
    m[0] := 0;
```

```
    k[0] := 0;
    FOR p := 2 TO np DO
        k[p] := ks * k[p];
    FOR p := 0 TO np DO
        m[p] := (thetas - thetar) * m[p] + thetar;
END {powerfunction1};


(*********************************************************)
(* Procedure: PowerFunction2                             *)
(* Purpose  : Piece-wise linear approximation of the hy-   *)
(*            draulic conductivity curve as represented by *)
(*            a power function, where a macro-pore is in-  *)
(*            cluded within the measured conductivity range*)
(* Interface: k          - var                           *)
(*            m          - var                           *)
(*            kf                                         *)
(*            ks                                         *)
(*            thetaf                                     *)
(*            thetar                                     *)
(*            thetas                                     *)
(*            alphakappa                                 *)
(*            fraction                                   *)
(*            np                                         *)
(*********************************************************)


PROCEDURE PowerFunction2(VAR k      : CutOffPointsP;
                         VAR m      : CutOffPoints;
                             kf,
                             ks,
                             thetaf,
                             thetar,
                             thetas : REAL;
                             alphakappa,
                             fraction,
                             np     : LONGINT);

VAR
    alpha,                   { power (1/n)                  }
    imd0,                    { intermediate result in cal-  }
    imd1,                    { culations                    }
    imd2,
    imd3,
    imd4,
    imd5,
    imd6,
    ms,                      { scaled saturated moisture    }
                             { content                      }
    q                        { intermediate result in cal-  }
                             { culating n                   }
                  : REAL;
    n,                       { number that shows the rela-  }
                             { tion between the gradients   }
                             { of the fastest and the next  }
                             { to fastest poregroups        }
    p                        { counter for poregroups       }
                  : LONGINT;


BEGIN {powerfunction2}
    alpha := alphakappa;
    ks    := ks/kf;
    ms    := thetas/thetaf;
```

```
    k[np] := ks;
    m[np] := ms;
    q     := (ks - 1)/((ms - 1) * alpha * fraction);
    IF q < (1/fraction) THEN
    BEGIN {if}
        WRITELN;
        WRITELN ('No solution possible for this combination',
                 ' of input parameters');
    END {if};
    n     := TRUNC(q);
    IF n <> q THEN
        n := n + 1;
    imd0  := LN(fraction);
    imd1  := EXP(alpha/(alpha - 1)* imd0);
    imd2  := LN((ks-1)/(ms-1));
    imd3  := EXP(-alpha/(alpha - 1) * LN(alpha));
    imd4  := (1 - alpha)/(1 - fraction) * (imd1 - 1) *
             EXP(1/(alpha - 1) * imd2) * imd3;
    imd5  := (1 - alpha) * ((imd1 - 1)/(1 - fraction) + 1) *
             EXP(alpha/(alpha - 1) * imd2);
    FOR p := 2 TO np-2 DO
    BEGIN (*for*)
        m[p] := imd4 * EXP(-(n + np - (p + 1))/(alpha - 1)
                * imd0);
        imd6 := 1/(alpha * EXP((n + np - (p + 1)) * imd0));
        k[p] := imd5 * EXP(alpha/(alpha - 1) * LN(imd6));
    END (*for*);
    imd4 := EXP(n * imd0);
    imd5 := EXP(alpha/(alpha-1) * (LN(1/(alpha * imd4)) +
            imd2));
    m[np-1] := (ms - ks -(1 - alpha) * (ms - 1) * imd5)/
               ((1/imd4 - 1) * (ks - 1));
    k[np-1] := ((ms - ks)/(ms - 1) - imd4 * (1 - alpha) *
               imd5)/(1 - imd4);
    m[1] := (alpha - 1) * EXP(1/(alpha - 1) * imd2) * imd3 *
            EXP(-(n + np - 3)/(alpha - 1) * imd0);
    k[1] := 0;
    m[0] := 0;
    k[0] := 0;
    FOR p := 2 TO np DO
        k[p] := kf * k[p];
    FOR p := 1 TO np DO
        m[p] := (thetaf - thetar) * m[p] + thetar;
END {powerfunction2};


(*********************************************************)
(* Procedure: PowerFunction3                             *)
(* Purpose  : Piece-wise linear approximation of the hy-   *)
(*            draulic conductivity curve as represented by *)
(*            a power function with an extra macro-pore    *)
(*            added on top of the measured hydraulic con-  *)
(*            ductivity curve                              *)
(* Interface: k          - var                           *)
(*            m          - var                           *)
(*            ks                                         *)
(*            thetam                                     *)
(*            thetar                                     *)
(*            thetas                                     *)
(*            alphakappa                                 *)
(*            fraction                                   *)
(*            np                                         *)
(*********************************************************)
```

```
PROCEDURE PowerFunction3(VAR k      : CutOffPointsP;
                         VAR m      : CutOffPoints;
                             ks,
                             thetam,
                             thetar,
                             thetas : REAL;
                             alphakappa,
                             fraction,
                             np     : LONGINT);

VAR
    p                        { counter for poregroups       }
                  : LONGINT;

{ The piecewise linear approximation of the conductivity    }
{ curve is calculated using powerfunction1.  After the      }
{ values for 0 Û p Û np-1 have been calculated, one         }
{ poregroup is added whose gradient is f times as steep as  }
{ that of the previous poregroup.  So:                      }
{                                                           }
{   m(np) = m(np-1) + thetam (or thetas + thetam)           }
{                                                           }
{               k(np-1)-k(np-2)                             }
{   k(np) = f * --------------- * (m(np)-m(np-1)) + k(np-1) }
{               m(np-1)-m(np-2)                             }

BEGIN {powerfunction3}
    p := np - 1;
    PowerFunction1 (k,m,ks,thetar,thetas,alphakappa,
                    fraction,p);
    m[np] := m[np-1] + thetam;
    k[np] := fraction * (k[np-1] - k[np-2])/(m[np-1] -
             m[np-2]) * (m[np] - m[np-1]) + k[np-1];
END   {powerfunction3};


(*********************************************************)
(* Procedure: ExponentialFunction                        *)
(* Purpose  : Piece-wise linear approximation of the hy-   *)
(*            draulic conductivity curve as represented by *)
(*            an exponential function                    *)
(* Interface: k          - var                           *)
(*            m          - var                           *)
(*            ks                                         *)
(*            thetar                                     *)
(*            thetas                                     *)
(*            alphakappa                                 *)
(*            fraction                                   *)
(*            np                                         *)
(*********************************************************)


PROCEDURE ExponentialFunction(VAR k      : CutOffPointsP;
                              VAR m      : CutOffPoints;
                                  ks,
                                  thetar,
                                  thetas : REAL;
                                  alphakappa,
                                  fraction,
                                  np     : LONGINT);

VAR
    imd1,                    { intermediate result in cal-  }
```

```
    imd2,                       { culations                   }
    kappa                       { constant for exponential    }
                    : REAL;
    p                           { counter for poregroups      }
                    : LONGINT;

{ The hydraulic conductivity curve is approximated by         }
{                                                             }
{              kappa*(m-1)    -kappa                          }
{             e            - e                              }
{    k(m) = ----------------------                            }
{                        -kappa                               }
{                  1 - e                                      }
{ where                                                       }
{    k(m) = normalized conductivity                           }
{         = K( )/Ks                                           }
{       m = reduced moisture content                          }
{         =                                                   }
{   kappa = constant for exponential                          }
{                                                             }
{ 'fraction' is the fraction that the gradient of the con-    }
{ ductivity curve is increased from one poregroup to the      }
{ next. Then the limiting values for the tangent lines are    }
{                                                             }
{                                                 f           }
{         (p - np) * ln(f) + kappa - 1 - ------- * ln(f)      }
{                                           (1 - f)           }
{  m(p) = ------------------------------------------------     }
{                            kappa                            }
{                                                             }
{          (p-np+1)   ln(f)                                   }
{         f          * -------                                }
{                       (f - 1)                               }
{  k(p) = -------------------                                 }
{                  -kappa                                     }
{              1 - e                                          }
{                                                             }
{ where                                                       }
{       p = 2, ...., np-1                                     }
{       f = 'fraction'                                        }
{                                                             }
{ In addition:                                                }
{                                                             }
{    k(0)  = 0, m(0)  = 0                                     }
{                                                      -kappa }
{                                                     e       }
{                       -1 + (2-np)*ln(f) + kappa + -------   }
{                                                      (2-np) }
{                                                     f       }
{    k(1)  = 0, m(1)  = ----------------------------------    }
{                                    kappa                    }
{    k(np) = 1, m(np) = 1                                     }
{                                                             }
{ Finally the real hydraaulic conductivities as calculated:   }
{                                                             }
{    k(p) := ks * k(p)                                        }
{                                                             }
{ and the real moisture contents as:                          }
{                                                             }
{    m(p) := (thetas - thetar) * m(p) + thetar                }

BEGIN {exponentialfunction}
    kappa := alphakappa;
    m[np] := 1;
```

```
    k[np] := 1;
    imd1  := LN(fraction);
    imd2  := EXP(-kappa);
    FOR p := 2 TO np-1 DO
    BEGIN {for}
        m[p] := ((p - np) * imd1 + kappa - 1 - fraction/
                (1 - fraction) * imd1)/kappa;
        k[p] := (EXP((p - np + 1) * imd1) * imd1/
                 (fraction - 1) - imd2)/(1 - imd2);
    END {for};
    m[1] := (-1 + (2 - np) * imd1 + kappa +
             imd2/(EXP((2 - np) * imd1)))/kappa;
    k[1] := 0;
    m[0] := 0;
    k[0] := 0;
    FOR p := 2 TO np DO
        k[p] := ks * k[p];
    FOR p := 0 TO np DO
        m[p] := (thetas - thetar) * m[p] + thetar;
END {exponentialfunction};

(*********************************************************)
(* Procedure: Velocity                                   *)
(* Purpose  : Calculate the velocity with which water and  *)
(*            solutes in each poregroup move and the num-  *)
(*            ber of nodes that water and solutes travel   *)
(*            in each large timestep                       *)
(* Interface: k                                            *)
(*            m                                            *)
(*            travelnodes - var                            *)
(*            v           - var                            *)
(*            np                                           *)
(*********************************************************)

PROCEDURE Velocity (    k           : CutOffPointsP;
                        m           : CutOffPoints;
                    VAR travelnodes : TravelDistance;
                    VAR v           : PoreGroupsP;
                        np          : LONGINT);

VAR
    p                       { counter for poregroups       }
                  : LONGINT;

{ The velocity in poregroup p (v[p]) is calculated          }
{ according to:                                             }
{                                                           }
{            k[p] - k[p-1]                                  }
{     v[p] = -------------                                  }
{            m[p] - m[p-1]                                  }
{ where                                                     }
{     k[p] and m[p] are calculated in either PowerFunction  }
{     or ExponentialFunction                                }
{     p = 1, ..., np                                       }
{                                                           }
{ The number of nodes that the water and solutes in each    }
{ poregroup travel is a multiple of the velocity of the     }
{ slowest poregroup (p=2, since poregroup 1 is not moving   }
{ at all).  This multiple depends on the fraction 'f' and   }
{ the macro-pore configuration and can be calculated accor- }
{ ding to:                                                  }
{                   v(p)                                    }
{     travelnodes = ----                                    }
{                   v(2)                                    }
```

```
BEGIN {velocity}
    FOR p := 1 TO np DO
        v[p] := (k[p] - k[p-1])/(m[p] - m[p-1]);
    FOR p := 1 TO np DO
        travelnodes[p] := ROUND(v[p]/v[2]);
END {velocity};


(*********************************************************)
(* Procedure: Nodes                                      *)
(* Purpose  : Calculates the total number of nodes in the  *)
(*            column or profile                          *)
(* Interface: numberofnodes      - var                   *)
(*            columnlength                               *)
(*            timestep                                   *)
(*            lowvelocity                                *)
(*            deltax             - var                   *)
(*********************************************************)


PROCEDURE Nodes(VAR numberofnodes : LONGINT;
                    columnlength,
                    timestep,
                    lowvelocity       : REAL;
                VAR deltax            : REAL);

{ The distance between 2 nodes (deltax), is given by:       }
{                                                           }
{   deltax = lowvelocity * timestep                         }
{ where                                                     }
{   lowvelocity = velocity with which water and solutes in  }
{                 poregroup 1 move                          }
{                                                           }
{ Therefor the water in the poregroup 1, the "slowest"      }
{ poregroup moves a distance deltax during one timestep     }
{ The total number of nodes is calculated by:               }
{                                                           }
{                   columnlength                            }
{   numberofnodes = ------------                            }
{                      deltax                               }
{                                                           }

BEGIN {nodes}
    deltax           := lowvelocity * timestep;
    numberofnodes    := ROUND(columnlength/deltax);
END {nodes};
```

```
(*********************************************************)
(* Procedure: ScreenOut1                                 *)
(* Purpose  : Conductivities and moisture content at cut-  *)
(*            off points, and velocities of poregroups are *)
(*            written to the screen                      *)
(* Interface: alpha                                      *)
(*            changeinput       - var                    *)
(*            functiontype                               *)
(*            initconcentration                          *)
(*            initmoisture                               *)
(*            kf                                         *)
(*            thetaf                                     *)
(*            k                 - output                 *)
(*            m                 - output                 *)
(*            v                 - output                 *)
(*            alphakappa        - output                 *)
(*            macromethos                                *)
(*            np                                         *)
(*            numberofnodes     - output                 *)
(*            numberoftimesteps - output                 *)
(*            outfile1          - var                    *)
(*********************************************************)


PROCEDURE ScreenOut1 (VAR alpha             : PoreGroups;
                      VAR changeinput       : CHAR;
                          functiontype      : BOOLEAN;
                          initconcentration,
                          initmoisture,
                          kf,
                          thetaf            : REAL;
                          k                 : CutOffPointsP;
                          m                 : CutOffPoints;
                          v                 : PoreGroupsP;
                          alphakappa,
                          macromethod,
                          np,
                          numberofnodes     : LONGINT;
                      VAR outfile1          : TEXT);

VAR
    p                       { counter for poregroups       }
                  : LONGINT;
    key                     { dummy for KeyPressed         }
                  : CHAR;
```

```
(*******************************************************)
(* Procedure: DrawCurve                                *)
(* Purpose  : draw the power and exponential curve and *)
(*            show the piecewise linear approximation  *)
(*            in the same graph to enable the user to  *)
(*            get an idea of the poregroup configura-  *)
(*            tion                                     *)
(* Interface: functiontype                             *)
(*            k                                        *)
(*            m                                        *)
(*            alphakappa                               *)
(*            macromethod                              *)
(*            np                                       *)
(*            kf                                       *)
(*            thetaf                                   *)
(*******************************************************)


PROCEDURE DrawCurve(functiontype : BOOLEAN;
                    k            : CutOffPointsP;
                    m            : CutOffPoints;
                    alphakappa,
                    macromethod,
                    np           : LONGINT;
                    kf,
                    thetaf       : REAL);

TYPE
    PolyPoints = ARRAY[1..100] OF PointType;

VAR
    curve,                {exponential or power function }
    line                  {piecewise linear approximation}
                : PolyPoints;
    p                     { counter for poregroups       }
                : LONGINT;
    key                   { dummy for KeyPressed         }
                : CHAR;
    imd1                  { intermediate result in cal-  }
                          { culations                    }
                : REAL;
    graphdriver,
    graphmode   : INTEGER;

BEGIN {drawcurve}

    { line for piecewise linear approximation           }

    FOR p := 1 TO np + 1 DO
    BEGIN {for}
        line[p].x := ROUND(640 * (m[p-1] - m[0])/
                           (m[np] - m[0]));
        line[p].y := 480 - ROUND(480 * k[p-1]/k[np]);
    END {for};

    { curve for power function                          }

    IF functiontype = TRUE THEN
    BEGIN {if}

        IF macromethod = 1 THEN
        BEGIN {if}
            FOR p := 2 TO 100 DO
            BEGIN {for}
```

```
            curve[p].x := ROUND(640 * p/100);
            curve[p].y := 480 - ROUND(480 * EXP
                          (alphakappa * LN(p/100)));
        END {for};
    END {if}

    ELSE IF macromethod = 2 THEN
    BEGIN {else if}
        FOR p := 2 TO 100 DO
        BEGIN {for}
            curve[p].x := ROUND(640 * p/100 *
                          (thetaf - m[0])/(m[np] -
                           m[0]));
            curve[p].y := 480 - ROUND(480 * EXP
                          (alphakappa * LN(p/100)) *
                          kf/k[np]);
        END {for};
    END {else if}

    ELSE IF macromethod = 3 THEN
    BEGIN {else if}
        FOR p := 2 TO 100 DO
        BEGIN {for}
            curve[p].x := ROUND(640 * p/100 *
                          (m[np-1] - m[0])/(m[np] -
                          m[0]));
            curve[p].y := 480 - ROUND(480 * k[np-1]
                          /k[np] * EXP(alphakappa *
                          LN(p/100)));
        END {for};
    END {else if};
END {if}

{ curve for exponential function                  }

ELSE
BEGIN {else}
    imd1 := EXP(-alphakappa);
    FOR p := 2 TO 100 DO
    BEGIN {for}
        curve[p].x := ROUND(640 * p/100);
        curve[p].y := 480 - ROUND(480 *
                      (EXP(alphakappa * (p/100-1))
                      - imd1)/(1-imd1));
    END {for};
END {else};
curve[1].x := 0;
curve[1].y := 480;

{ draw line and curve                             }

DetectGraph(graphdriver,graphmode);
InitGraph(graphdriver,graphmode,
          'c:\compiler\tp\bgi');
SetBkColor(15);
SetLineStyle(solidln,0,thickwidth);
SetColor(9);
DrawPoly(100,curve);
SetColor(12);
DrawPoly  (np+1,line);
SetColor(1);
SetTextStyle(SansSerifFont,HorizDir,4);
SetTextJustify(CenterText,CenterText);
OutTextXY (320,25,'HYDRAULIC CONDUCTIVITY CURVES');
```

```
        SetTextStyle(SansSerifFont,HorizDir,2);
        OutTextXY (320,100,'<<Press Key>>');
        SetTextJustify(LeftText,CenterText);
        SetColor(9);
        OutTextXY(50,150,'-Power Function');
        SetColor(12);
        OutTextXY(50,175,'-Piecewise Linear Appr.');
        REPEAT UNTIL KeyPressed;
        key := ReadKey;
        CloseGraph;
    END {drawcurve};


BEGIN {screenout1}
    IF ((m[1] <= 0) OR (m[2] <= m[1]) OR (k[2] < 0)) THEN
    BEGIN {if}
        WRITELN ('The desired combination of poregroups, ',
                 'exponent and gradient change ');
        WRITELN ('can not be realised.  You have to adjust ',
                 'these variables in one of the ');
        WRITELN ('following ways: ');
        WRITELN ('                - decrease number of ',
                 'poregroups');
        WRITELN ('                - increase exponent');
        WRITELN ('                - increase fraction');
        WRITELN ('                - use power function.');
    END {if}
    ELSE
    BEGIN {else}
        WRITELN (outfile1,'1/n = ',alphakappa);
        WRITELN (outfile1,'f   = ',fraction);
        WRITELN (outfile1,'np  = ',np);
        WRITELN (outfile1);
        WRITELN (outfile1);
        ClrScr;
        WRITELN ('  p       m[p]        k[p]       v[p]    ',
                 '  alpha[p]');
        WRITELN ('-----------------------------------------',
                 '-------------');
        WRITELN (outfile1,'  p       m[p]        k[p]      ',
                          ' v[p]      alpha[p]');
        WRITELN (outfile1,'-------------------------------',
                          '----------------------');
        p := 0;
        WRITELN ('        ',m[p]:6:5,'     ',k[p]:7:5);
        WRITELN (outfile1,'        ',m[p]:6:5,'     ',
                        k[p]:7:5);
        FOR p := 1 TO np DO
        BEGIN {for}
            IF p MOD 9 = 0 THEN
            BEGIN {if}
                GoToXY  (65,22);
                WRITE   ('<<Press Key>>');
                REPEAT UNTIL KeyPressed;
                key := ReadKey;
                ClrScr;
                WRITELN ('  p       m[p]        k[p]       ',
                         'v[p]      alpha[p]');
                WRITELN ('-------------------------------',
                         '----------------------');
            END {if};
            WRITELN (p:3,'                                 '
                     ,v[p]:8:5,'     ',alpha[p]:6:5);
            WRITELN ('        ',m[p]:6:5,'     ',k[p]:6:5);
```

```
            WRITELN (outfile1,p:3,'                          ',
                                '    ',v[p]:8:5,'     ',
                                alpha[p]:6:5);
            WRITELN (outfile1,'         ',m[p]:6:5,'     ',
                                k[p]:6:5);
        END {for};
        WRITELN;
        WRITELN;
        WRITELN ('Number of nodes in the profile       : ',
                  numberofnodes:1);
        WRITELN (outfile1);
        WRITELN (outfile1);
        WRITELN (outfile1,'Initial concentration of water ',
                          'resident in the soil: ',
                           initconcentration:6:5);
        WRITELN (outfile1,'Initial moisture content: ',
                           initmoisture:6:5);
        WRITELN (outfile1,'Number of nodes in the profile  ',
                          '     : ',numberofnodes:1);
        GoToXY  (60,22);
        WRITE   ('<<Press Key>>');
        REPEAT UNTIL KeyPressed;
        key := ReadKey;
        GoToXY  (30,22);
        WRITE   ('<<One Moment, Please>>');
        DrawCurve (functiontype,k,m,alphakappa,macromethod,
                   np,kf,thetaf);
    END {else};
    WRITELN;
    WRITELN (outfile1);
    WRITE   ('Do you want to change your input? Y/N ');
    READLN  (changeinput);
    WRITELN;
END {screenout1};
```

```
BEGIN {findporegroups}
    changeinput := 'Y';
    WHILE ((changeinput = 'Y') OR (changeinput = 'y')) DO
    BEGIN {while}
        REWRITE (outfile1);
        ScreenIn1 (alpha,alphakappa,fraction,macromethod,np,
                   columnlength,initconcentration,initmoisture,
                   kf,ks,lengthofexperiment,thetaf,thetam,thetar,
                   thetas,timestep,functiontype);
        IF functiontype = TRUE THEN
        BEGIN {if}
            IF macromethod = 1 THEN
                PowerFunction1 (k,m,ks,thetar,thetas,alphakappa,
                                fraction,np)
            ELSE IF macromethod = 2 THEN
                Powerfunction2 (k,m,kf,ks,thetaf,thetar,thetas,
                                alphakappa,fraction,np)
            ELSE IF macromethod = 3 THEN
                Powerfunction3 (k,m,ks,thetam,thetar,thetas,
                                alphakappa,fraction,np)
        END {if}
        ELSE
            ExponentialFunction (k,m,ks,thetar,thetas,alphakappa,
                                 fraction,np);
        Velocity (k,m,travelnodes,v,np);
        Nodes (numberofnodes,columnlength,timestep,v[2],deltax);
        ScreenOut1 (alpha,changeinput,functiontype,
                    initconcentration,initmoisture,kf,thetaf,k,m,
                    v,alphakappa,macromethod,np,numberofnodes,
                    outfile1);
    END {while};
END {findporegroups};
```

```
(*************************************************************)
(* Procedure: ReadWeather                                    *)
(* Purpose  : Read the precipitation, precipitation concentra- *)
(*            tion and evaporation data into a linked list     *)
(* Interface: infile1       - var                            *)
(*            weatheranchor - var                            *)
(*************************************************************)

PROCEDURE ReadWeather(VAR infile1       : TEXT;
                      VAR weatheranchor : ClimatePtr);

VAR
    bead,                    { new element of linked list       }
    tail                     { last element of linked list      }
                  : ClimatePtr;

BEGIN {readweather}
    RESET(infile1);
    weatheranchor := NIL;
    WHILE NOT EOF(infile1) DO
    BEGIN {while}
        NEW(bead);
        bead^.nxt := NIL;
        READLN (infile1,bead^.realtime,bead^.rainfall,
                        bead^.concent,bead^.realevap);
        bead^.nxt := NIL;
        IF weatheranchor = NIL THEN
            weatheranchor := bead
        ELSE
            tail^.nxt := bead;
        tail := bead;
    END {while};
END {readweather};
```

```
(*************************************************************)
(* Procedure: InitialConditions                              *)
(* Purpose  : Initialize the watermatrix and solutematrix in *)
(*            accordance with the initial conditions         *)
(* Interface: m                                              *)
(*            alphakappa                                     *)
(*            np                                             *)
(*            numberofnodes                                  *)
(*            initconcentration                              *)
(*            initmoisture                                   *)
(*            alphakappa                                     *)
(*            porematrix    - var                            *)
(*            outfile1      - var                            *)
(*************************************************************)


PROCEDURE InitialConditions(    m             : CutOffPoints;
                                alphakappa,
                                np,
                                numberofnodes : LONGINT;
                                initconcentration,
                                initmoisture
                                              : REAL;
                            VAR porematrix    : TransportMatrix;
                            VAR outfile1      : TEXT);

VAR
    p,                        { counter for poregroups          }
    x                         { counter for nodes               }
                    : LONGINT;
    moisture                  { moisture content at depth z     }
                    : REAL;

{ All poregroups are filled according to an intial moisture     }
{ profile of the form:                                          }
{                                                               }
{                  1/alphakappa                                 }
{             m = z                                             }
{   where                                                       }
{             m = moisture content at depth z                   }
{             z = normalized depth (x/L)                        }

BEGIN {initialconditions}
    WRITELN (outfile1,'0');
    FOR x := 1 TO numberofnodes DO
    BEGIN {for}
        IF initmoisture > m[np] THEN
            initmoisture := m[np];
        IF x = 1 THEN
            moisture := initmoisture
                        {m[0]}
        ELSE
            moisture := initmoisture;
                        {EXP(1/alphakappa *}
                        {LN((x-1)/(numberofnodes-1))) *}
                        {(m[np] - m[0]) + m[0];}
        WRITELN (outfile1,moisture);
        FOR p := 1 TO np DO
        BEGIN {for}
            IF moisture > (m[p] - m[p-1]) THEN
            BEGIN {if}
                porematrix[p,x].water := m[p] - m[p-1];
                moisture := moisture - porematrix[p,x].water;
            END {if}
```

```
            ELSE
            BEGIN {else}
                prematrix[p,x].water := moisture;
                moisture := 0;
            END {else};
            prematrix[p,x].solute := initconcentration *
                                     prematrix[p,x].water;
        END {for};
    END {for};
    WRITELN (outfile1);
END {initialconditions};


(*************************************************************)
(* Procedure: MoveAndMix                                     *)
(* Purpose  : Simulate the movement of water during one      *)
(*            large timestep                                 *)
(* Interface: alpha                                          *)
(*            travelnodes                                    *)
(*            fraction                                       *)
(*            np                                             *)
(*            numberofnodes                                  *)
(*            deltax                                         *)
(*            time                                           *)
(*            timestep                                       *)
(*            m                                              *)
(*            n             - var                            *)
(*            prematrix     - var                            *)
(*            outfile1      - var                            *)
(*            outfile2      - var                            *)
(*************************************************************)


PROCEDURE MoveAndMix(    alpha         : PoreGroups;
                         travelnodes   : TravelDistance;
                         fraction,
                         n1,
                         np,
                         numberofnodes : LONGINT;
                         deltax,
                         time,
                         timestep      : REAL;
                         m             : CutOffPoints;
                     VAR n2            : LONGINT;
                     VAR prematrix     : TransportMatrix;
                     VAR outfile1,
                         outfile2      : TEXT;
                     VAR weatheranchor : ClimatePtr);
```

```
TYPE
    PoreGroupsP    = ARRAY[1..MaxPoreGroups] OF REAL;
    NodeColumn     = ARRAY[1..MaxNodes]      OF Mass;

VAR
    evapcolumn              { column containing solute amounts }
                            { and moisture taken out of the    }
                            { poregroups due to evaporation    }
                : NodeColumn;
    p,                      { counter for poregroups           }
    smalltstep,             { counter for timesteps during     }
                            { which the fastest poregroup      }
                            { moves one node                   }
    x                       { counter for nodes                }
                : LONGINT;
    key                     { dummy for KeyPressed             }
                : CHAR;
    evapamount,             { total amount of actual evapora-  }
                            { tion during one timestep         }
    moisture,               { moisture content of a node       }
    outflowsolute,          { amount of solute flowing out of  }
                            { the soil in one timestep         }
    outflowwater,           { amount of water flowing out of   }
                            { the soil in one timestep         }
    precipconcentration,    { solute concentration of the pre- }
                            { cipitation                       }
    rainfallamount          { amount of rainfall               }
                : REAL;
    topboundarys,           { amount of solutes put into each  }
                            { topnode                          }
    topboundaryw            { amount of rainfall put into each }
                            { topnode                          }
                : PoreGroupsP;

    (*********************************************************)
    (* Procedure: InterpolateWeather                         *)
    (* Purpose  : Determine the amount of precipitation and the*)
    (*            concentration of that precipitation during   *)
    (*            one large timestep                           *)
    (* Interface:                                            *)
    (*********************************************************)

    PROCEDURE InterpolateWeather(VAR evapamount,
                                     precipconcentration,
                                     rainfallamount,
                                     time,
                                     timestep      : REAL;
                                 VAR weatheranchor : ClimatePtr);

    VAR
        portion,                { amount of rainfall or evapo- }
                                { ration occurring before the  }
                                { next value in the linked     }
                                { list that actually occurs in }
                                { one timestep                 }
        imd1,                   { intermediate results in cal- }
        imd2                    { culations                    }
                        : REAL;

    BEGIN {interpolateweather};
        IF time = weatheranchor^.realtime THEN
        BEGIN {if}
            precipconcentration := weatheranchor^.concent;
            rainfallamount      := weatheranchor^.rainfall;
```

```
        evapamount          := weatheranchor^.realevap;
    END {if}
    ELSE IF time < weatheranchor^.realtime THEN
    BEGIN {else if}
        precipconcentration := weatheranchor^.concent;
        portion             := timestep/(weatheranchor^.
                               realtime-(time-timestep));
        rainfallamount      := portion *
                               weatheranchor^.rainfall;
        evapamount          := portion *
                               weatheranchor^.realevap;
        weatheranchor^.rainfall := weatheranchor^.rainfall -
                                   rainfallamount;
        weatheranchor^.realevap := weatheranchor^.realevap -
                                   evapamount;
    END {else if}
    ELSE
    BEGIN {else}
        rainfallamount      := weatheranchor^.rainfall;
        evapamount          := weatheranchor^.realevap;
        precipconcentration := weatheranchor^.concent *
                               rainfallamount;
        portion := (time - weatheranchor^.realtime)/
                   (weatheranchor^.nxt^.realtime -
                    weatheranchor^.realtime);
        weatheranchor       := weatheranchor^.nxt;
        imd1                := portion *
                               weatheranchor^.rainfall;
        rainfallamount      := rainfallamount + imd1;
        imd2                := portion *
                               weatheranchor^.realevap;
        evapamount          := evapamount + imd2;
        precipconcentration := (precipconcentration +
                                imd1 * weatheranchor^.
                                concent)/rainfallamount;
        weatheranchor^.rainfall := weatheranchor^.rainfall -
                                   imd1;
        weatheranchor^.realevap := weatheranchor^.realevap -
                                   imd2;
    END {else};
END {interpolateweather};
```

```
(*********************************************************)
(* Procedure: Precipitation                              *)
(* Purpose  : Calculates how much water is put into each *)
(*            topnode of each poregroup, when that one be- *)
(*            comes vacant due to movement. The water is *)
(*            put in according to the topboundary condi- *)
(*            tion, e.g. rainfall or irrigation          *)
(* Interface: deltax                                     *)
(*            precipconcentration                        *)
(*            rainfallamount                             *)
(*            m                                          *)
(*            fraction                                   *)
(*            np                                         *)
(*            travelnodes                                *)
(*            topboundarys - var                         *)
(*            topboundaryw - var                         *)
(*********************************************************)

PROCEDURE Precipitation(    deltax,
                            precipconcentration,
                            rainfallamount : REAL;
                            m              : CutOffPoints;
                            fraction,
                            np             : LONGINT;
                            travelnodes    : TravelDistance;
                        VAR topboundarys,
                            topboundaryw   : PoreGroupsP);

VAR
    emptynodes,              { number of nodes in each pore-}
                             { group, that have been emp-   }
                             { tied during the previous move}
    p                        { counter for poregroups       }
                  : LONGINT;
    availablespace,          { space available in each pore-}
                             { group per timestep           }
    runoff                   { amount of not infiltrated    }
                             { water (no ponding)           }
                  : REAL;


BEGIN {precipitation}


    { initialize topboundaryw and topboundarys             }

    FOR p := 1 TO np DO
    BEGIN {for}
        topboundaryw[p] := 0;
        topboundarys[p] := 0;
    END {for};

    { determine how much rainfall can be absorbed by each  }
    { poregroup during a timestep and next calculate how   }
    { that amount will be distributed in time. This amount }
    { (the amount absorbed by one poregroup during one     }
    { smalltstep) is stored in the array topboundaryw.     }
    { The solutes added to each porgeroup in a small time- }
    { step are calculated by multiplying the concentration }
    { of the rainfall with the rainfall added during a     }
    { small timestep.                                      }

    p := 2;
```

```
    WHILE (rainfallamount > 0) AND (p <= np) DO
    BEGIN {while}
        emptynodes := travelnodes[p];
        availablespace := (m[p] - m[p-1]) * emptynodes *
                          deltax;
        IF rainfallamount > availablespace THEN
        BEGIN {if}
            topboundaryw[p] := m[p] - m[p-1];
            rainfallamount  := rainfallamount -
                               availablespace;
        END {if}
        ELSE IF rainfallamount > 0 THEN
        BEGIN {else if}
            topboundaryw[p] := rainfallamount/
                               (deltax * emptynodes);
            rainfallamount := 0;
        END {else if};
        topboundarys[p] := precipconcentration *
                           topboundaryw[p];
        p := p + 1;
    END {while};
    runoff := 0;
    IF rainfallamount > 0 THEN
    BEGIN {if}
        runoff := rainfallamount;
        WRITELN ('runoff is ',runoff:3:8,'.');
    END {if}
END {precipitation};
```

```
(*********************************************************)
(* Procedure: Move                                       *)
(* Purpose  : Simulate the transport of a substance during *)
(*            one timestep                               *)
(* Interface: fraction                                   *)
(*            np                                         *)
(*            numberofnodes                              *)
(*            smalltstep                                 *)
(*            deltax                                     *)
(*            outflowsolute - var                        *)
(*            outflowwater  - var                        *)
(*            porematrix    - var                        *)
(*            topboundarys                               *)
(*            topboundaryw                               *)
(*            travelnodes                                *)
(*            outfile2      - var                        *)
(*********************************************************)


PROCEDURE Move(    fraction,
                   np,
                   numberofnodes,
                   smalltstep    : LONGINT;
                   deltax        : REAL;
               VAR outflowsolute,
                   outflowwater  : REAL;
               VAR porematrix    : TransportMatrix;
                   topboundarys,
                   topboundaryw  : PoreGroupsP;
                   travelnodes   : TravelDistance;
               VAR outfile2      : TEXT);

VAR
    p,                        { counter for poregroups       }
    x                         { counter for nodes            }
                  : LONGINT;

{ During one timestep, the water and the solutes in the     }
{ different poregroups move a distance according to:        }
{                                                           }
{                            p - 2                          }
{   numberofnodes = fraction                                }
{                                          np - 2           }
{ When we divide one timestep into fraction       small     }
{ timesteps (smalltstep) then the fastest poregroup moves   }
{ one node every timestep. The other poregroups move only   }
{               np - p                                      }
{ every fraction       timesteps.                           }
{ The topnodes that are emptied due to the movement are     }
{ filled with according to the topboundaryw as calculated   }
{ in precipation procedure                                  }
{ The outflow is calculated as the sum of the moisture con- }
{ tents of the bottomnodes that are moved beyond the bottom }
{ boundary, during one large timestep.                      }
```

```
BEGIN {move}
    FOR p := 2 TO np DO
    BEGIN {for}
        IF smalltstep MOD (travelnodes[np] DIV
                           travelnodes[p]) = 0 THEN
        BEGIN {if}
            outflowwater  := outflowwater +
                             porematrix[p,numberofnodes].
                             water * deltax;
            outflowsolute := outflowsolute +
                             porematrix[p,numberofnodes].
                             solute * deltax;
            FOR x := numberofnodes DOWNTO 2 DO
                porematrix[p,x] := porematrix[p,x-1];
            porematrix[p,1].water  := topboundaryw[p];
            porematrix[p,1].solute := topboundarys[p];
        END {if};
    END {for};
END {move};


(*********************************************************)
(* Procedure: ActualEvaporation                          *)
(* Purpose  : Empty nodes in the rootzone according to the *)
(*            actual evapotranspiration                  *)
(* Interface: deltax                                     *)
(*            evapamount                                 *)
(*            fraction                                   *)
(*            np                                         *)
(*            numberofnodes                              *)
(*            travelnodes                                *)
(*            evapcolumn   - var                         *)
(*            porematrix   - var                         *)
(*********************************************************)

PROCEDURE ActualEvaporation(   deltax,
                               evapamount : REAL;
                               fraction,
                               np,
                               numberofnodes
                                          : LONGINT;
                               travelnodes: TravelDistance;
                           VAR evapcolumn : NodeColumn;
                           VAR porematrix : TransportMatrix);

VAR
    p,                        { counter for poregroups       }
    rootnodes,                { number of nodes in the root- }
                              { zone                         }
    x                         { counter for nodes            }
                  : LONGINT;

    extrasolute,              { amount of solute in the wa-  }
                              { ter that has evaporated in   }
                              { one poregroup                }
    nodeevaporation,          { moisture content evaporated  }
                              { one node during one timestep }
    moisture,                 { total moisture content in    }
                              { one node                     }
    rootmoisture,             { total moisture content of    }
                              { the rootzone                 }
    rootzone                  { depth of rootzone            }
                  : REAL;
```

```
BEGIN {actualevaporation}

    rootzone := 0;

    { evaporation occuring during one small timestep        }

    evapamount := evapamount/travelnodes[np];

    { Calculate number of nodes in the rootzone             }

    rootnodes := ROUND(rootzone/deltax);
    IF rootnodes = 0 THEN
        rootnodes := 1;

    { Set array containing solute amounts that are taken    }
    { out of the poregroups as a result of evaporation and  }
    { array containing moisture evaporated per node to zero }

    FOR x := 1 TO rootnodes DO
    BEGIN {for}
        evapcolumn[x].solute := 0;
        evapcolumn[x].water  := 0;
    END {for};

    { Calculate total amount of moisture present in the     }
    { rootzone                                              }

    rootmoisture := 0;
    FOR x := 1 TO rootnodes DO
        FOR p := 1 TO np DO
            rootmoisture := rootmoisture +
                            porematrix[p,x].water;

    { Calculate amount of actual evaporation attributed to  }
    { each node                                             }

    FOR x := 1 TO rootnodes DO
    BEGIN {for}
        moisture := 0;
        FOR p := 1 TO np DO
            moisture := moisture + porematrix[p,x].water;
        nodeevaporation     := evapamount * moisture/
                               (rootmoisture * deltax);
        evapcolumn[x].water := nodeevaporation;

        { Take water out of the largest pores first         }

        p := np;
```

```
        WHILE (nodeevaporation > 0) AND (p >= 1) DO
        BEGIN {while}
            porematrix[p,x].water := porematrix[p,x].water -
                                     nodeevaporation;
            IF porematrix[p,x].water < 0 THEN
            BEGIN {if}
                nodeevaporation := -porematrix[p,x].water;
                porematrix[p,x].water := 0;
                evapcolumn[x].solute := evapcolumn[x].solute
                                      + porematrix[p,x].solute;
                porematrix[p,x].solute := 0;
                p := p - 1;
            END {if}
            ELSE
            BEGIN {else}
                extrasolute := nodeevaporation/
                    (porematrix[p,x].water + nodeevaporation)
                    * porematrix[p,x].solute;
                evapcolumn[x].solute := evapcolumn[x].solute
                                        + extrasolute;
                porematrix[p,x].solute := porematrix[p,x].
                                        solute - extrasolute;
                nodeevaporation := 0;
            END {else};
        END {while};
    END {for};
END {actualevaporation};


(*********************************************************)
(* Procedure: RedistributeWater                          *)
(* Purpose  : Redistribute the water in the different    *)
(*            poregroups according to a coefficient eta[p] *)
(*            and the empty porespace in each poregroup. *)
(*            Different redistribution strategies can be *)
(*            implemented by changing the parameter eta[p].*)
(* Interface: alpha                                      *)
(*            m                                          *)
(*            np                                         *)
(*            numberofnodes                              *)
(*            porematrix    - var                        *)
(*********************************************************)

PROCEDURE RedistributeWater(    alpha         : PoreGroups;
                                m             : CutOffPoints;
                                np,
                                numberofnodes : LONGINT;
                            VAR porematrix
                                          : TransportMatrix);
```

```
VAR
    p,                          { counter for poregroups       }
    x                           { counter for nodes            }
                    : LONGINT;
    sumalpha,                   { check for mixing-coefficients}
    totalspace,                 { total empty pore space       }
    waterpool                   { water to be redistributed    }
                    : REAL;
    eta,                        { part of waterpool to be put  }
                                { back into poregroup[p]       }
    emptyspace                  { empty poresapce in pore-     }
                                { group[p]                     }
                    : PoreGroups;

{ Redistribution of water is implemented in the following   }
{ way:  - Out of each poregroup an amount eta[p] *          }
{         moisture content (m.c.) is taken and put into a   }
{         common pool                                       }
{       - Next water from the common pool is put back into  }
{         the different poregroups relative to the amount   }
{         of empty porespace available in each poregroup    }
{ This proces can be summarized with the following equa-    }
{ tions:                                                    }
{              p=np                                         }
{ waterpool = SUM(alpha[p] * m.c.[p])                       }
{             p=1                                           }
{                                                           }
{ emptyspace[p] = (m[p] - m[p-1]) - (1 - alpha[p]) * m.c.[p]}
{                                                           }
{               p=np                                        }
{ totalspace = SUM(emptyspace[p])                           }
{              p=1                                          }
{                                                           }
{ m.c.[p] := (1-alpha[p]) * m.c.[p] + eta[p] * waterpool    }
{                                                           }
{ where                                                     }
{           emptyspace[p]                                   }
{ eta[p] = -------------                                    }
{            totalspace                                     }

BEGIN {redistributewater}
    sumalpha := 0;
    FOR p := 1 TO np DO
        sumalpha := sumalpha + alpha[p];
    IF sumalpha > 0 THEN
    BEGIN {if}
        FOR x := 1 TO numberofnodes DO
        BEGIN {for}
            totalspace := 0;
            waterpool  := 0;
            FOR p := 1 TO np DO
            BEGIN {for}
                waterpool     := waterpool + alpha[p] *
                                 porematrix[p,x].water;
                emptyspace[p] := (m[p] - m[p-1]) -
                                 (1 - alpha[p]) *
                                 porematrix[p,x].water;
                totalspace    := totalspace + emptyspace[p];
            END {for};
            FOR p := 1 TO np DO
            BEGIN {for}
                eta[p]        := emptyspace[p]/totalspace;
                porematrix[p,x].water := (1 - alpha[p]) *
```

```
                                        porematrix[p,x].water +
                                        eta[p] * waterpool;
            END {for};
        END {for};
    END {if};
END {redistributewater};


(*********************************************************)
(* Procedure: MixSolute                                  *)
(* Purpose  : Redistribute the water in the different    *)
(*            poregroups according to a coefficient eta[p] *)
(*            and the empty porespace in each poregroup. *)
(*            Different redistribution strategies can be *)
(*            implemented by changing the parameter eta[p].*)
(* Interface: alpha,                                     *)
(*            m                                          *)
(*            np                                         *)
(*            numberofnodes                              *)
(*            porematrix    - var                        *)
(*********************************************************)

PROCEDURE MixSolute(    alpha         : PoreGroups;
                        m             : CutOffPoints;
                        np,
                        numberofnodes : LONGINT;
                    VAR porematrix    : TransportMatrix);

VAR
    p,                          { counter for poregroups       }
    x                           { counter for nodes            }
                    : LONGINT;
    concentration,
    corrector,
    sumalpha,                   { check for mixing-coefficients}
    totalspace,                 { total empty pore space       }
    solutepool,                 { solute to be redistributed   }
    waterpool
                    : REAL;
    eta,                        { part of solutepool to be put }
                                { back into poregroup[p]       }
    emptyspace                  { empty poresapce in pore-     }
                                { group[p]                     }
                    : PoreGroups;

{ Redistribution of solute is implemented in the following  }
{ way:  - Out of each poregroup an amount eta[p] *          }
{         solute content (m.c.) is taken and put into a     }
{         common pool                                       }
{       - Next solute from the common pool is put back into }
{         the different poregroups according to a parameter }
{         eta[p]                                            }
{ This proces can be summarized with the following equa-    }
{ tions:                                                    }
{            p=np                                           }
{ solutepool = SUM(alpha[p] * mass[p])                      }
{            p=1                                            }
{                                                           }
{ mass[p] := (1-alpha[p]) * mass[p] + eta[p] * solutepool   }

BEGIN {mixsolute}
    sumalpha := 0;
    FOR p := 1 TO np DO
        sumalpha := sumalpha + alpha[p];
```

```
    FOR x := 1 TO numberofnodes DO
    BEGIN {for}
        IF (sumalpha > 0) OR (evapcolumn[x].water > 0) THEN
        BEGIN {if}
            moisture := 0;
            FOR p := 1 TO np DO
                moisture := moisture + porematrix[p,x].water;
            IF moisture > 0 THEN
            BEGIN {if}
                solutepool := evapcolumn[x].solute;
                waterpool  := evapcolumn[x].water;
                corrector := 1 - evapcolumn[x].water/moisture;
                FOR p := 1 TO np DO
                BEGIN {for}
                    alpha[p]   := alpha[p] * corrector;
                    solutepool := solutepool + alpha[p] *
                                  porematrix[p,x].solute;
                    waterpool  := waterpool  + alpha[p] *
                                  porematrix[p,x].water *
                                  corrector;
                END {for};
                IF waterpool > 0 THEN
                    concentration := solutepool/waterpool
                ELSE
                    concentration := 0;
                corrector     := 1 - corrector;
                FOR p := 1 TO np DO
                porematrix[p,x].solute := (1 - alpha[p]) *
                    porematrix[p,x].solute + (corrector *
                    (1 - alpha[p]) + alpha[p]) *
                    concentration * porematrix[p,x].water;
            END {if};
        END {if};
    END {for};
END {mixsolute};
```

```
BEGIN {moveandmix}
    outflowwater  := 0;
    outflowsolute := 0;

    { Determine amount of precipitation and precipitation con-  }
    { centration and actual evaporation during large timestep   }

    InterpolateWeather (evapamount,precipconcentration,
                        rainfallamount,time,timestep,
                        weatheranchor);

    { Calculate how the nodes at the topboundary will be filled }
    { , according to the top boundary condition, e.g. rainfall  }
    { or irrigation                                             }

    Precipitation(deltax,precipconcentration,rainfallamount,m,
                  fraction,np,travelnodes,topboundarys,
                  topboundaryw);

    { Initialize evapcolumn                                     }

    FOR x := 1 TO numberofnodes DO
    BEGIN {for}
        evapcolumn[x].solute := 0;
        evapcolumn[x].water  := 0;
    END {for};


    FOR smalltstep := 1 TO travelnodes[np] DO
    BEGIN {for}

        { Move water and solute                                 }

        Move (fraction,np,numberofnodes,smalltstep,deltax,
              outflowsolute,outflowwater,porematrix,topboundarys,
              topboundaryw,travelnodes,outfile2);

        { Empty nodes in the rootzone according to the actual   }
        { evapotranspiration                                    }

        ActualEvaporation (deltax,evapamount,fraction,np,
                           numberofnodes,travelnodes,evapcolumn,
                           porematrix);

        { Redistribute the water over the different poregroups  }

{       RedistributeWater (alpha,m,np,numberofnodes,porematrix);}

        { Redistribute the solute over the different poregroups }

        MixSolute (alpha,m,np,numberofnodes,porematrix);

    END {for};
```

```
    WRITELN (outfile2,time,'     ',outflowwater:12:10,
                        '      ',outflowsolute:12:10);

    IF n1 MOD n2 = 0 THEN
    BEGIN {if}
        n2 := n2 * 2;
        WRITELN (outfile1,time);
        FOR x := 1 TO numberofnodes DO
        BEGIN {for}
            moisture := 0;
            FOR p := 1 TO np DO
            BEGIN {for}
                moisture := moisture + porematrix[p,x].water;
            END {for};
            WRITELN (outfile1,moisture);
        END {for};
        WRITELN (outfile1);
    END {if};
END {moveandmix};


(*************************************************************)
(* Procedure: CloseFiles                                     *)
(* Purpose  : Close all files                                *)
(* Interface: infile1  - var                                 *)
(*            outfile1 - var                                 *)
(*            outfile2 - var                                 *)
(*************************************************************)

PROCEDURE CloseFiles(VAR infile1,
                         outfile1,
                         outfile2 : TEXT);

BEGIN {closefiles}
    CLOSE (infile1);
    CLOSE (outfile1);
    CLOSE (outfile2);
END {closefiles};


BEGIN {prefflow}

    ASSIGN  (input,'');
    RESET   (input);
    ASSIGN  (output,'');
    REWRITE (output);

    DetectGraph(graphdriver,graphmode);
    InitGraph(graphdriver,graphmode,'c:\compiler\tp\bgi');
    SetBkColor(15);
    SetLineStyle(solidln,0,thickwidth);
    SetColor(1);
    SetTextStyle(TriplexFont,HorizDir,6);
    SetTextJustify(CenterText,CenterText);
    OutTextXY (320,25,'WELCOME');
    OutTextXY (320,125,'TO THE');
    OutTextXY (320,225,'PREFERENTIAL FLOW');
    OutTextXY (320,325,'MODEL');
    SetTextStyle(SmallFont,HorizDir,6);
    OutTextXY (320,425,'<<Press Key>>');
    SetTextJustify(LeftText,CenterText);
    OutTextXY(50,450,
              'Authors: Bart Nijssen and Tammo Steenhuis');
    REPEAT UNTIL KeyPressed;
```

```
    key := ReadKey;
    CloseGraph;

    { Prompt user for the filenames of the files containing the }
    { input and the output                                      }

    OpenFiles (infile1,outfile1,outfile2);

    { Prompt the user for the necessary input; calculate the    }
    { sizes and travel velocities for the different poregroups, }
    { calculate the number of nodes.                            }

    FindPoreGroups (alpha,travelnodes,alphakappa,fraction,np,
                    numberofnodes,deltax,initconcentration,
                    initmoisture,lengthofexperiment,timestep,m,
                    outfile1);

    { Read the infile containing the data about precipitation   }
    { and evaporation                                           }

    ReadWeather(infile1,weatheranchor);

    { Assign moisture contents and solute amounts to the diffe- }
    { rent porenodes in accordance with the initial conditions. }

    InitialConditions (m,alphakappa,np,numberofnodes,
                       initconcentration,initmoisture,porematrix,
                       outfile1);

    { Move the moisture and solute and afterwards implement the }
    { mixing.                                                   }

    n1   := 1;
    n2   := 1;
    time := timestep;
    WHILE time <= lengthofexperiment DO
    BEGIN {while}
        MoveAndMix (alpha,travelnodes,fraction,n1,np,
                    numberofnodes,deltax,time,timestep,m,n2,
                    porematrix,outfile1,outfile2,weatheranchor);
        WRITELN ('time is ',time:8:2);
        time := time + timestep;
        n1 := n1 + 1;
    END {while};

    { Close all files                                           }

    CloseFiles (infile1,outfile1,outfile2);

END {prefflow}.
```