ZEF Bonn

RUNOFF01

Software for calculating two-dimensional Hortonian flow and associated scale effects, manual and code description

Nick van de Giesen

Center for Development Research

Bonn University

Germany

Tjeerd-Jan Stomph

Department of Plant Sciences

Wageningen University

Netherlands

ZEF Documentation of Research 3/2003

Contents

1 Introduction

The here presented software, RUNOFF01, has evolved over the period 1996-2002. The software development accompanied field experiments in Côte d'Ivoire, West Africa, and laboratory experiments in Wageningen, Netherlands. The goal of all these activities was to understand and quantify scale effects associated with surface runoff. The type of surface runoff under discussion is known as "infiltration-capacity-excess overland" or "Horton" flow. It occurs when rainfall intensity is so high that it exceeds the infiltration capacity of the soil. This type of surface runoff is wide spread throughout the tropics due to the high rainfall intensities of tropical storms. However, although Horton flow can often be observed everywhere in the landscape, it is common that only a fraction of the water on the surface reaches the bottom of a slope. It is this phenomenon that is the focus of these studies. The expectation is that through better understanding of this process, better management of soil and vegetation can be developed to reduce erosion and to improve water use at the watershed level.

The research has been reported upon in the following articles, to which the reader is referred for more detailed background: Van de Giesen *et al.*, 2000; Stomph *et al.*, 2001; Stomph *et al.*, 2002a; Stomph *et al.*, 2002b; and Van de Giesen *et al.*, 2003 (submitted). Full references can be found in the literature section.

The model underlying the presented software has been verified extensively. This documentation serves to bring the modeling software in the public domain and to allow use and evaluation by interested researchers.

2 Overview of available files

Besides this manual, eleven files are available that make up the total "RUNOFF01" package. These files can be divided over two groups. The *first group* contains the compiled (executable) software and auxiliary files:

RUNOFF01.EXE: The executable file that can be run under Windows, detailed instructions can be found in Chapter 3.

VBRUN300.DLL: This file needs to be available on the computer in order to run RUNOFF01.EXE. This version was copied from:
http://www.infoworld.com/pageone/softwarelibrary/pcsftwre/VBRUN300.ZIP

PARAM.DAT: A simple ASCII file containing slope and computing parameters. This file needs to be in the same directory as RUNOFF.EXE, otherwise the program will not run properly.

AUG17.DAT and SEP26B.DAT: Two sample rainfall intensity input files with the correct structure measured in the 1996 rainy season at the WARDA experimental station of M'Bé, near Bouaké, Côte d'Ivoire.

SAMPLE.PRN: Sample output file, produced with the above files. The name of the output file has a maximum length of eight characters, no spaces or other special characters, and is given the extension ".PRN".

The files from this first group are all that is needed to simply run the software as is and to explore the effects that different parameter combinations may have on the runoff process.

The *second group* of files contains Visual Basic 3.0 code in ASCII format and can be used to check and, if needed, update the code:

RUNOFF01.MAK: Project file containing information on modules and frames.

RUNOFF01.FRM: The main/startup window.

HELP.FRM: A simple help window with the main instructions and file structures.

PARAMINP.FRM: The window used for manual change of input parameters and file names.

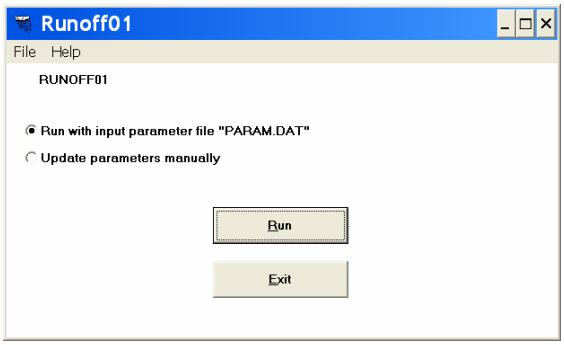
RUNOFF01.BAS: The central calculation module.

The main code is found in the module "RUNOFF01.BAS". This module was originally written for a DOS environment. "RUNOFF01.BAS" forms the core of the simulation calculations, is annotated to facilitate better reading of the code, and can be found in the Appendix of this manual. Chapter 4 of this documentation describes general ideas and algorithms of the code. The remaining windows or frames ("RUNOFF01.FRM", "HELP.FRM", and "PARAMINP.FRM") and the project file ("RUNOFF01.MAK") are not annotated and are only made available to allow easy compilation of code changed by the user with Visual Basic.

3 Using RUNOFF01.EXE

3.1 Standard use under Windows

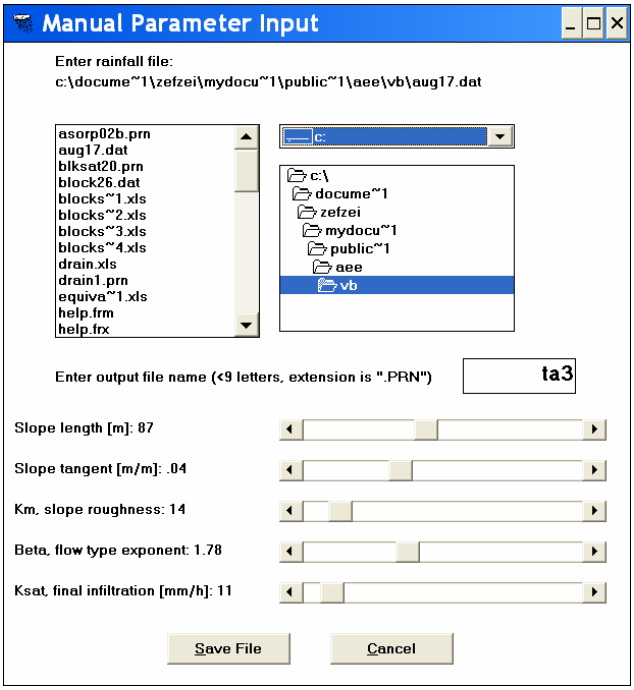
To start RUNOFF01.EXE under Windows, simply double click on the file name or icon. The following opening screen appears:



There are only a few choices available. The two menus “File” and “Help” are straightforward. The “File” menu has as only entry “Exit” for which there is also a shortcut “Ctrl”+”E”. Clicking the “Help” menu opens a simple help window with the main directions and file structures.

The radio buttons allow the choice to run the program with the existing parameters contained in the file “PARAM.DAT” or to first update the parameters manually. If the parameters are correct, simply press the “Run” button and the program calculates and reports point runoff and slope runoff. Once the calculation is finished, a remark to that extent appears in the window.

Clicking on the update button opens a window in which parameters can be changed:



The standard file dialogue at the top allows to select any rainfall intensity input file, the structure of which will be described below. The output file name has a maximum length

of eight characters, no spaces or other special characters are allowed. The program gives the output file always the extension ".PRN" and stores it in the same directory as where RUNOFF01.EXE is located.

The slides allow adjustment of five key parameters within value ranges that have been shown to produce valid results for normal inputs. Slope length can be adjusted from 1 m to 200 m. The slope tangent can be changed from 1% to 10%. Manning's K_m, often referred to via its inverse, Manning's n (K_m=1/n), can range from 5 to 100, with the correct dimensions depending on the value of beta. Beta, the flow type exponent can vary from 1.6 to 2.1, whereby 5/3 is the theoretical minimum for fully developed turbulence. The maximum value would be 3 for completely laminar flow, which is usually not found on natural slopes. The saturated hydraulic conductivity K_{sat} or, formally better, the infiltration rate under ponding for large times, can be varied from 5 to 100 mm/h.

Once the parameters have been adjusted, pushing on the "Save file" button returns the user to the opening window with an updated parameter file. Pushing "Cancel" returns the user to the opening window without changing the original parameter values.

The parameters are always read from and stored in a simple ASCII file called "PARAM.DAT", the structure of which will be described below. "PARAM.DAT" always has to be present in the same directory as RUNOFF01.EXE, otherwise the program will terminate with an error message.

3.2 File structures

All files used and produced by RUNOFF01 are in ASCII format and can be read with any text editor. There are two input files, "PARAM.DAT" and a rainfall intensity file without a fixed name, and one output file that is given the ".PRN" extension.

PARAM.DAT

This file consists of ten lines:
Line 1: Path and name of rainfall file ('file.ext', see structure below)
Line 2: Name of output file containing results
Line 3: Length of slope [m]
Line 4: Beta, Flow type exponent (tested values: 1.66 - 2.1) [-]
Line 5: S_{in}, Initial sorptivity *10^6 (tested values: 60-200) [m/s^(-0.5)]
Line 6: K_{sat}, final infiltration rate (tested values: 5-100) [mm/hr]
Line 7: K_m, Manning's roughness coefficient, 1/n, (tested values: 5-100) [m^(2-beta)/s]
Line 8: dt, Time step, (tested value: 1) [s]
Line 9: tail, maximum ponding time after rain (tested values: <4000) [s]
Line 10: Slope, tangent of slope (tested values 0.01/0.10) [m/m]

Lines 5, 8 and 9 are not available for manual adjustment during run time but can be adjusted off-line by editing this file. The sorptivity can have an important impact in the early stages of a rainstorm but is difficult to quantify without detailed tests. The time step, dt, seems to work well with a value of 1 s but there may reason to experiment. The variable "tail" is rather technical as it helps to prevent that the program runs for a very long time under certain parameter combinations by setting a maximum time during which runoff is calculated after the rain stops.

A valid sample of PARAM.DAT would be:

```
c:\docume~1\zefzei\mydocu~1\public~1\aee\vb\aug17.dat
ta3.PRN
 87
 1.78
 200
 11
 14
 1
 3000
 .04
```

Path and file names follow old DOS and Windows 3.1 conventions.

Rainfall intensity data

The rainfall intensity file can have any name and can be placed anywhere. If the path is not correct when the "Run" button is pushed, RUNOFF01 will simply terminate with an error message. The structure of this ASCII file fits data collected with a tipping bucket rain gage:

The first two lines are not used by the program and can be used for user comments and descriptions. The third line gives the amount of rainfall for one tip of the tipping bucket in mm. The following lines contain the tipping times, usually starting with zero. A valid beginning of a rainfall file would be as AUG17.DAT:

```
Rainfall measured on 17 August 1996, M'Bé, B2N, start 7h47

0.3
0
22
34
42
52
58
68
```

The file will simply be read until the end and the total rainfall is equal to the number of lines (tips) times the amount per tip.

Output file

Through the file PARAM.DAT, it is possible to give any valid name to the output file including a path. It is recommended, however, to simply let RUNOFF01 write a standard output with extension ".PRN" into the same directory as where the program is located. The output file is a comma delimited ASCII file that starts with the parameter values used. After a header, the following lines contain the output: times (s), point runoff (rainfall minus infiltration rate, m/s) and plot runoff (m^2/s). Normally, the file ends when the plot runoff is (almost) zero, at which moment the point runoff is usually negative. The time intervals vary once the build-up phase ends (see Chapter 4). The first lines of a sample output file may look like:

```
L,              93
Beta,           1.78
Sorpin,         200
Ksat,           11
Km,             14
Slope,          .04
t [s], pointRO [m/s], plotRO [m²/s]
37,3.2508E-06,1.3694E-10
38,4.6659E-06,9.7700E-10
39,5.9045E-06,3.2351E-09
40,7.0003E-06,7.6319E-09
```

The output files are easy to read by other programs, including spreadsheets.

3.3 Command line use and external calls

The preceding text describes standard, more or less manual use of the program. For research purposes, it may be necessary, however, to run the program many times with many different parameter combinations, for example to fit parameters to measured runoff values. This would be tedious to do by hand which is why it is possible to run the program via a command line, for example via the run line under Windows, in a script, or a SHELL statement in another program. When the command line "runoff01.exe a" is executed, only the core module will run, without opening any windows, after which the program will exit. This allows automated use of the program by other programs. First, such a program would adjust PARAM.DAT, after which it calls RUNOFF01 with the command line argument "a". After RUNOFF01 is finished, control passes back to the main program, which may access the results for further comparison and analysis, then adjust PARAM.DAT, run RUNOFF01 again, etc.

4 Description of code RUNOFF01.BAS

Only the core code as found in the module RUNOFF01.BAS is described. The code for the project and windows or frames is provided without further comments and annotations. It should be noted that the program was developed with Visual Basic 3.0.

RUNOFF01.BAS consists of three parts. The first part (pseudo-code numbers starting with "0.") declares all variables, assigns them initial values, either internally or through reading them from the file PARAM.DAT, and reads in the input data from a rainfall intensity file. The role of the individual variables is briefly characterized in the actual code (see Appendix).

The second part (pseudo-code numbers starting with "1.") calculates the point runoff which is the difference between the rainfall rate and the infiltration rate. First, tipping times are read and converted to rainfall intensities. Usually, rain first infiltrates before runoff occurs. To calculate the moment at which runoff starts, the so called Time Compression Algorithm (TCA) is used (Reeves & Miller, 1975; Mls, 1980). TCA assumes that the instantaneous infiltration rate only depends on the cumulative amount of rain that has infiltrated up to that point. This is why the program calculates cumulative infiltration. Once the rainfall rate exceeds the infiltration rate, ponding occurs and the point runoff is calculated as the difference between the two rates. The Philip-Two-Term equation (PTT, Philip, 1957) is used to calculate instantaneous infiltration rate:

```
infrate = (.5 * Sorp / Sqr(timeoffset + t)) + Ks
```

The infiltration rates are calculated until `tail` seconds after the end of the rain. The values are stored in the array `dhdt()` because the point runoff equals the local change in water height.

The third part (pseudo-code numbers starting with "2.") is the most elaborate part of the program and routes point runoff downhill through direct integration along the runoff characteristics of a kinematic wave (Singh, 1996; Van de Giesen *et al.*, 2003, submitted). The characteristics result from the simultaneous solution of two differential equations:

$$\frac{dh}{dt} = pntRO \qquad (1)$$

$$\frac{dx}{dt} = \alpha\beta h^{\beta-1} \qquad (2)$$

which can be found back in the code in integrated form as:

```
h = h + RO * dt
```

and

```
deltax = (alpha * (h ^ Beta - oldh ^ Beta)) / RO
x=x+deltax
```

Two runoff phases are distinguished: Build-Up and Equilibrium. During the Build-Up phase, water that fell on the top of the slope (or characteristics that started at x=0, t=0) has not yet reached the bottom of the slope. The plot runoff is calculated directly through the integration of Equation (1). Once the first characteristic from the top has reached the bottom, the Equilibrium phase starts. The next characteristic that reaches the bottom of the slope also started at the top of the slope but slightly later than the previous

characteristic. So during the Equilibrium phase, “new” characteristics are continuously started at `x=0`, `t=tlastchar+dt` and integrated until `x=L` at which time the water height determines the plot runoff.

Complications occur when the top of the plot falls dry for a short period after which a new Build-Up phase starts. Much of the code and variables are included to handle this and other “pathological” cases such as for example time steps during which the rainfall rate equals the infiltration rate (stationary water depth, `RO=0`).

Output is written to an ASCII file for each time step for which the plot runoff is calculated. Because in the Equilibrium phase, there is no reason why characteristics would arrive at x=L at any particular interval, the reporting intervals are unequal. Once the plot runoff is zero after the rain has stopped, the program halts.

Literature

Mls, J., 1980. Effective rainfall estimation. Journal of Hydrology, 45, 305-311

Philip, J.R., 1957. The theory of infiltration: 4. Sorptivity and algebraic infiltration equations. Soil Science, 84, 257-264

Reeves, M., Miller, E.E., 1975. Estimating infiltration for erratic rainfall. Water Resources Research, 11, 102-110

Singh, V.P., 1996. Kinematic wave modeling in water resources: surface-water hydrology. John Wiley&Sons, New York, 1399 pp.

Stomph, T.J., De Ridder, N., Van de Giesen, N.C., 2001. A flume design for the study of slope length effects on runoff. Earth surface processes and landforms 26(6): 647-655.

Stomph, T.J., De Ridder, N., Van de Giesen, N.C., 2002a. A flowmeter for low discharges from laboratory flumes. Transactions American Society of Agricultural Engineers, 2002, Vol 45(2): 345-349

Stomph, T.J., De Ridder, N., Steenhuis, T.S., Van de Giesen, N.C., 2002a. Scale effects of Hortonian overland flow and rainfall-runoff dynamics: Laboratory validation of a process-based model. Earth Surface Processes and Landforms, 27: 847-855

Van de Giesen, N.C., Stomph, T.J., de Ridder, N., 2000. Scale effects of Hortonian overland flow and rainfall-runoff dynamics in a West African catena landscape. Hydrological Processes, 14, 165-175

Van de Giesen, N., Stomph, T.J., De Ridder, N., 2003. Surface Runoff Scale Effects in West African Watersheds: Modeling and Management Options, submitted to Agricultural Water Management

```
Sub ro_algo ()
'*************************************************************************
'*                            RUNOFF01.BAS:                              *
'*       - Integrates characteristics of kinematic wave equation         *
'*       - Infiltration based on PTT model and time compression           *
'*                                                                        *
'*         Last revised: February 2003                                    *
'*         Authors: N. van de Giesen and T.J. Stomph                      *
'*                                                                        *
'*         Interface:                                                     *
'*         Raindata read (ASCII file with tipping times)                  *
'*         Change parameters in parameter file "param.dat"                *
'*         Set file names in "param.dat"                                  *
'*         Runoff written to ASCII file                                   *
'*         Infiltration written to ASCII file                             *
'*                                                                        *
'*         Documentation:                                                 *
'*         See ZEF Research Documentation                                 *
'*                                                                        *
'*************************************************************************

'*************************************************************************
'* 0 Initialize                                             ************
'*************************************************************************

'* 0.1 Declaration of variables and parameters             ************

'* 0.1.1  Plot related variables

 Dim Slope As Double            'Slope [-] of runoff plot
 Dim L As Double                'Length [m] of runoff plot

'* 0.1.2 Run-off parameters
 Dim alpha As Double            'Kin wave constant [m2-beta/s]
 Dim Beta As Double             'Kin wave exponent [-]
 Dim Km As Double               'K-manning         [s/m2-beta]

'* 0.1.3 Infiltration input parameters
 Dim Ks As Double               'Saturated conductivity [m/s]
 Dim Ksat As Double             'Saturated conductivity [mm/h]
 Dim Sorp As Double             'Sorptivity [m/s^.5]
 Dim Sorpin As Double           'Sorptivity*10^6

'* 0.1.4 Rainfall related parameters
 Dim tipsize As Double          'Size of raingauge tip [m]
 Dim tail As Double             'Duration of simulation after rain [s]

'* 0.1.5 Calculated rainfall, infiltration and runoff characteristics
 Dim intens As Double           'Rainfall intensity [mm/s]
 Static rain(1000, 2) As Double   'Rainfall intensities ([s],[m/s])
 Static Pcum(8000) As Single      'Cumulative rainfall every dt sec.
 Static Icumar(8000) As Single    'Cumulative infil every dt sec.
 Dim dp As Double               'dP
 Dim di As Double               'dI
 Static dhdt(8000) As Single    'dh(t)/dt
 Dim runoff As Double           'Runoff (point)
 Dim Icum As Double             'Cumulative infiltration
 Dim infrate As Double          'Infiltration rate

'* 0.1.4 Programming variables
 Dim dt As Double               'Time interval [s]
 Dim t As Double                'Time [s]
 Dim deltat As Double           'Time increase as calculated [s]
 Dim told As Double             'memory of previous t [s]
 Dim tpond As Double            'Time ponding occurs first [s]
 Dim timeoffset As Double       'Time compression needed for ponding [s]
 Dim tlastchar As Double        'Start time last characteristic [s]
 Dim tlastend As Double         'Time last characteristic reached end of plot [s]
 Dim endinterval As Double      'Time of last rain tip [s]
 Dim endrain                    'Time of last raintip [s]
 Dim x As Double                'x [m], progress along plot
 Dim h As Double                'h [m], depth water layer
 Dim oldh As Double             'memory of h [m]
```

```
        Dim deltax As Double            'increase in x as calculated [m]
 75     Dim plotRO As Double            'Runoff from plot [m2/s]
        Dim oldplotRO As Double         'Memory of runoff from plot [m2/s]
        Dim complotRO As Double         'Cumulative plot runoff [m]
        Dim cnt                         'Time interval counter [-]
        Dim cnt2                        'Counter [-]
 80     Dim roflag                      'Ponding/runoff flag (boolean) [-]
        Dim dum As Single               'Dummy variable for print output

        Dim file As String              'Name of rainfall file
        Dim file2 As String             'Name of output file
 85
       '* 0.2 Set values of variables and parameters              ***********

       '* 0.2.1 set all programming variables to zero
        t = 0#
 90     x = 0#
        h = 0#
        Icum = 0#
        plotRO = 0#
        cumplotro = 0#
 95     cnt = 0
        oldh = 0#
        tlastchar = 0#
        tlastend = 0#
        endrain = 0
100     intens = 0#
        oldplotRO = 0#

       '* 0.2.2  Parameter input from file
        parfile$ = "param.dat"          'Parameter in same directory as program
105     Close #1
        Open parfile$ For Input As #1

        Input #1, file                  'Name of rainfall file
        Input #1, file2                 'Name of output file
110
        Input #1, L                     'Plot length (m)
        Input #1, Beta                  'Turbulence exponent
        Input #1, Sorpin                'Initial sorptivity rate x 10^6
        Input #1, Ksat                  'Final infiltration rate
115     Input #1, Km                    '1/mannings roughness coefficient
        Input #1, dt                    'Time step [s]
        Input #1, tail                  'Time allowed for calculations after end of rain
        Input #1, Slope                 'Slope [-]

120     Close                           'Parameter file

       '* 0.2.3 Convert input data to correct dimensions
        alpha = Sqr(Slope) * Km         'Kinematic wave constant
        Ks = Ksat / (3600# * 1000#)     'Saturated conductivity [m/s]
125     Sorp = Sorpin / 1000000#        'Sorptivity fitted


       '* 0.3 Prepare output file                                 ***********

130     Open file2 For Output As #2     'File for runoff results

       '* 0.3.1 File header
        Print #2, "L,", L
        Print #2, "Beta,", Beta
135     Print #2, "Sorpin,", Sorpin
        Print #2, "Ksat,", Ksat
        Print #2, "Km,", Km
        Print #2, "Slope,", Slope

140     Print #2, "t [s], pointRO [m/s], plotRO [m²/s]"
```

```
'*********************************************************************
'* 1 Convert rainfall to point runoff                      ***********
'*********************************************************************

'* 1.1 Read rainfall data from file and calculate intensities ********

'* 1.1.1 Open rainfall data file
 Open file For Input As #1          'Rainfall data

'* 1.1.2 Read rainfall data file
 Line Input #1, tek$                '1st line of header
 Line Input #1, tek$                '2nd line of header
 Input #1, tipsize                  'Rain per tip [mm]
 tipsize = tipsize / 1000#          'Rain per tip [m]

 While NOT EOF(1)
   Input #1, rain(cnt, 1)           'Read tiptimes into first column rain array
   cnt = cnt + 1
 Wend
 length = cnt - 2                   'Store number of lines in rain array
 endrain = rain(length, 1)          'Store last tiptime of rain
 rain(0, 2) = 0#                    'Set rain intensity at start to zero
 cnt = 0                            'Reset counter

'* 1.1.3 Calculate rain intensity at every tiptime
 While cnt < length
   cnt = cnt + 1
   rain(cnt, 2) = tipsize / (rain(cnt, 1) - rain(cnt - 1, 1))
 Wend

 cnt = 0                            'Reset counter
 Pcum(0) = 0                        'Set cumulative rain at time 0 to 0
 endinterval = 0                    'Set end of current time interval between tips to 0
 cnt2 = 0                           'Reset counter

'* 1.1.4 Calculate and store cumulative rainfall for every dt between time=0
'        and time is 'tail' seconds later than end of rain
 While t < (endrain + tail)
   t = t + dt

  '1.1.4.1 When current time is greater than time of last tip then adjust time
  '        last tip AND intensity. Otherwise keep the old values
   If t > endinterval Then

     '1.1.4.2 When the end of the rain is not yet reached increase line counter and
     '        get new values from rain array. Otherwise set intensity to 0 and
     '        time last tip to beyond final time of simulation
      If cnt2 < length Then
        cnt2 = cnt2 + 1
        endinterval = rain(cnt2, 1)
        intens = rain(cnt2, 2)
      Else
        endinterval = endrain + tail + 1
        intens = 0
      End If
   End If

   cnt = cnt + 1                            'increase counter
   Pcum(cnt) = Pcum(cnt - 1) + intens * dt  'set cumulative rainfall array

 Wend

 t = 0                                      'reset time
 cnt = 0                                    'reset counter
```

```
'* 1.2 Calculate infiltration and point runoff or rainfall excess ****

roflag = 0                      'Set runoff flag to no runoff
Icum = .00000001                'Set cumulative infiltration to almost zero

While t < (endrain + tail)      'Until end of runoff simulation period
  t = t + dt
  cnt = Int(t / dt)

'* 1.2.1 When there is ponding (so point runoff) calculate infiltration rate directly
'        Otherwise calculate time needed to arrive at ponding at this moment and
'        calculate the corresponding infiltration rate. This is the so-called
'        "Time Compression Algorithm"

  If roflag = 1 Then
    infrate = (.5 * Sorp / Sqr(timeoffset + t)) + Ks
  Else
    tpond = ((-Sorp + Sqr(Sorp ^ 2 + 4 * Ks * Icum)) / (2 * Ks)) ^ 2
    infrate = (.5 * Sorp / Sqr(tpond)) + Ks
    intens = (Pcum(cnt) - Pcum(cnt - 1)) / dt

'* 1.2.2 As long as infiltration rate is superior to rainfall intensity keep
'        on going. Otherwise set runoff flag to 1 and calculate time offset
    If infrate >= intens Then
      infrate = intens
    Else
      roflag = 1
      timeoffset = tpond - t
    End If
  End If
  Icum = Icum + infrate * dt                    'Increase cumulative infiltration
  Icumar(cnt) = Icumar(cnt - 1) + infrate * dt  'Adjust cumulative infiltration array
Wend
roflag = 0                                      'Reset runoff flag
t = 0                                           'Reset time
cnt = 0                                         'Reset counter

'* 1.2.3 Calculate the increase in depth of the water layer (dhdt) from the
'        change in cumulative rainfall and in cumulative infiltration
While cnt < Int((endrain + tail) / dt)
  cnt = cnt + 1
  dp = Pcum(cnt) - Pcum(cnt - 1)
  di = Icumar(cnt) - Icumar(cnt - 1)
  dhdt(cnt) = (dp - di) / dt                    'These values are used for routing
                                                'kinematic wave (see 2 below)
Wend
cnt = 0                                         'Reset counter


'*********************************************************************
'* 2 Calculate plot runoff                                 ***********
'*********************************************************************

While t < (endrain + tail)                      'Until end simulation

'* 2.1 Build-up phase                                     ***********

'* 2.1.1 Check for runoff
  t = t + dt                                  'Increase time
  cnt = Int(t / dt)                           'Calculate array counter
  RO = dhdt(cnt)                              'Get runoff from array

  While (RO <= 0) And (t < (endrain + tail))  'Wait for runoff
    t = t + dt                                'Increase time
    cnt = Int(t / dt)                         'Calculate array counter
    RO = dhdt(cnt)                            'Set runoff
  Wend

  tlastchar = t                               'Start of last characteristic
  x = 0#
  oldh = 0#
  h = 0#
```

```
        If tlastend = 0 Then
          told = t - dt
285     Else
          told = tlastend
        End If

        While (x < L) And (t < (endrain + tail))      'Until plot or simulation end reached
290
    '* 2.1.2 Check for positive depth of water layer
    '        Calculate new depth water layer or set to zero
          If (h + RO * dt) > 0 Then
            h = h + RO * dt
295       Else
            h = 0
          End If

    '* 2.1.3 Calculate increase in x for increase in t
300
         '2.1.3.1 Check for stationary depth of water layer (thus division by zero)
          If RO <> 0 Then
            deltax = (alpha * (h ^ Beta - oldh ^ Beta)) / RO
          Else
305         deltax = dt * alpha * Beta * h ^ (Beta - 1) 'No change in depth
          End If

         '2.1.3.2 Calculate new x
          x = x + deltax
310
    '* 2.1.4 Check whether characteristic still before end of plot
    '        If still before end, of plot reset memory of h, increase time and counter
    '        read runoff at new time from array
    '        If end is reached, calculate t at exactly end of plot set x= length of plot
315 '        and calculate new h at this time, avoiding negative h.

         '2.1.4.1 Check x < L
          If x < L Then                       'Plot end not yet reached
            oldh = h
320         t = t + dt
            cnt = Int(t / dt)
            RO = dhdt(cnt)
          Else                                'Plot end reached

325        '2.1.4.2 Calculate exact time at which x=L
            If RO <> 0 Then
              deltat = ((RO * (L - (x - deltax)) / alpha + oldh ^ Beta) ^ (1 / Beta) - oldh) /
    RO
            Else
330           deltat = (L - (x - deltax)) / (alpha * Beta * h ^ (Beta - 1))
            End If
            t = t - dt + deltat
            x = L

335        '2.1.4.3 Calculate new h
            If (h + RO * deltat) > 0 Then
              h = h + RO * deltat
            Else
              h = 0
340         End If
          End If
```

```
'* 2.1.5 Check for height water layer, calculate actual plotRO accordingly.
'        In the first build-up fase cumulative plot runoff is calculated and
'        recorded, and memory of time and plotrunoff is reset. When the
'        build-up phase runs while time t is inferior to the last time, a
'        characteristic reached the end of the plot and no cumulative runoff
'        is calculated or recorded and memories are not reset. If water layer
'        is zero seek for time new runoff starts,set start time of last
'        characteristic and reset starting values of x and memory of water height.

    If h > 0 Then                                   'Runoff exists

      plotRO = alpha * h ^ Beta

      If t > tlastend Then                          'Characteristic arrives after last one
        cumplotro = cumplotro + (t - told) * .5 * (plotRO + oldplotRO)
        oldplotRO = plotRO
        told = t

       '2.1.5.1 Report
        dum = t                                     'To prevent double precision in output
        Print #2, Format$(dum, "#######.##");       't
        Print #2, ",";
        dum = RO
        Print #2, Format$(dum, "0.0000E+00");       'pointRO
        Print #2, ",";
        dum = plotRO
        Print #2, Format$(dum, "0.0000E+00")        'plotRO

      End If

    Else                                            'No runoff
      plotRO = 0

      If t > tlastend Then                          'Characteristic arrives after last one
        cumplotro = cumplotro + (t - told) * .5 * (plotRO + oldplotRO)
        oldplotRO = plotRO
        told = t

       '2.1.5.2 Report
        dum = t                                     'To prevent double precision in output
        Print #2, Format$(dum, "#######.##");       't
        Print #2, ",";
        dum = RO
        Print #2, Format$(dum, "0.0000E+00");       'pointRO
        Print #2, ",";
        dum = plotRO
        Print #2, Format$(dum, "0.0000E+00")        'plotRO

      End If

      While (RO <= 0) And (t < (endrain + tail)) 'Wait for more runoff
        t = t + dt
        cnt = Int(t / dt)
        RO = dhdt(cnt)
      Wend

      tlastchar = t
      x = 0
      oldh = 0

    End If

  Wend                                              'End of build-up phase

'2.1.6 Check whether t<endtime. If t>endtime skip next "While" loop (no more runoff).
  If t < (endrain + tail) Then
    tlastend = t

```

```
'* 2.2 Equilibrium phase                                    ***********

'* 2.2.1 Start new characteristic at x=0
415       While h > 0                                   'Continue while water on plot
            h = 0                                       'Reset water height
            oldh = 0                                    'Reset oldh
            x = 0                                       'Start at x=0
            roflag = 1                                  'Set runoff flag
420         t = tlastchar + dt                          'Update t
            tlastchar = t                               'Remember starting time
            cnt = Int(t / dt)                           'Calculate counter
            RO = dhdt(cnt)                              'Get point runoff

425         While (x < L) And (roflag = 1)              'While not end plot or end runoff

             '2.2.1.1 Check for positive depth of water layer
             '        Calculate new depth water layer or set to zero
             If (h + RO * dt) > 0 Then
430            h = h + RO * dt
             Else
               h = 0
               roflag = 0
             End If
435
            '2.2.1.2 Check for stationary depth of water layer (thus division by zero)
             If RO <> 0 Then
               deltax = (alpha * (h ^ Beta - oldh ^ Beta)) / RO
             Else
440            deltax = dt * alpha * Beta * h ^ (Beta - 1)
             End If

            '2.2.1.3 Calculate increase in x for increase in t
             x = x + deltax
445
'* 2.2.2 Check whether characteristic still before end of plot.
'        If still before end of plot, reset memory of h, increase time and counter
'        read runoff at new time from array
'        If end is reached calculate t at exactly end of plot set x= length of plot
450  '        and calculate new h at this time, avoiding negative h.
             If x < L Then                          'Characteristic not yet at end plot
               oldh = h
               t = t + dt
               cnt = Int(t / dt)
455            RO = dhdt(cnt)
             Else                                   'Characteristic reached end plot
              '2.2.2.1 Calculate exact time at which x=L
              '2.2.2.2 Check for point runoff
               If RO <> 0 Then
460              deltat = ((RO * (L - (x - deltax)) / alpha + oldh ^ Beta) ^ (1 / Beta) -
oldh) / RO
               Else
                 deltat = (L - (x - deltax)) / (alpha * Beta * h ^ (Beta - 1)) 'No change in
depth
465            End If

              '2.2.2.3 Update t and x
               t = t - dt + deltat
               x = L
470
              '2.2.2.4 Calculate h
               If (h + RO * deltat) > 0 Then
                 h = h + RO * deltat
               Else
475              h = 0
                 roflag = 0
               End If

             End If
480
         Wend                                 'End plot or end runoff
```

```
'* 2.2.3 Calculate plot runoff

485         If h > 0 Then                     'Check for water on plot
              plotRO = alpha * h ^ Beta       'Calculate plot runoff
            Else                              'No water on plot
              plotRO = 0                      'Set plot runoff zero
            End If
490
'* 2.2.4 When the end of the plot is reached, set memory of last characteristic which
'         reached end of plot to present time and calculate cumulative and report
'         actual runoff, otherwise just continue
            If x = L Then                     'End of plot reached
495           cumplotro = cumplotro + (t - told) * .5 * (plotRO + oldplotRO)
              tlastend = t

             '2.2.4.1 Report
              dum = t                                     'To prevent double precision in output
500           Print #2, Format$(dum, "#######.##");       't
              Print #2, ",";
              dum = RO
              Print #2, Format$(dum, "0.0000E+00");       'pointRO
              Print #2, ",";
505           dum = plotRO
              Print #2, Format$(dum, "0.0000E+00")        'plotRO

            End If

510        '2.2.4.2 Set memory of plotrunoff and time of last runoff calculation
            oldplotRO = plotRO
            told = t

          Wend                              'End of equilibrium phase
515
        End If                             'This "If - End If" checked for runoff

       Wend                               'End of simulation

520    '*********************************************************************
       'End              'OF MAIN PROGRAM                          ***********
       '*********************************************************************

       End Sub
525
```