THE DEVELOPMENT AND APPLICATION OF A KINEMATIC STORMWATER MANAGEMENT MODEL

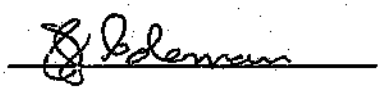
Trevor John Coleman

A project report submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master Science in Engineering.

Johannesburg 1990

DECLARATION

I declare that this project report is my own, unaided work. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University



eleventh day of MAY 1990

ABSTRACT

Urban stormwater drainage designers and planners are having to deal with the dramatic effect that urbanization has on the hydrological regime. To cope economically with the increases in runoff volumes and peaks due to urbanization, more sophisticated approaches are required for the design and planning of stormwater drainage systems. This has led to the need for better tools to enable stormwater drainage system designers to evaluate such drainage options as dual drainage, use of flood plains, and detention/retention storage facilities. WITSKM (Witwatersrand Stormwater Kinematic Model) was developed specifically for the analysis of these stormwater drainage options. The model is therefore a single event model and does not include an evaporation component.

WITSKM has been developed using the BASIC computer programming language for IBM compatible micro-computers. To reduce the tedious process of data input and editing, an editor has been included in the program. In developing WITSKM the modular approach was adopted so that the necessary flexibility in modelling urban drainage systems can be achieved. Modules are included that can model flow over impermeable and permeable surfaces, aquifers, pipes, trapezoidal channels, and detention/retention storage facilities. A system of module numbers is used to determine the connectivity and the modules to which overflows should be routed in the simulation of dual drainage systems. With the inclusion of aquifer modules, the processes of subsurface flow and interflow can be modelled. The Green-Ampt infiltration model has been used as the parameters for this model can be physically measured.

In WITSKM the kinematic routing approach has been adopted for the routing of flows over permeable and impermeable surfaces as well as through pipes and channels. Two numerical methods of solving the kinematic equations were tested. These were an explicit backward

difference scheme as used in an earlier version of WITSKM, and the Muskingum-Cunge routing method. The Muskingum-Cunge method proved to be more stable and give more consistent answers when varying the computation time step than the explicit backward difference scheme and was therefore used in WITSKM. The employment of this technique required a revision of the connectivity algorithm. The shortcomings in the original version in modelling combinations of spillway and outlets for the detention/retention storage module were also overcome.

The model was calibrated on three catchments viz Newark Street, Sunninghill Park, and Zululand. The results were compared using various goodness-of-fit criteria to those obtained by using WITWAT. The Zululand catchment was used to test the subsurface and interflow capabilities of the model by comparing runs using the Hortonian concept of overland flow with those having subsurface flow and interflow. The model was further used to compare possible stormwater drainage options for a hypothetical township layout on the Waterval catchment to the north of Johannesburg. This catchment has been monitored by the Water Systems Research Group of the University of the Witwatersrand and recorded rainfall events were used as input to WITSKM. In this way the effectiveness of the different drainage options in limiting runoff volumes and peaks from the proposed township could be compared to the measured runoff.

In this report WITSKM has been shown to compare favourably with the WITWAT model. However the versatility achieved in WITSKM by using the modular approach of catchment discretization, allows for the redirection of flows and overflows to easily simulate the different stormwater drainage options for an urban area. This together with the inclusion of the Muskingum-Cunge routing technique, and the interflow capabilities makes WITSKM a more versatile model than the WITWAT model as far as the analysis of stormwater drainage options are concerned.

ACKNOWLEDGEMENTS

I wish to acknowledge my supervisor Professor D. Stephenson for his guidance and encouragement throughout this study. In particular his assistance with the computer coding of the original version of WITSKM. I am also indebted to Mr Andrew Holden for his assistance with the Muskingum-Cunge routing procedures and to Dr Jeffrey Lambourne for his advice and assistance with the editor's computer coding. Furthermore I wish to thank the Water Research Commission for supporting the research project under which the work for this project report was done.

TABLE OF CONTENTS

		Page
1	INTRODUCTION	1
1.1	General	1
1.2	Available Simulation Models	2
1.3	Description of WITSKM	3
1.4	Scope of Report	3
2	KINEMATIC THEORY	5
2.1	Introduction	5
2.2	Equations of Flow	5
2.3	Kinematic Equations	8
2.4	Numerical Solution of Kinematic Equations	9
2.4.1	Introduction	9
2.4.2	Stability and Accuracy Criteria for Numerical Schemes	10
2.4.3	Numerical Solutions of the Kinematic Equations	12
3	ROUTING PROCEDURES USED IN WITSKM	19
3.1	Introduction	19
3.2	Formula for the Calculation of the Celerity	19
3.3	Overland Flow	20
3.4	Trapezoidal Channel Routing	23
3.5	Pipe Routing	26
3.6	Aquifer Routing	29
3.7	Storage Routing	31
4	GENERAL DESCRIPTION OF PROGRAM	34
4.1	General Background	34
4.2	Model Description	34
4.3	Connectivity	35

5	INFILTRATION AND INTERFLOW	38
5.1	Introduction	38
5.2	The Green-Ampt Model	39
5.3	Subsurface Flow and Interflow	42
6	CALIBRATION AND VERIFICATION OF WITSKM	45
6.1	Introduction	45
6.2	Newark Street	46
6.3	Sunninghill Park	47
6.4	Zululand Catchment W1M17	59
7	APPLICATION OF WITSKM TO THE WATERVAL CATCHMENT	67
7.1	Introduction	67
7.2	Description of Proposed Township	67
7.3	Storms used in Analysis	69
7.4	Drainage System Options	71
7.4.1	Introduction	71
7.4.2	Drainage Systems	72
7.5	Results of Simulations	75
8	CONCLUSIONS AND RECOMMENDATIONS	83
REFERENCES		85

APPENDICES

Appendix A Observed Storm Events
Appendix B Goodness of Fit Criteria
Appendix C User Manual
Appendix D Program Listings

LIST OF FIGURES

Figure		Page
2.1	Continuity of flow	6
2.2	Momentum balance	7
2.3	Effect of grid spacing on stability and accuracy for an explicit finite difference scheme	11
2.4	Computational cell on finite difference grid	12
3.1	Change of hydrograph shape with level of discretization	22
3.2	Trapezoidal channel cross section	23
3.3	Trapezoidal channel routing	25
3.4	Pipe cross section	26
3.5	Pipe routing	28
3.6	Definition sketch aquifer routing	29
3.7	Definition sketch of storage facility	32
4.1	Program flow chart	36
5.1	Definition sketch foe Green-Ampt infiltration model	41
5.2	Definition sketch of interflow process	44
6.1	Discretization of Newark Street	46
6.2	Comparison of observed and simulated hydrographs (Newark Street storm no 15)	49
6.3	Comparison of observed and simulated hydrographs (Newark Street storm no 23)	50
6.4	Discretization of Sunninghill Park Catchment	53
6.5	Comparison of observed and simulated hydrographs (Sunninghill Park storm 7-01-87)	56
6.6	Comparison of observed and simulated hydrographs (Sunninghill Park Storm 9-01-87)	57
6.7	Discretization of Zululand catchment (WITWAT Green and Stephenson, 1986)	60
6.8	Discretization of Zululand catchment (revised discretization for subsurface flow)	60
6.9	Comparison of hydrographs Zululand catchment (Storm 7-02-1977)	64

6.10 Comparison of hydrographs for Zululand catchment (Storm 9-11-1977) 65
7.1 Discretization of proposed Waterval town 68
7.2 Output hydrographs for drainage systems (event 1 26-03-1987) 77
7.3 Output hydrographs for drainage systems (event 2 3-02-1987) 78
7.4 Output hydrographs for drainage systems (event 3 21-03-1987) 79

LIST OF TABLES

Table		page
3.1	Peak flows (m^3/s) for overland flow plane	21
3.2	Peak flows (m^3/s) for trapezoidal channel	24
3.3	Peak flows (m^3/s) for pipes	27
6.1	Newark street module data	48
6.2	Goodness-of-fit statistics for Newark Street	51
6.3	Sunninghill Park module data	54
6.4	Goodness-of-fit statistics Sunninghill Park	58
6.5	Discretization Zululand catchment-WITWAT	61
6.6	Revised discretization of Zululand catchment	62
6.7	Infiltration parameters used for Zululand catchment	63
7.1	Module numbers for the different land uses	69
7.2	Infiltration and roughness parameters	70
7.3	Rainfall and runoff information for recorded strom events	70
7.4	Results of simulations	76
7.5	Maximium flow depths (m) on roads and flood plains	81

CHAPTER 1 INTRODUCTION

1.1 General

Hydrologists, stormwater drainage system designers, and water resource planners in South Africa are having to deal with higher levels of urbanization as well as greater population densities than previously. The impact of the urbanizing process on the hydrological regime is dramatic and in order to cope with these changes more sophisticated computational techniques have to be used to determine viable stormwater management policies and for the design of drainage systems. The policy of constructing a stormwater reticulation network to remove the higher recurrence interval runoff events from a catchment as quickly as possible results in costly systems. To implement this approach very often the natural drainage channels have to be enlarged and lined to cope with the increased runoff due to urbanization. This type of functional engineering structure is finding less favour with a more environmentally conscious public. The drainage engineer therefore not only has to look for more aesthetically attractive and environmentally acceptable solutions but more cost effective solutions as well.

The requirement of more sophisticated stormwater management and design approaches has led to the need for better tools to enable the optimum planning and design of stormwater systems to be undertaken in new developments or for the upgrade of existing stormwater drainage systems. The techniques being adopted are those of computer simulation models, which are capable of accounting for most of the physical processes involved in the rainfall-runoff process. This results in a greater level of sophistication and refinement in describing the urban runoff process and has found acceptance amongst the majority of designers and planners.

1.2 Available Simulation Models

There are a number of computer based hydrological simulation models available for use and some of the models applicable to urban drainage applications are briefly discussed here.

SWMM (Storm Water Management Model (Huber et al,1982)) is probably one of the best known stormwater hydrological simulation models. The model's structure consists of a number of blocks for carrying out different tasks. SWMM has the capability of modelling runoff quantity and quality as well as simulating treatment facilities and receiving water quality. The kinematic flow routing procedure is used for most applications. However where backwater effects are important the full dynamic equations can be used. One of the shortcomings of this model was that it needed to run on a main frame computer and requires extensive input data. A micro-computer version of SWMM has however been developed (James and Robinson, 1985).

ILLUDAS (Illinois Urban Drainage Area Simulator (Terstriep and Stall, 1974)) has been tested and adapted for South African conditions by Watson (1981). Both ILLUDAS and ILLUDAS-SA, developed by Watson, use the time-area or isochronal approach for the routing of flows and have been found useful in the design of pipe networks.

KINE 2 (Constantinides, 1982) employs the 2 dimensional kinematic equations for flow routing and has been found to yield extremely reliable results for the catchments studied. This model can be used for the prediction of runoff hydrographs as well as for assessing the effects of man-made changes on runoff. This model has been developed for use on a mainframe computer and is data intensive. It was not developed for modelling urban areas, and is primarily for surface runoff off 2-dimensional sloping catchments.

WITWAT (Green and Stephenson, 1984) employs the kinematic method of flow routing with the option of using time-shift routing in conduits. A simplified routine was included in the model for the evaluation of detention storage facilities. The advantage of this model over SWMM and KINE 2 is that it was written specifically for a micro-computer thus alleviating many of the problems associated with mainframe data processing. The fact that WITWAT was developed on a micro-computer has made the computer simulation of urban drainage systems more readily available to hydrologists and stormwater drainage design engineers. The model was improved by Kolovopoulos (1986) to include a dual drainage capability and to account for compound channels. However the dual drainage system was inflexible in that surcharge from pipes had to flow into a channel immediately above the pipe. This does not allow for surcharges onto overland flow areas or into soakaways. Both of which are accepted stormwater management policies.

1.3 Description of WITSKM

WITSKM (Witwatersrand Stormwater Kinematic Model (Stephenson,1989)) was developed to overcome the shortcomings of the WITWAT model in analysing the effects of stormwater management policies. This model uses the kinematic method of flow routing including the routing of flow through conduits. The method of using modules to represent the components that go to make up a stormwater drainage system was adopted in WITSKM because of the resultant flexibility and versatility of the model. Modules are included that can model impermeable and permeable surfaces, aquifers, pipes, trapezoidal channels, and detention/retention storage dams.

1.4 Scope of Report

In applying the model to the Sunninghill catchment in Sandton (Stephenson, 1989) shortcomings were discovered in the model. The WITSKM model used an explicit backward difference finite difference scheme for solving the kinematic routing equations which can become

unstable especially when applied to conduit routing. This results in an impractically small computation time step having to be used to ensure stability which results in longer computation times and the use of excessive amounts of computer memory to store rainfall input data and output data. This limited the duration of the storms that could be simulated as well as the number of modules that could be used to model a catchment. During its application, shortcomings in the module used to simulate detention/retention storage were highlighted as the model did not cater for combinations of spillway and outlets.

The aims of this study are twofold; first to improve the existing program by including a more robust solution algorithm to the kinematic routing equations viz the Muskingum-Cunge method (Holden and Stephenson, 1988). Due to this methodology a new connectivity and calculation order routine had to be introduced. The detention/retention storage module was also improved to include flow through culverts a feature which can be used for flood attenuation.

The primary aim of this study is the application of the model. For this purpose the recorded data for Newark Street, Zululand catchment W1M17 as well as the Waterval and Sunninghill Park catchments were used. The Sunninghill Park and Waterval catchments are monitored by the Water Systems Research Group of the University of the Witwatersrand. The Newark Street and Sunninghill Park catchments were used to compare the results of WITSKM with those obtained using WITWAT. The Zululand catchment was used to test the interflow and subsurface flow capabilities of the program. The Waterval catchment is undeveloped at present. A town was set out on the catchment and using recorded storm events, the effectiveness of the possible stormwater management options for the town could be examined and compared to the recorded runoff.

CHAPTER 2 KINEMATIC THEORY

2.1 Introduction

The kinematic theory was introduced by Lighthill and Whitham (1955) and later used by Henderson and Wooding (1964) to study the runoff hydrograph resulting from excess rain. This theory has since been incorporated in models such as SWMM and WITWAT to model overland flow. Kinematic theory has advantages over time-area methods in that its basis is founded in hydraulic theory and the non-linearity of the surface flow process can be taken into account. The solution of the kinematic wave equation is much simpler than that of the general flow equations. This results in simpler numerical schemes which enable fast computer programs to be developed on micro-computers. WITSKM employs the kinematic routing theory not only for overland flow routing but for the routing of flows through conduits and aquifers.

2.2 Equations of Flow

The St Venant equations describing one dimensional flow in open channels are used as a basis for the development of the kinematic routing technique used in WITSKM. The equations are the continuity equation and the dynamic equation and their derivation is based on the following assumptions.

(a) The fluid is homogeneous and incompressible.

(b) Flow is one dimensional

(c) Flow must be gradually varied. This implies that there must be no rapid changes in flow cross sectional area.

(d) Pressure distribution across any section is hydrostatic.

(e) The friction and turbulence can be accounted for using steady state resistance laws.

(f) Velocity is uniform over the cross section

(g) Bed slope of channel (β) is small so that $\beta \simeq \sin\beta \simeq \tan\beta$.

The continuity equation is derived by balancing mass around an element of fluid. By equating the difference between inflow rate and outflow rate to the rate of change in storage for the element shown in Fig 2.1 results in the following:

$$Qdt + q_L dxdt - (Q + \frac{\partial Q}{\partial x} dx)dt = \frac{\partial A}{\partial t} dxdt \qquad (2.1)$$

where Q is the flowrate

q_l is the lateral inflow per unit length along the x-axis.

A is the cross-sectional area.

t is time

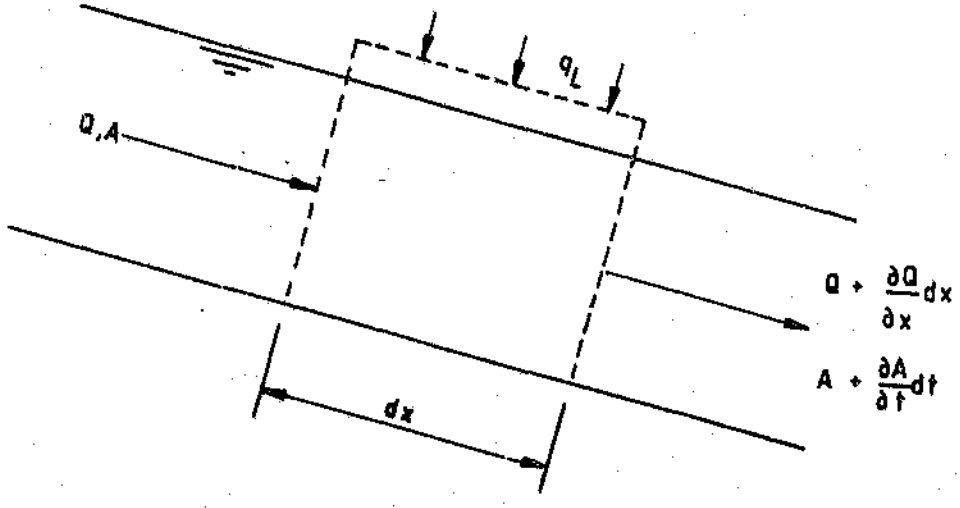


Figure 2.1 Continuity of flow (Green et al,1984)

Rearranging equation 2.1 yields

$$\frac{\partial Q}{\partial x} + \frac{\partial A}{\partial t} = q_L \qquad (2.2)$$

The dynamic equation is derived using Newton's second law of motion which states that the net force acting on an element of fluid flowing through a control volume is equal to the rate of change of momentum with time. Consider the control volume shown in Fig 2.2.

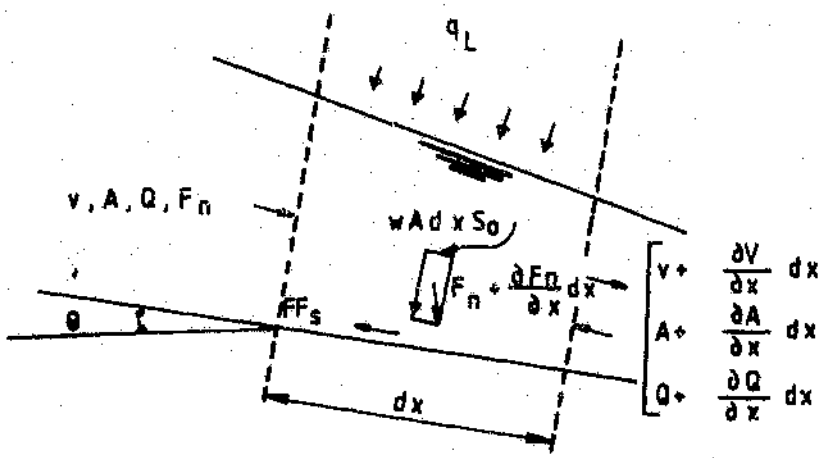


Figure 2.2 Momentum balance (Constantinides, 1982)

The resultant force in the direction of flow is given by:

$$WAS_o dx - WAS_f dx - \frac{\partial}{\partial x}(W\bar{y}A)dx = w\bar{y}A \qquad (2.3)$$

where F_n= $w\bar{y}A$ is a hydrostatic force perpendicular to the element

F_s= $wAS_f dx$ is the friction force due to boundary resistance

w is the unit weight of fluid

S_0 is bed slope

S_f is friction gradient

and $\bar{y}$ is the depth of the centroid of the cross-sectional area.

The rate of change of momentum with time is given by :

$$\frac{d}{dt}(M_a v) = \frac{\partial}{\partial t}(M_a v) + \frac{\partial}{\partial x}(M_a v)\frac{dx}{dt} \qquad (2.4)$$

where M_a is the mass of the fluid element

v is the velocity of flow

g is acceleration due to gravity

By expanding and re-arranging equation 2.4 the following results:

$$\frac{d}{dt}(M_a v) = \frac{W}{g}\,dx\,\left(v\left(v\frac{\partial A}{\partial x} + A\frac{\partial v}{\partial x} + \frac{\partial A}{\partial t}\right) + A\frac{\partial v}{\partial t} + Av\frac{\partial v}{\partial x}\right) \qquad (2.5)$$

Using the continuity equation 2.2 together with equations 2.3 and 2.5, the dynamic equation 2.6 can be derived.

$$S_f = S_o - \frac{\partial y}{\partial x} - \frac{v}{g}\frac{\partial v}{\partial x} - \frac{1}{g}\frac{\partial v}{\partial t} - \frac{q_L v}{Ag} \qquad (2.6)$$

2.3 Kinematic Equations

The kinematic equations are obtained by assuming that the friction slope S_f is equal to the bed slope S_0 i.e the pressure and inertial terms in the dynamic equation are small compared to the bed slope. The flow is therefore considered to be steady uniform flow and there exists a single valued relationship between the flow Q and the depth of flow y such that $Q=Q(y)$ and $y=y(Q)$. The term $\partial A/\partial t$ in the continuity equation can thus be rewritten in the following form:

$$\frac{\partial A}{\partial t} = \frac{dA}{dQ}\frac{\partial Q}{\partial t} \qquad (2.7)$$

The continuity equation can then be written as follows:

$$\frac{\partial Q}{\partial x} + \frac{dA}{dQ}\frac{\partial Q}{\partial t} = q_L \qquad (2.8)$$

This equation is known as the kinematic wave equation and to gain an understanding of its behaviour, the variation of Q(x,t) along a line in the (x,t) plane can be expressed as:

$$dQ = \frac{\partial Q}{\partial t} dt + \frac{\partial Q}{\partial x} dx \qquad (2.9)$$

or

$$\frac{dQ}{dt} = \frac{\partial Q}{\partial t} + \frac{\partial Q}{\partial x} \frac{dx}{dt} \qquad (2.10)$$

Comparing 2.8 and 2.10 shows that $dQ/dt = q_l(dQ/dA)$ along lines in the (x,t) plane described by

$$\frac{dx}{dt} = \frac{dQ}{dA} \qquad (2.11)$$

These lines are known as characteristics and the derivative dQ/dA as the speed of propagatior r celerity c of a kinematic wave in the (x,t) plane. Unlike the st Venant equations, the kinematic wave equation only has forward characteristics i.e information can only be carried downstream. This means that by using the kinematic equations WITSKM cannot model backwater effects.

For the routing of flows through conduits the lateral inflow term q_l in the continuity equation is dropped and the equation becomes

$$\frac{\partial Q}{\partial x} + \frac{dA}{dQ} \frac{\partial Q}{\partial t} = 0 \qquad (2.12)$$

which means that along a characteristic dQ/dt=0. This implies that there can be no hydraulic dispersion or diffusion i.e there is no lateral spreading of a hydrograph or any subsidence or attenuation of the peaks.

2.4 Numerical Solution of Kinematic Equations

2.4.1 Introduction

The kinematic equation is a non-linear partial differential equation and although for simple cases(such as a constant rainfall on an

overland flow plane of simple geometry), analytical solutions can be achieved, for modelling purposes numerical schemes have to be employed. Most numerical methods of solution can be classified into:

(a) Explicit finite difference schemes
(b) Implicit finite difference schemes
(c) Finite element methods

The application of finite element methods results in complex computer programs which are expensive to run and the accuracy and stability criteria can become tedious to apply. This approach will not be considered for use in WITSKM and will not be considered further.

For the explicit finite difference scheme the flow properties at a particular time are expressed as a function of the flow properties of the previous time step which results in an explicit solution of all the flow properties. In an implicit finite difference scheme however all the flow properties are solved for simultaneously by means of solving a matrix. The advantage of an implicit scheme is that irrespective of the grid spacing in the (x,t) plane the method is considered to remain stable. However the simultaneous solution of the flow properties by means of the matrix is time consuming and requires complex programming. The implicit scheme at large grid spacings loses accuracy. While an explicit scheme may be unstable if the grid spacing has not been chosen correctly, the method, if used correctly, is economic and accurate.

2.4.2 Stability and Accuracy Criteria for Numerical Schemes

In solving a set of non-linear partial differential equations such as the kinematic equations, the stability and accuracy of the solution have been found to depend on the choice of values for the length increment (x) and the time increment (t). Furthermore a critical ratio

$$\frac{\Delta x}{\Delta t} = \left(\frac{\Delta x}{\Delta t} \right)_{CR} \qquad (2.13)$$

has been found to exist for determining whether the solution will be stable or not. Constantinides (1982) produced a diagram (fig 2.3) showing the effect of the time and length increments on the accuracy and stability of explicit solutions to the kinematic equations. Fig. 2.3 highlights the following:

- The solution is stable as long as $\Delta x/\Delta t >= (\Delta x/\Delta t)_{cr}$
- $\Delta x/\Delta t$ should be as close to $(\Delta x/\Delta t)_{cr}$ for accuracy considerations.

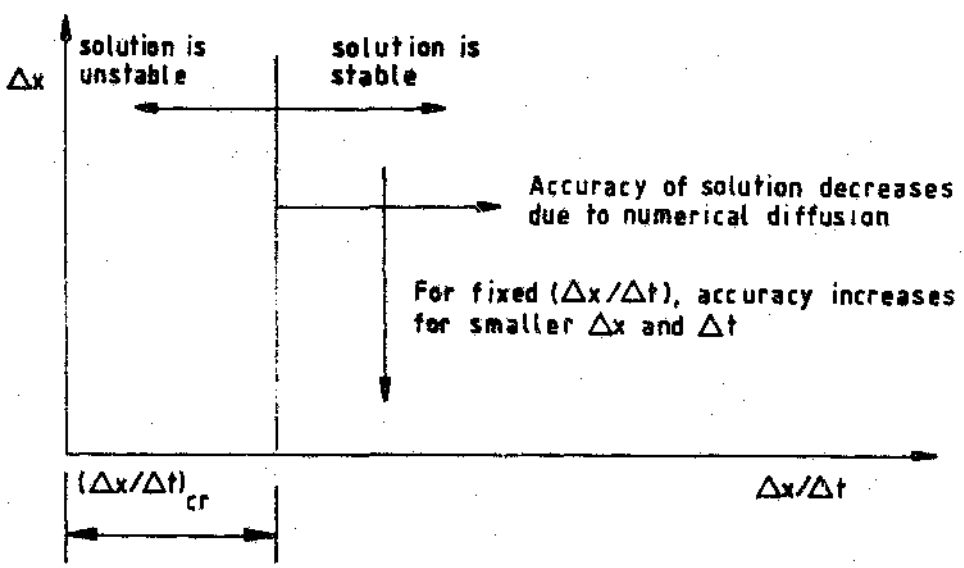


Figure 2.3 Effect of grid spacing on stability and accuracy for an explicit finite difference scheme (Constantinides, 1982)

The ratio $(\Delta x/\Delta t)_{cr}$ is the speed of propagation of a wave disturbance and can be represented by dx/dt and therefore for stability the following criterion, known as the Courant Criterion after Courant et al (1928), must hold

$$\frac{\Delta x}{\Delta t} \geq \frac{dx}{dt} \qquad (2.14)$$

In adopting a numerical solution, the solution domain has to be discretized. This representation of the partial derivatives by a finite difference scheme introduces variable amounts of numerical diffusion with the solution resembling a diffusion wave rather than a kinematic wave. This is shown in fig. 2.3. As the ratio $\Delta x/\Delta t$ increases larger amounts of numerical diffusion are introduced into the solution. As was shown in section 2.2, the kinematic equations do not allow for any attenuation of a flood wave which is somewhat unrealistic and therefore a certain amount of numerical diffusion in the kinematic wave solution would be an advantage. The aim would however be to control the amount of numerical diffusion in a way that it matches the diffusion of the physical problem.

2.4.3 Numerical Solutions to the Kinematic Equations

Most numerical finite difference solutions are expressed in terms of some or all of four discrete adjacent values of the flow Q and wave celerity c in space and time (Fig 2.4).

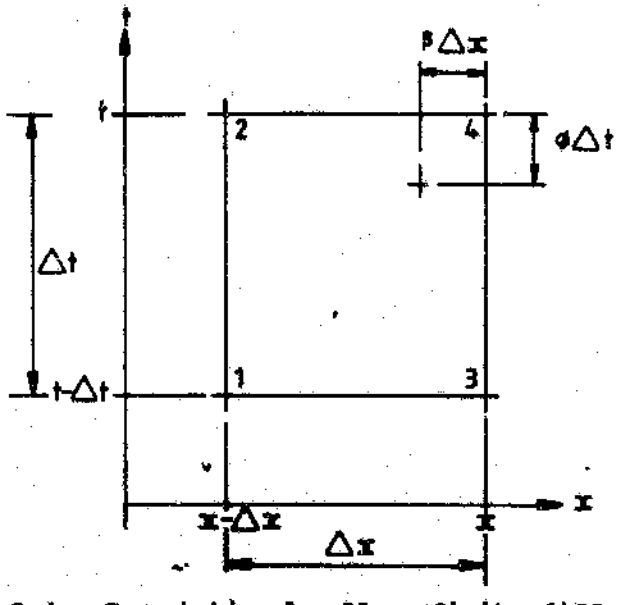


Figure 2.4. Computational cell on finite difference grid

Preissmann (1961) presented a general expression for the finite difference formulation of the kinematic equation (equation 2.15) for the computational cell shown in Fig 2.4. Q_1, Q_2, Q_3, and Q_4 are the discharges at the nodes of the computational cell and c is an average wave celerity for the cell. The β, Φ are weighting parameters used to provide flexibility in the finite difference formulation. Φ=1 implies an explicit scheme while Φ=0 results in an implicit scheme. The value β=0 results in a backward difference and β=1 in a forward difference scheme.

$$\frac{\phi(Q_3-Q_1) + (1-\phi)(Q_4-Q_2)}{\Delta x} + \frac{1}{C}\frac{\beta(Q_2-Q_1) + (1-\beta)(Q_4-Q_3)}{\Delta t} = q_L \qquad (2.15)$$

In general Φ controls the stability of the numerical scheme while β controls the accuracy or numerical diffusion of the scheme.

For use in WITSKM three schemes are examined viz

1 Explicit Backward Difference Scheme (E.B.D.S)

2 Implicit Muskingum-Cunge Scheme (I.M.C)

3 Explicit Muskingum-Cunge Scheme (E.M.C)

Explicit Backward Difference Scheme

This scheme was used in the original version of WITSKM and stability problems were experienced in the routing of flow through short, steep, smooth conduits. The problems were generally experienced in the pipe routing module of the program. This scheme is considered as a basis for comparison with the I.M.C and E.M.C schemes to determine if these schemes are worthwhile for inclusion in WITSKM

Considering equation 2.15 and putting Φ=1 and β=0, the following equation results:

$$\frac{(Q_3 - Q_1)}{\Delta x} + \frac{1}{C\Delta t}(Q_4 - Q_3) = q_L \qquad (2.16)$$

Remembering that $(Q_4-Q_3)/(\partial t)=(A_4-A_3)/\partial t$, equation 2.16 can be expressed in the following form.

$$\frac{(Q_3 - Q_1)}{\Delta x} + \frac{(A_4 - A_3)}{\Delta t} = q_L \qquad (2.17)$$

The difference (A_4-A_3) is estimated using $b_3(Y_4-Y_3)$ where b_3 is the top width of the flow whether the flow is in a pipe, channel, overland or in an aquifer module. Substituting the above into equation 2.17 the following expression results:

$$y_4 = y_3 - \frac{(Q_3 - Q_1)\ \Delta t}{\Delta x b_3} + \frac{q_L \Delta t}{b_3} \qquad (2.18)$$

Thus the flow conditions at points 1 and 3 and the depth and top width of the flow at point 3 are used to solve for the depth at point 4. Q_4 can then be calculated f on the flow depth y_4 using the Manning equation.

$$Q = \frac{\sqrt{S_o}}{n} AR^{2/3} \qquad (2.19)$$

In overland flow the width w is large compared to the flow depth y and the hydraulic radius R can be estimate by the flow depth y. The Manning equation then reduces to:

$$Q = W \frac{\sqrt{S_o}}{n} y^{5/3} \qquad (2.20)$$

Muskingum-Cunge Routing

Cunge (1969) explained why the Muskingum flood wave computation method, although assuming a linear stage/discharge relationship still attenuates a flood wave travelling along a stream. Cunge was able to derive the formula for the Muskingum coefficients from the general finite difference formulation 2.15 using a value of Φ=0.5. The attenuation achieved by the Muskingum method is due to the numerical diffusion introduced by approximating the partial differential equation by a finite difference scheme. Cunge estimated this error

using a Taylor expansion of the terms of the continuity equation and showed that the Muskingum formulation is in fact a second order approximation of the diffusion equation:

$$\frac{\partial Q}{\partial x} + \frac{1}{C}\frac{\partial Q}{\partial t} = B\frac{\partial^2 Q}{\partial x^2} \qquad (2.21)$$

where c is the wave speed
B is a diffusion coefficient

when the weighting coefficient ß is found from:

$$ß = \frac{1}{2}\left(1 - \frac{Q}{bcdxS_o}\right) \quad 0 \leq ß \leq 0,5 \qquad (2.22)$$

where b is top width of the flow

A number of researchers have used this "matched diffusivity" approach of artificially modelling attenuation by choosing ß for each computational step such that the numerical and physical diffusion are matched. Ponce and Yevjevich (1978) allowed ß to vary in time and space as the flow varies in modelling flow off an overland flow plane. Ponce (1986) compared the Muskingum-Cunge approach for routing of overland flows to the more traditional kinematic routing methods. He found that for the overland flow system analysed, the method had better convergence properties than the traditional explicit methods and the simulations were essentially independent of grid size. Holden and Stephenson (1988) applied the approach of Cunge with a variable ß for overland flow routing. This approach was compared to finite difference schemes having fixed values of ß and as in the case of Ponce, the peak was found to be independent of grid spacing. In both Ponce's and Holden and Stephenson's work the grid spacings used in the numerical experiments were fine. In the case of Ponce, the overland flow length was 36.6 m while in Holden and Stephenson's work, the tests were carried out on a plane 100m long with a grid spacing of 20m. Holden (1989) applied this method of routing to trapezoidal channels and found that putting $\Phi=0$ improved the numerical stability of the scheme.

Hydrographs were routed down a trapezoidal channel having different slopes, roughnesses and lengths. The resultant peaks of the output hydrographs were compared with those obtained using the full solution to the St Venant equations. The peaks compared favourably, with the error ranging from 0,1% to 7,3%.

The Muskingum-Cunge routing scheme is formulated by putting $\Phi=0$ in equation 2.15 and solving for Q_4 while allowing β to vary according to equation 2.23 which will allow for the matching of numerical diffusion to the physical diffusion.

$$\beta = \frac{1}{2}\left(1 + \frac{Cdt}{dx} - \frac{Q}{bcdxS_o}\right) \quad 0 \leq \beta \leq 1 \tag{2.23}$$

The following equation 2.24 results:

$$Q_4 = C_1Q_1 + C_2Q_2 + C_3Q_3 + C_4Q_L \tag{2.24}$$

where the routing coefficients are given by:

$$C_1 = \frac{K\beta}{\Delta t + K(1 - \beta)} \tag{2.24a}$$

$$C_2 = \frac{\Delta t - K\beta}{\Delta t + K(1 - \beta)} \tag{2.24b}$$

$$C_3 = \frac{K(1 - \beta)}{\Delta t + K(1 - \beta)} \tag{2.24c}$$

$$C_4 = \frac{\Delta t}{\Delta t + K(1 - \beta)} \tag{2.24d}$$

In the above equations $K = dx/\langle c \rangle$ and Q_1 is the total lateral inflow equal to q_1dx where dx is the overland flow length and $\langle c \rangle$ is the average celerity for a computational cell. In order to calculate the coefficients given in equations 2.24a to 2.24d values of K and β are needed. To do this average values of c, b, and Q for the computational cells are required. Two methods of estimating these parameters are used. One is using the average of the values of the parameters at the

four points of the computational cell given by the equations below:

$$<c> = (c_1 + c_2 + c_3 + c_4)/4 \qquad (2.25)$$

$$<Q> = (Q_1 + Q_2 + Q_3 + Q_4)/4 \qquad (2.26)$$

$$<b> = (b_1 + b_2 + b_3 + b_4)/4 \qquad (2.27)$$

As the conditions at point 4 are not known, an iterative solution is required to calculate the values of <Q>, <b>, and <c>. This approach will be called the implicit Muskingum-Cunge scheme. Another approach to estimating <c>, <b>, and <Q> is that used by Holden and Stephenson (1988) where a weighted average is used with the weighting in favour of point 3 to compensate for the missing values at point 4. This approach results in an explicit scheme requiring no iteration and will be called explicit Muskingum-Cunge routing. The equations for the average values of <c>, <b>, and <Q> are given below

$$<c> = (c_1 + c_2 + 2c_3)/4 \qquad (2.28)$$

$$<Q> = (Q_1 + Q_2 + 2Q_3)/4 \qquad (2.29)$$

$$<b> = (b_1 + b_2 + 2b_3)/4 \qquad (2.30)$$

The work of the researchers above suggests that the Muskingum-Cunge routing approach employing the matched diffusivity method should be considered for inclusion in WITSKM. However the method has only been tested at a relatively fine level of discretization with overland flow lengths in the region of 20 m. This level of discretization is too fine to use in a model such as WITSKM as too many modules would result for the computer memory. This would mean having to read and write data to and from disc to save computer memory. To model urban catchments, lengths of the order of 100-500m for both overland and conduit modules

are required to be used. A series of tests have therefore been undertaken and are described in Chapter 3 comparing the Muskingum-Cunge routing method with the explicit backward difference scheme.

CHAPTER 3 ROUTING PROCEDURES USED IN WITSKM

3.1 Introduction

In this chapter results of the two formulations of the Muskingum-Cunge routing procedure (viz the implicit Muskingum-Cunge (I.M.C) and the explicit Muskingum-Cunge (E.M.C) schemes) are compared to the explicit backward difference scheme (E.B.D.S) at levels of discretization that one would expect in modelling an urban catchment. The comparisons are done for overland flow, trapezoidal channels, and pipes. In addition the routing procedures adopted in WITSKM for reservoir and aquifer routing are described.

The criteria that were used in the comparisons of the behaviour of the schemes are the peak flow and the shape of the output hydrograph. In the case of trapezoidal channels, the results were compared to the solution of the full dynamic equation. For pipes the experimental data of Sevuk as presented by Sternberg (1989) were used to compare the accuracy of the schemes.

3.2 Formula for the Calculation of the Celerity

The celerity dQ/dA can be expressed as follows:

$$\frac{dQ}{dA} = \left(\frac{dQ}{dy}\right)\left(\frac{dy}{dA}\right) \qquad (3.1)$$

The Manning equation is given by:

$$Q = \frac{\sqrt{S_o}}{n} A R^{m-1} \qquad (3.2)$$

where m=5/3 and n is the Manning roughness coefficient. Using equation 3.1 and differentiating 3.2 with respect to flow depth y gives:

$$C = \frac{\sqrt{S_o}}{n}\left[R^{m-1} \frac{dA}{dy} + (m-1) AR^{m-2} \frac{dR}{dy} \right] \frac{dy}{dA} \qquad (3.3)$$

3.3 Overland Flow

For overland flow the area A = wy and as the flow depths are small as compared to the width w of the flow, the wetted perimeter can be approximated by w. The hydraulic radius R is therefore equal to the flow depth y and the top width of the flow b is equal to w. Substituting into equation 3.3 the celerity is given by:

$$C = \frac{\sqrt{S_o}}{n} my^{m-1} \qquad (3.4)$$

In the case of the E.B.D.S the top width b_3 in equation 2.17 is equal to w and for the M-C routing methods the celerity is given by 3.4 and the top width b required for the calculation is equal to w

The schemes were applied to an impermeable plane 250m long and 100m wide with an excess rainfall of 50 mm/h. The plane was not discretized into sub-planes i.e dx was taken as 250m and the schemes were run for two cases:

(a) a smooth plane with slope 0,085 and roughness 0,018

(b) a rough plane with slope 0,008 and roughness 0,25.

The time step dt was varied for each scheme from 1 to 10 mins. The peak was noted for each run and the results are presented below in Table 3.1. The change of shape of the output hydrograph was also examined.

From Table 3.1 for case a the results are essentially the same for the schemes. The E.B.D.S scheme went unstable for a time interval of 10 mins while the Muskingum-Cunge schemes remained stable. The reason why the E.B.D.S went unstable for the steeper smoother plane (case a) is that the Courant condition given by equation 3.4 for the celerity is higher due to the ratio $\sqrt{S_0}/n$ being greater than for the flatter, rougher plane. The I.M.C scheme produced a peak that essentially

Table 3.1 Peak flows (m^3/s) for Overland Flow Plane

Plane slope=0,085 n=0,018					
dt(mins)	1	2,5	5	7,5	10
E.B.D.S	0,347	0,347	0,348	0,350	unstable
E.M.C	0,346	0,345	0,343	0,342	0,340
I.M.C	0,346	0,346	0,346	0,347	0,347
Plane slope=0,008 n=0,25					
dt(mins)	1	2.5	5	7.5	10
E.B.D.S	0,291	0,292	0,295	0,298	0,307
E.M.C	0,291	0,288	0,282	0,277	0,271
I.M.C	0,290	0,292	0,291	0,291	0,291

remained constant with change in time step, while the E.M.C scheme's peak dropped with increasing time interval dt. These trends are more evident for the flat plane with a larger decrease in peak with dt for the E.M.C scheme. In the case of the I.M.C scheme, the peak was once again essentially constant. In addition for the I.M.C scheme, the shape of the hydrograph remained the same regardless of time step as long as the time interval was small enough to ensure sufficient time steps to describe the hydrograph and input rainfall. A further point which was highlighted by these numerical experiments, is the sensitivity of the peak and hydrograph shape to the grid spacing dx used when the ratio $\sqrt{S_0}/n$ is small (i.e case b). This is shown in Fig 3.1 where the I.M.C scheme is used for the flat plane for a fixed value of dx/dt of 50 m/min for values of dx ranging from 25m to 250m. Referring to Fig 3.1 there is little difference between the hydrographs produced using values of dx of 25 and 50m however the accuracy of the solution deteriorates for values of dx of 125 and 250m. The same trend was produced by the E.B.D.S when applied to the flat plane. Thus for overland routing, the advantage that the I.M.C scheme has in terms of stability over the E.B.D.S and the fact that the peak remains constant with changing time step for a particular grid spacing dx, makes it the best choice of the schemes considered.

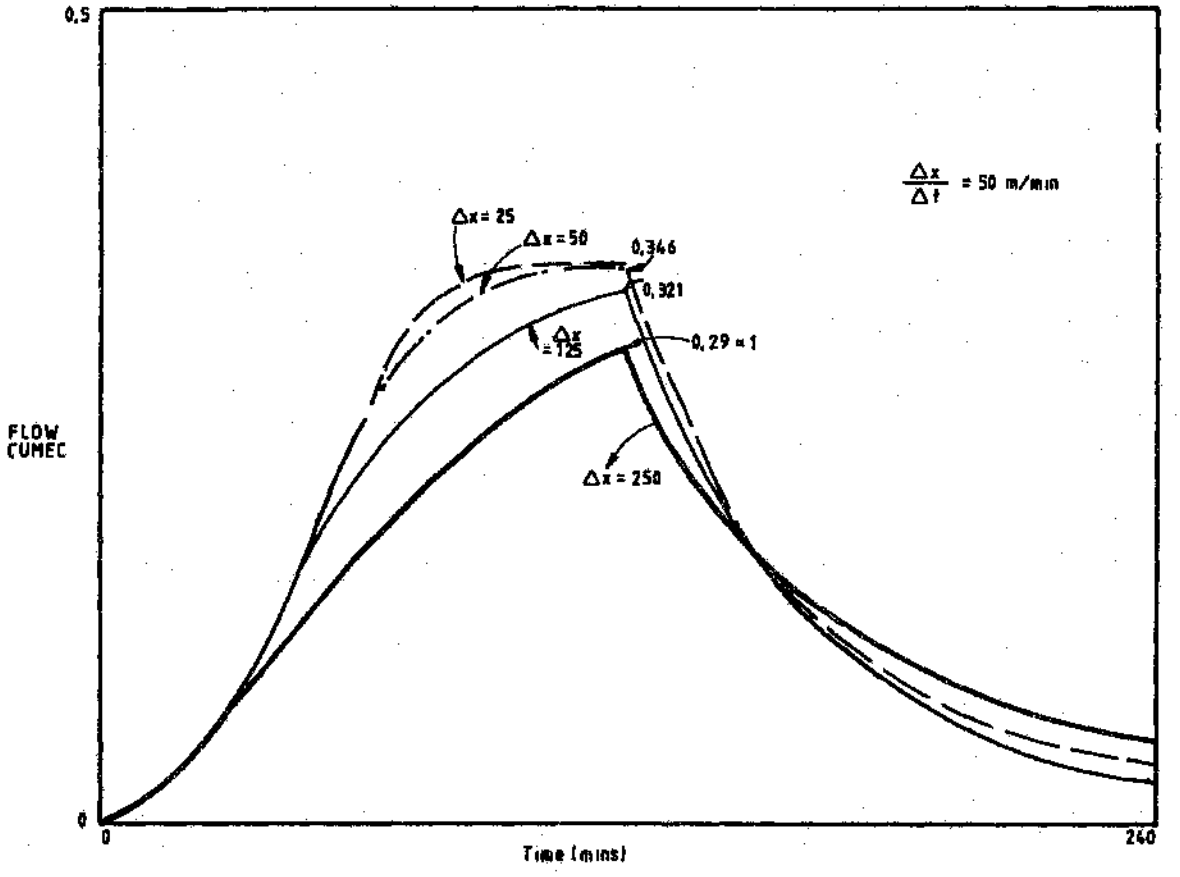


Figure 3.1. Change of hydrograph shape with level of discretisation

3.4 Trapezoidal Channel Routing

Consider the trapezoidal channel cross section shown in Fig 3.2. The flow width b, area A, and wetted perimeter P of such a channel are given by:

$$b = W + y\,(SS1 + SS2) \tag{3.5}$$

$$A = Wy + \frac{y^2}{2}\,(SS1 + SS2) \tag{3.6}$$

$$P = W + y\,((SS1^2 + 1)^{1/2} + (SS2^2 + 1)^{1/2}) \tag{3.7}$$

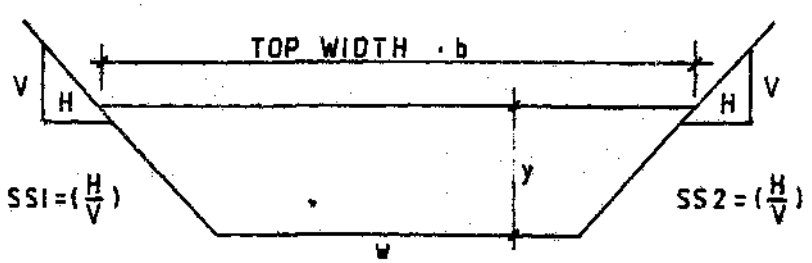


Figure 3.2. Trapezoidal channel cross section

For the E.B.D.S in calculating the flow depth y_4, the surface width b_3 in equation 2.18 is estimated using y_3 in equation 3.5. The resulting flow depth y_4 can be used in equations 3.6 to 3.7 to give the area and wetted perimeter which is used in the Manning equation to calculate the flow Q_4. The derivatives of equations 3.6 and 3.7 can be used in equation 3.4 to derive the equation for the celerity for a trapezoidal channel given in equation 3.8 below.

$$C = \frac{\sqrt{S_o}}{n}\, R^{m-1}\, \left[m - \frac{2\,(m-1)\,R}{b}\,\left((SS1^2 + 1)^{1/2} + (SS2^2 + 1)^{1/2}\right)\right] \tag{3.8}$$

Equation 3.8 and 3.5 are used to calculate the values of <b> and <c> needed for the calculation of β for the M-C routing techniques.

To compare the schemes two channels are used viz:

1 A 250m long channel with bottom width 1,2m and side slopes (horiz/vert) of (1,5/1) with a roughness of 0,03 and a slope of 0,001.

2 A 500m long channel with bottom width 1,2m and side slopes (horiz/vert) of (1,5/1) with a roughness of 0,15 and a slope of 0,001.

A hydrograph was input to the channels and routed through using the schemes described above. The computer program OSYRIS (Kolovopoulos,1989), a complete solution of the St Venant equations, was used to route the hydrograph down the two channels so the results could be compared to the full dynamic solution. The results are given below in Table 3.2.

Table 3.2 : Peak Flows (m^3/s) for Trapezoidal Channels

	250m LONG CHANNEL			
dt(mins)	1	2	4	5
E.B.D.S	21,5	21,6	unstable	
I.M.C	21,4	21,4	21,3	21,2
E.M.C	21,4	21,4	21,3	21,2
Dynamic	21,4	21,4	21,4	21,4
	500m LONG CHANNEL			
dt(mins)	1	2	3	5
E.B.D.S	14,8	15,1	15,3	16,1
I.M.C	15,0	14,9	14,7	14,4
E.M.C	15,0	14,8	14,6	14,3
Dynamic	15,7	15,7	15,7	15,7

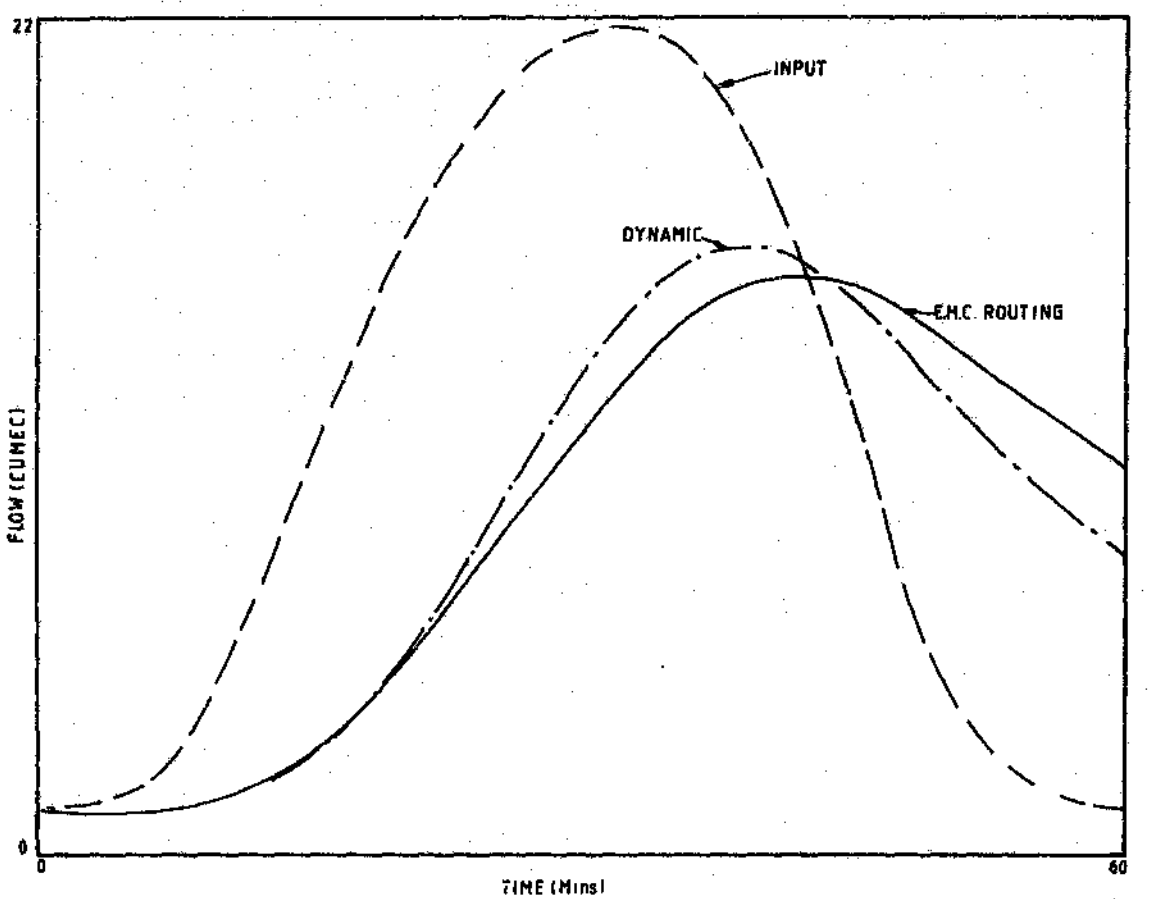


Figure 3.3. Trapezoidal Channel Routing

Table 3.2 shows that the E.B.D.S for the shorter steeper channel goes unstable relatively easily and short time steps are required to ensure that the Courant criteria for stability is adhered to. On the other hand the Muskingum-Cunge formulation remained stable although not as accurate as the E.B.D.S for the channels considered. There is little to choose between the results produced by the I.M.C and the E.M.C schemes. A plot of the input and output hydrographs and the hydrograph produced using the E.M.C scheme for the flat plane are presented in fig. 3.3. As the E.M.C scheme is a faster algorithm than that of the I.M.C scheme and because of the stability of the M.C formulation, the E.M.C scheme will be adopted in WITSKM.

3.5 Pipe Routing

In order to apply the kinematic equations to the routing of flow through pipes, the area, wetted perimeter, and top width are expressed in terms of the angle σ as shown in Fig 3.4.

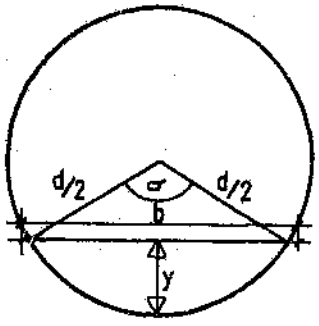


Figure 3.4 Pipe cross section

The relationships are given below in equations 3.9 to 3.13.

$$\sigma = 2 \text{ ARCTAN } \left\{ \frac{(dy - y^2)^{1/2}}{(d/2 - y)} \right\} \tag{3.9}$$

$$y = \frac{d}{2} \left[1 - \cos \left(\frac{\sigma}{2} \right) \right] \tag{3.10}$$

$$A = \frac{d^2}{4} \left[\frac{\sigma}{2} - \sin \left(\frac{\sigma}{2} \right) \cos \left(\frac{\sigma}{2} \right) \right] \tag{3.11}$$

$$P = \frac{\sigma}{2} d \tag{3.12}$$

$$b = 2(dy - y^2) \tag{3.13}$$

To apply the E.B.D.S, the angle σ is calculated using equation 3.9 and the flow depth y_4. The angle σ can then be used to calculate the area A, wetted perimeter P, and the hydraulic radius R for use in the Manning equation to calculate Q_4. For M-C routing the celerity c is required. To do this the relationship $dQ/dA = (dQ/d\sigma)(d\sigma/dA)$ is used together with the equations 3.9 to 3.13 in differentiating the Manning equation with respect to σ to give:

$$C = \frac{\sqrt{S_o}}{n} \left[R^{m-1} + (m - 1)AR^{m-2} \left[\frac{2}{d\sigma^2 \tan\left(\frac{\sigma}{2}\right)} + \frac{1}{d\sigma} \left(2 - \frac{1}{\sin^2\left(\frac{\sigma}{2}\right)} \right) \right] \right] \tag{3.14}$$

To compare the schemes a 1,83m diameter pipeline 1100m long having a slope of 0,0007 and a Manning n of 0,01 was used together with the input and output hydrographs as used by Sevuk in his experiments. The results are presented in Table 3.3

Table 3.3 : Peak Flows(cumec) for Pipes

	1100m LONG PIPE			
dt(mins)	1	2	3	4
E.B.D.S	1,48	1,49	1,51	1,54
I.M.C	1,50	1,50	1,50	1,50
E.M.C	1,50	1,50	1,50	1,50
Sevuk	1,55	1,55	1,55	1,55

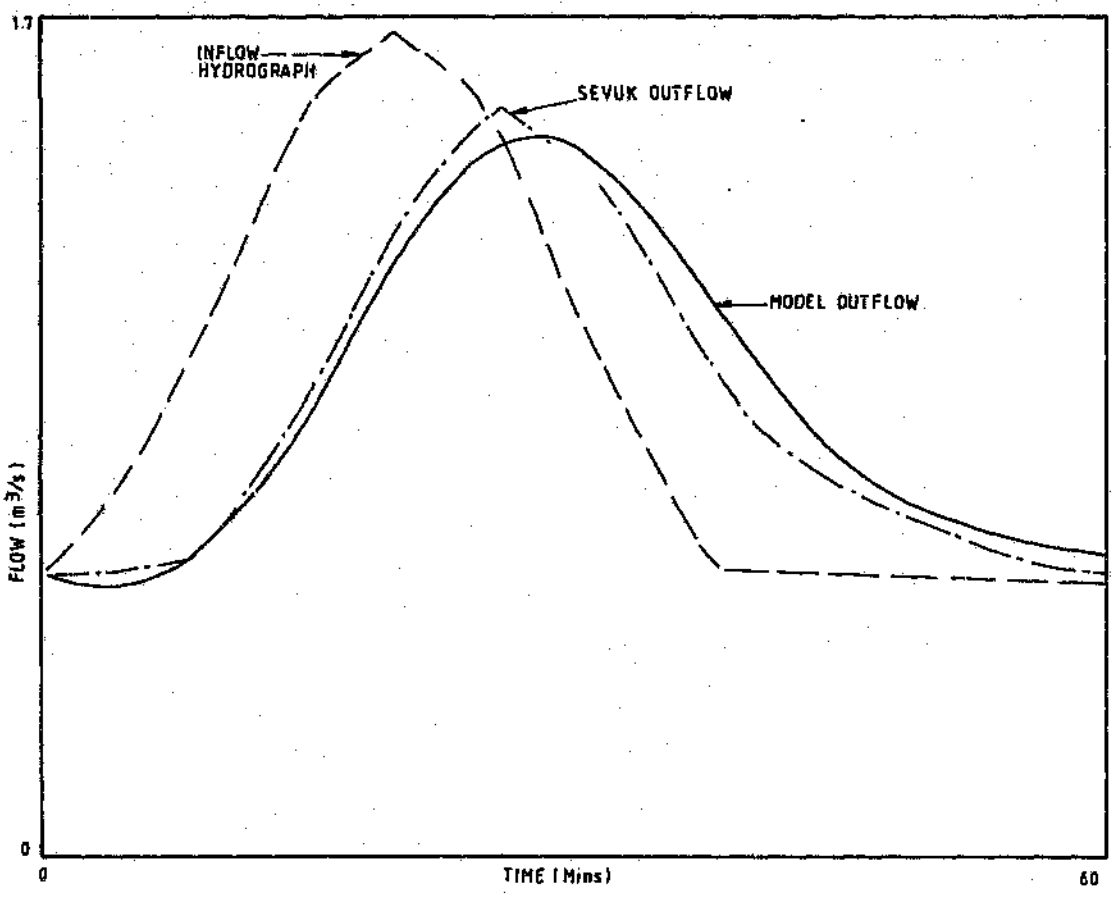


Figure 3.5. Pipe Routing

The pipe tested by Sevuk had a relatively small slope and was long so stability problems were not experienced when using the E.B.D.S. However when the pipe length was subdivided into sub lengths problems were experienced with stability at the larger time intervals. The peaks produced by the schemes are of similar magnitude with the I.M.C and the E.M.C giving the same results. In fig 3.5 plots of the input hydrograph and the output hydrographs for the different schemes are shown plotted. The E.M.C scheme will be used in WITSKM.

3.6 Aquifer Routing

Consider fig 3.6 showing flow through a block of soil of length L, width w, and of depth d having a water table of depth y. The soil is considered to be homogeneous and isotropic with a saturated hydraulic conductivity K_{sat}, saturated moisture content (porosity) t_{sat}, and an initial moisture content t_i.

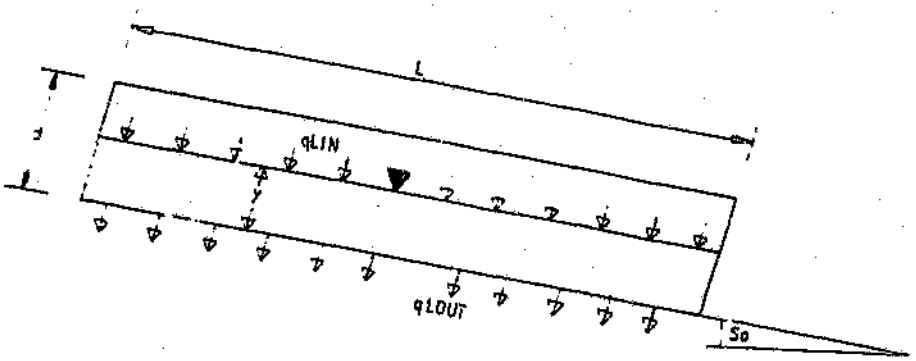


Figure 3.6. Definition sketch aquifer routing

The moisture contents are expressed as a volume ratio. q_{lin} and q_{lout} are lateral inflows due to seepage into and out of the water table respectively. This could be representative of a perched aquifer situation. The continuity equation for the routing of the flow in the saturated zone is given by:

$$\frac{\partial Q}{\partial x} + (t_{sat} - t_i)\frac{\partial A}{\partial t} = qLIN - qLOUT \qquad (3.15)$$

This can be expressed in the same finite difference form as given in equation 2.15. The E.B.D.S as used in the original WITSKM was carried over to the new version. The cross sectional area of the flow is given by w.y and the top width b of the flow is w. Substituting into 2.18, the flow depth y_4 is given by:

$$y_4 = y_3 - \frac{(Q_3 - Q_1)\Delta t}{\Delta x\, W(t_{sat} - t_i)} + \frac{(qLIN - qLOUT)}{W(t_{sat} - t_i)} \qquad (3.16)$$

The Darcy equation which describes saturated flow through homogeneous porous media is given by:

$$v = -k_{sat}\frac{dh}{dx} \qquad (3.17)$$

where v is the velocity of flow, h is the hydraulic head, and dh is the change in head in the direction of flow dx. The slope dh/dx is taken as the slope of the soil surface S_0 and the flow rate Q is therefore given by:

$$Q = wyK_{sat}S_o \qquad (3.18)$$

The depth y_4 calculated using 3.16 is used in equation 3.18 to give the flow rate Q.

3.7 Storage Routing

The approach adopted for the routing of flows through a reservoir is similar to that proposed by Puls (Wilson, 1974). Equation 2.1 can be re-written by putting $(\partial A/\partial t)dx = \partial S/\partial t$ where S is the volume or storage in the element which gives:

$$\frac{\partial C}{\partial x} + \frac{1}{dx}\frac{\partial S}{\partial t} = q_L \qquad (3.19)$$

Ignoring seepage from or rain onto the reservoir i.e $q_1=0$, the finite difference form of equation 3.19 is

$$\frac{\phi(Q_3 - Q_1) + (1 - \phi)(Q_4 - Q_2)}{\Delta x} + \frac{\beta(S_4 - S_1) + (1 - \beta)(S_4 - S_3)}{\Delta x \quad \Delta t} = 0 \qquad (3.20)$$

Putting $\phi=.5$ and $\beta=0$ the method of Puls for level pool reservoir routing can be arrived at. In WITSKM a simpler method is adopted where $\phi=0$ and $\beta=0$ is used in equation 3.20 and the following equation is obtained

$$Q_2\Delta t + S_3 = S_4 + Q_4\Delta t \qquad (3.21)$$

The values of the terms on the right hand side of equation 3.21 are known and the methodology of the storage-indication method of Puls can be applied. The relationship between the storage and the discharge from the dam can be used to solve for S_4 and Q_4 such that equation 3.21 is satisfied.

The relationship between the storage S and the discharge Q in equation 3.21 is the water depth y at the reservoir. Storage can be expressed as a function of the depth using the following expression:

$$S = ay^b \qquad (3.22)$$

For the discharge the type of outlet configuration used in the design of the reservoir will determine the relationship between the discharge

Q and the depth y. Consider the storage reservoir shown in fig. 3.7 which has a bottom outlet and a spillway.

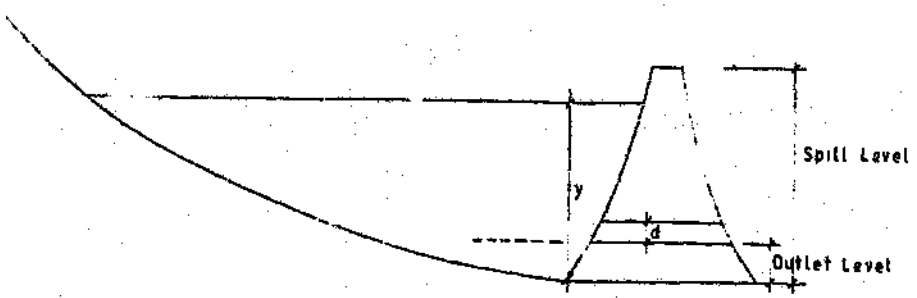


Figure 3.7 Definition sketch of storage facility

The reservoir can be divided into 4 zones viz:

1 y < outlet level

For this zone Q=0. The inflow to the reservoir will go into storage until the outlet level is reached.

2 outlet level < y < 1.5d + outlet level

The flow through an outlet can be considered to be open channel if the outlet is flowing partly full. The characteristics of the flow are complicated because the flow is controlled by many variables such as inlet geometry, slope, size, roughness, approach, and tailwater conditions. The outlet will only be submerged if the depth of the headwater is above a critical value as long as the

downstream end of the outlet is not submerged. The critical value varies between 1.2d and 1.5d where d is the depth or diameter of the outlet. Chow (1959) suggests the upper limit of 1.3d as computations have shown that where submergence is uncertain greater accuracy is obtained by assuming the entrance not to be submerged. In WITSKM therefore the range of depths for unsubmerged flow through the outlet is taken as the outlet level to 1.5d above the outlet level. For unsubmerged flow, the flow Q through the outlet will be that of a weir and can be described by an equation of the form:

$$Q = C_1(y - \text{outlet level})^{\exp_1} \qquad (3.23)$$

3 outlet level + 1.5d < y < spillway level

In this zone the outlet will be submerged and the flow can be considered to be orifice flow. This can also be described by an equation of the form 3.24

$$Q = C_2(y - \text{outlet level} - \frac{d}{2})^{\exp_2} \qquad (3.24)$$

4 y > spillway level

The discharge in this zone is over the spillway and through the outlet. The equation describing the discharge is a combination of the orifice equation given by 3.24 and an equation describing the flow over a spillway which is given by:

$$Q = C_2(y - \text{outlet level} - \frac{d}{2})^{\exp_2} + C_3(y - \text{spill level})^{\exp_3} \qquad (3.25)$$

In equation 3.22 the terms S_4 and Q_4 can be written in terms of reservoir depth y using equations 3.23 to 3.25 depending on the zone. The Newton-Raphson technique is then used to solve for the depth y from which the discharge and storage can be calculated.

CHAPTER 4 GENERAL DESCRIPTION OF THE PROGRAM

4.1 General Background

The modular approach was adopted in WITSKM because of the flexibility that can be achieved in modelling a catchment and for the analysis of stormwater management strategies the different hydrological units can easily be linked together to model the required stormwater policy. The modules that are available are overland flow planes (permeable or impermeable), aquifers, pipes, trapezoidal channels, and storage basins. The routing methods used in WITSKM for the different modules are described in Chapter 3. With the inclusion of aquifer modules in WITSKM the process of interflow and subsurface flow, which can be the dominant runoff process, particularly in a rural type catchment, can be modelled. This increases the area of application of the program not only to urban catchments but to rural catchments as well. The routing of subsurface flow also allows for the modelling of the recession limbs of hydrographs. However, even though groundwater can be modelled, WITSKM remains essentially a single event model as there is no evaporation component and the aquifer module in its present form caters only for the routing of subsurface flow once the aquifers are saturated i.e the changes in moisture content during infiltration and drainage are not modelled. An attempt has also been made to make the model as physically based as possible. For this reason the Green and Ampt infiltration model has been used in the program as this is based on physically measurable parameters as opposed to the Horton type approach which is empirical. The infiltration and interflow aspects of the program are given in more detail in Chapter 5.

4.2 Model Description

The program has been written in the BASIC programming language for use on IBM compatible micro-computers. A structured approach has been adopted in developing the program i.e the program being developed in a number of sub-programs which are linked together. The original version

of WITSKM used a system of menus to drive the program to make the arduous task of data input, editing, and output as easy as possible for the user. This editor was adapted to interface with the revised computational sub-program. The editor is used for the:

- Entering of new data
- Editing of existing data
- Creating of duplicate data files
- Starting of the running of the computational sub-program
- Output of results to screen or printer
- Saving of data files

To apply the model, the catchment is subdivided into overland flow modules by considering the land-use type and topography. The overland flow planes can either be linked by conduits or connected together to form a cascade of planes. A flowchart detailing the main processes of the model is given in Fig. 4.1

4.3 Connectivity

Each module used to model a catchment is assigned a module number chosen by the user. In addition for each module, the module number of the downstream module and in the case of conduits, the module to which overflows should be routed must also be provided. In the case of overland flow planes and aquifers, the module number to which infiltration water must be routed is also required. In many cases information on the depths and permeabilities of the layers making up the soil profile will not be known and in this case a dummy aquifer of infinite depth is created by the program so that the information required for the infiltration routines can be entered. The module numbers used for the dummy aquifers are from 900 upwards. To make the program easy to use, a connectivity routine has been written so that the modules can be entered in any order. The routine sets up a connectivity matrix and determines the order in which the modules should be calculated so that the modules are computed from the most

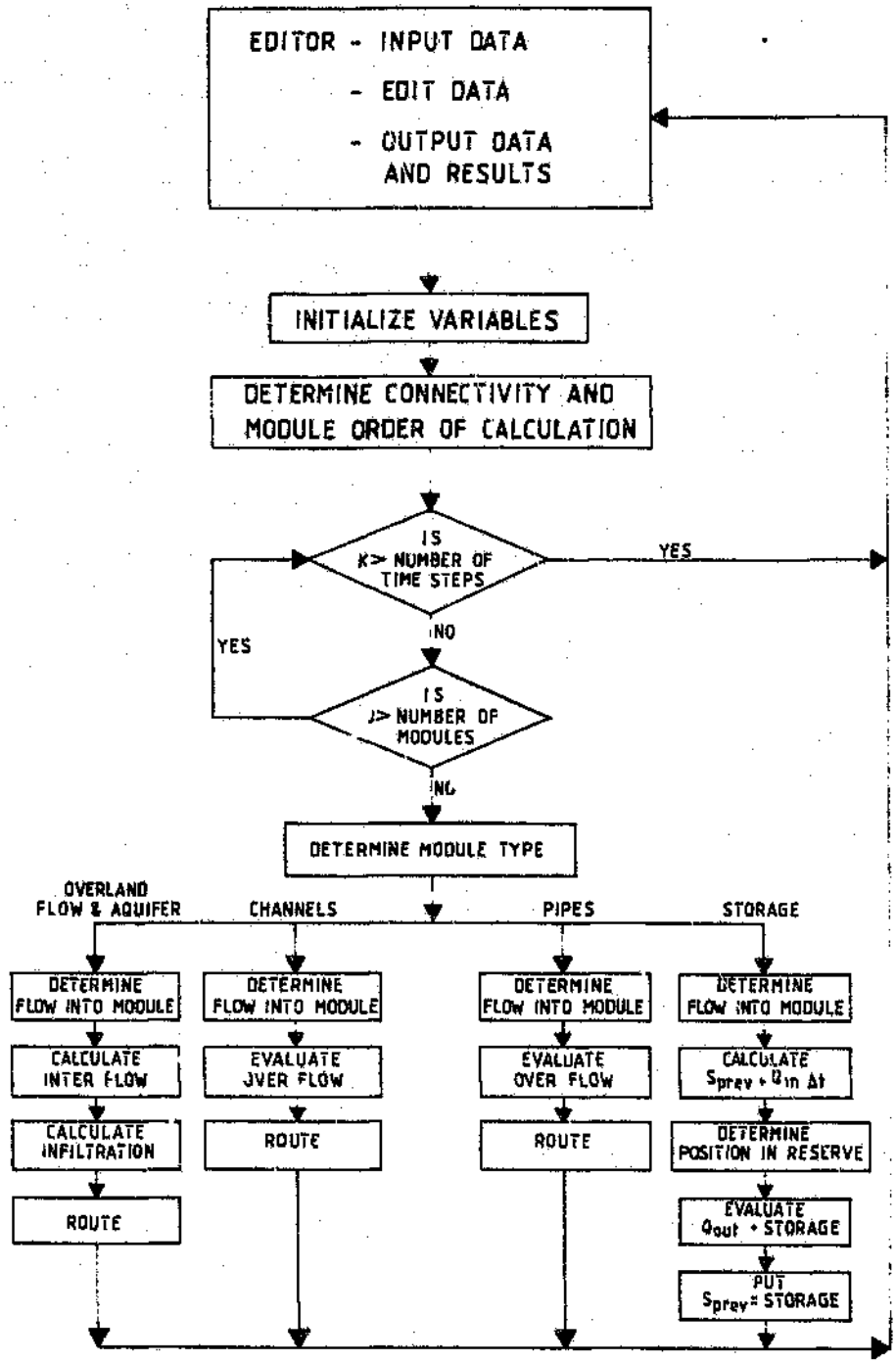


Figure 4.1 Program flow chart

upstream module down to the catchment outlet.

The calculation procedure proceeds through all the modules for each time step. Using the connectivity matrix, the total inflow to a module from all upstream and surcharging modules is calculated. If the module is a conduit or an aquifer, the inflow is checked against the flow capacity of the module. If the capacity is exceeded, the excess flow value is stored in a matrix for later addition to the inflow to the module on to which the module overflows.

CHAPTER 5 INFILTRATION AND INTERFLOW

5.1 Introduction

In the 1930's Horton put forward his classical model on so called hillslope hydrology. Horton hypothesised that the soil surface acts as a sieve which has the ability to separate rainfall into two basic components. The one component, for rainfall intensities exceeding the infiltration capacity of the soil, goes via overland flow to the stream channels, while the other goes through groundwater flow to the stream channels or is returned to the air by evaporation. Horton recognized, that with prolonged rain, the infiltration capacity of a soil would decrease asymptotically with time and the following equation using a negative exponential decay function was proposed.

$$f = f_c + (f_o - f_c)\, e^{-kt} \qquad (5.1)$$

where
- f Rate of instantaneous infiltration (mm/h)
- f_c The limiting, steady minimum infiltration rate (mm/h)
- f_0 The initial maximum infiltration rate at the start of the storm
- k decay constant or shape factor for a given soil
- t Time from beginning of storm

Simply stated, Horton's infiltration theory of runoff predicts that a basin having a uniform initial infiltration capacity will, if the rainfall intensity is greater than the lower limiting infiltration capacity, produce overland flow more or less simultaneously over all the basin after an initial abstraction for depression storage. The surface runoff or overland flow was considered to be the sole contributor to the storm-runoff hydrograph peak.

Kirkby (1978) reports on observations by Tischendorf (1969), Rawitz et al (1970) that Hortonian overland flow was not observed on a catchment having a good vegetative cover. Similar observations have been made by

Mulder (1984) on the experimental catchments in Zululand in Natal. In these catchments the processes of interflow, and subsurface flow play an important role. Thus to model this type of catchment correctly and to be able to model soakaways as a stormwater management strategy, aquifer modules were included in WITSKM. These modules can be stacked under an overland flow module to form a cascade of aquifers to represent the different soil layers. The method used to model infiltration is that of Green and Ampt (1911). This model of the infiltration process is preferred to that of Horton as the parameters are physically based and can be measured in the laboratory.

5.2 The Green-Ampt Model

The Green-Ampt infiltration model is a simple, physically based, infiltration equation which can be derived by the direct application of Darcy's law under the following assumptions:

1. A distinct piston wetting front exists.
2. The hysteresis effects in the soil properties are negligible.
3. Suction at the wetting front (Ω) remains essentially constant regardless of of time and depth.
4. Below the wetting front the soil moisture content remains unchanged from its original value m_i.
5. The soil is uniformly wet above the wetting front and of constant hydraulic conductivity K_{sat}

Darcy's law can be expressed as

$$v = -K_{sat} \frac{\partial h}{\partial z} \qquad (5.2)$$

where
- v is the velocity of flow in the z direction
- h is the piezometric head
- z is the vertical dimension

Referring to fig 5.1, the piezometric head h is given by

$$h = -\Omega - z \tag{5.3}$$

Because of the assumption of the piston wetting front equation 5.2 using 5.3 can be written as

$$v = k_{sat} \frac{z + \Omega}{z} \tag{5.4}$$

From continuity, assuming the fluid to be incompressible, the infiltration rate f must equal the average velocity of the wetting front. From fig 5.1, the volume of water entering the soil, F , is given by:

$$F = (m_{sat} - m_i) z \tag{5.5}$$

in which m_{sat} is the saturated moisture content of the soil taken to be equal to the porosity, and m_i is the initial moisture content of the soil. Substituting 5.5 into equation 5.4 and realizing that f=dF/dt, the following equation results:

$$\frac{dF}{dt} = K_{sat} \left(1 + \frac{(m_{sat} - m_i)\Omega}{F}\right) \tag{5.6}$$

This is essentially the Green-Ampt equation which can be integrated to yield where $\infty = \Omega(m_{sat}-m_i)$

$$F - \infty \ln\left(1 + \frac{F}{\infty}\right) = K_{sat} t \tag{5.7}$$

Li and Rogers(1976) found an explicit formulation for the incremental infiltration volume F, during an incremental time interval t, given by

$$\Delta F = \frac{-(2F_t - K_{sat}\Delta t)}{2} + \frac{[(2F - K_{sat}\Delta t)^2 + 8K_{sat}\Delta t(\infty + F)]^{1/2}}{2} \tag{5.8}$$

where F_t is the volume at time t.

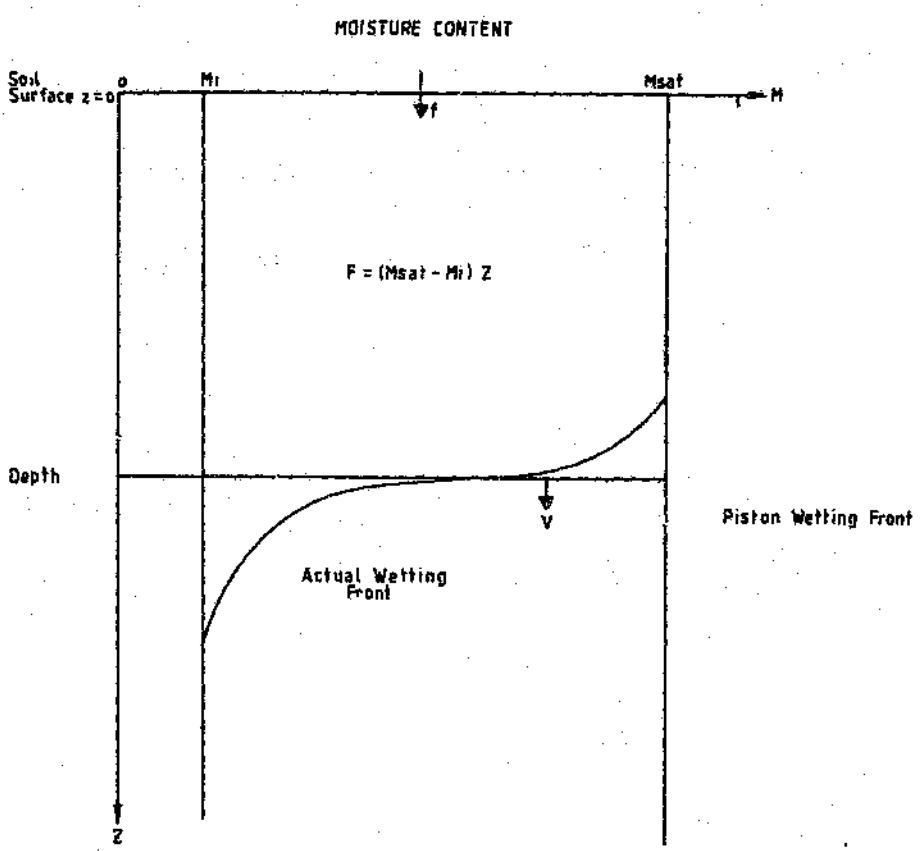


Figure 5.1. Definition sketch for Green-Ampt infiltration model

The average infiltration rate f for the time interval t is then given by:

$$f = \frac{\Delta F}{\Delta t} \qquad (5.9)$$

Equations 5.7 and 5.8 are used to determine the the rainfall excess to be routed off the overland flow plane.

5.3 Subsurface Flow and Interflow

The most usual soil profile that causes subsurface flow is that of a reduction in permeability and porosity with depth which results in a perched aquifer being formed just below the ground surface. This can be represented using a cascade of aquifer modules underlying an overland flow module. This is shown in fig 5.2. There could be cascades of modules upstream of the cascade shown in fig 5.2 depending on the catchment topology. The module numbers are used to specify the module number of each module in the cascade and the modules to which the infiltration water from the modules must be routed. Because the modules can be entered in any order, a routine has been included in the program to group the overland flow modules and their associated aquifers into cascades.

The methodology is based on the following:

1. Subsurface flow does not occur in an aquifer until saturation occurs i.e. the wetting front reaches the bottom of the aquifer, unless a water table level is specified for that aquifer.
2. The rainfall intensity exceeds the hydraulic conductivity of the soil so that the soil can be saturated and a wetting front formed as assumed in the Green-Ampt formulation.

The approach adopted is to calculate the inflow to each module in a cascade starting with the lowest module in the cascade. If the inflow exceeds the capacity of the aquifer module, the excess is added to the inflow of the module above. The potential infiltration from each of the modules is calculated using the Green-Ampt infiltration model based on the status of the wetting front in the underlying aquifer module. The potential infiltration is checked against the available water for infiltration, to determine the actual infiltration rate or the lateral flow out of the module. In the case of the overland flow module the lateral flow into the module is the rainfall intensity and for aquifers the infiltration rate from the module above. Once the lateral flows and the upstream inflow into the modules have been determined, the routing procedures described in chapter 3 are applied. In the case of an aquifer with no upstream inflow or initial water table level the routing procedures are only applied once the aquifer is saturated.

The interflow can occur in two ways:

1. When the inflow to an aquifer module exceeds the capacity of the aquifer the surplus is added to the module above. In this way flow can be returned to the surface as overland flow.
2. If after routing the upstream inflow and lateral flows through an aquifer module, the outflow from the module exceeds the capacity of the module the excess is added to the outflow of the module above.

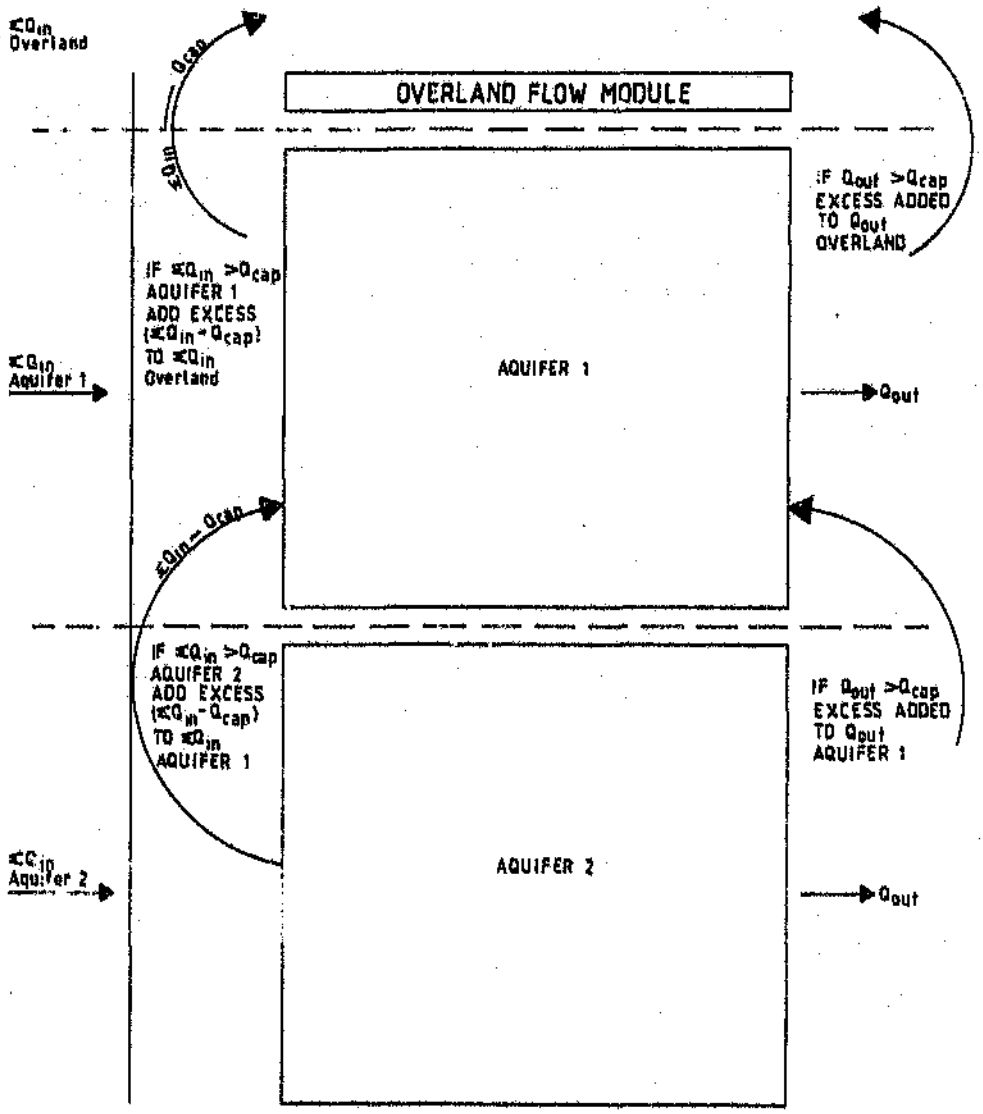


Figure 5.2 Definition Sketch of Interflow Process

CHAPTER 6 CALIBRATION AND VERIFICATION OF WITSKM

6.1 Introduction

For the calibration and verification of WITSKM, three catchments having different land-uses, ranging in size from 0,257 ha to 75 ha were used. The catchments were Newark Street, Sunninghill Park, and Zululand WIMI7. For each catchment, recorded storm events having different durations and intensities were used together with the recorded runoff for calibration and verification. The events used are given in Appendix A. In the case of Newark Street and Sunninghill Park comparisons were made with runs done using WITWAT. The same catchment discretisation was used and for comparison purposes a number of goodness-of-fit criteria described in Appendix B were used. The following rules were adhered to in determining the model parameters during calibration:

(a) WITSKM is a deterministic model and the parameters used have physical significance. The parameter values chosen were therefore kept within a range representative of field conditions

(b) The value of a parameter should not be varied at will without reference to its relationship to other model parameters

(c) The only parameter that was varied from one event to the other on any watershed was the initial moisture content to allow for initial soil moisture conditions.

A direct comparison of WITWAT and WITSKM is not possible for catchments having permeable areas as WITSKM's infiltration routine (Green-Ampt) differs from that of WITWAT which uses Horton's equation. WITSKM has the capability of simulating subsurface or interflow and where applicable this capability is employed and compared to the results without this component.

6.2 Newark Street

Newark Street is an urban area consisting of a section of street (Newark Street) shown in Fig 6.1. This area was gauged as part of the Storm Drainage Research Project at John Hopkins University. The entire area is 0.257 ha and is considered to be 100% impervious. The catchment was discretized into four subcatchments being sections of the street. The pipes were modelled exactly as they exist and the gutters or kerbs were modelled as channels. The discretization, parameters and topographical data used for the simulation are given in Table 6.1. The parameters were those used in Green and Stephenson (1986).

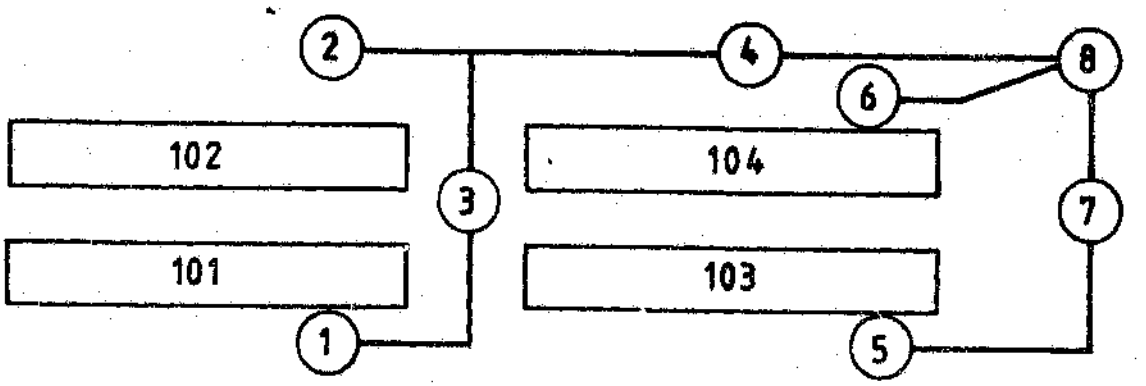


Figure 6.1. Discretization of Newark Street

Page 47 omitted in the pagination.

6.2 Newark Street

Newark Street is an urban area consisting of a section of street (Newark Street) shown in Fig 6.1. This area was gauged as part of the Storm Drainage Research Project at John Hopkins University. The entire area is 0.257 ha and is considered to be 100% impervious. The catchment was discretized into four subcatchments being sections of the street. The pipes were modelled exactly as they exist and the gutters or kerbs were modelled as channels. The discretization, parameters and topographical data used for the simulation are given in Table 6.1. The parameters were those used in Green and Stephenson (1986).

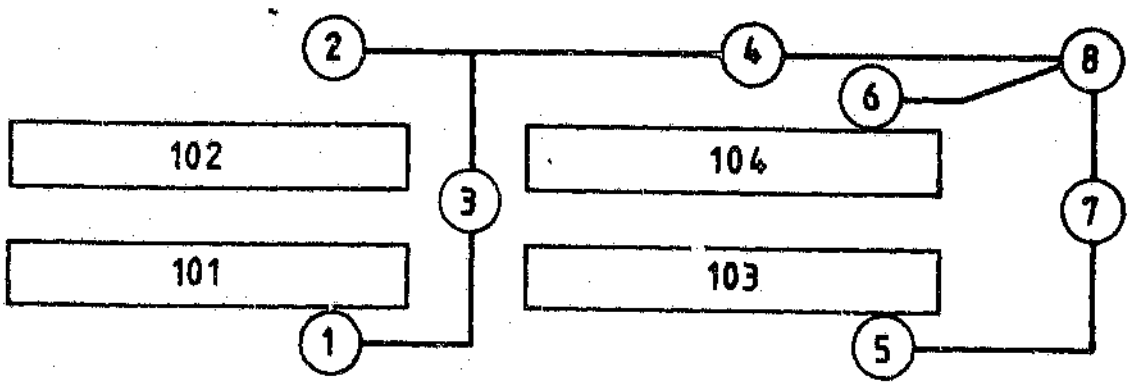


Figure 6.1. Discretization of Newark Street

Page 47 omitted in the pagination.

Table 6.1: Newark street module data

Overland Flow Modules					
Module number	D/s mod number	Length (m)	Width (m)	Slope (m/m)	Roughness (n)
101	1	85,3	7,0	0,03	0,018
102	2	85,3	7,0	0,03	0,018
103	5	98,6	7,0	0,03	0,018
104	6	98,6	7,0	0,03	0,018

Pipe Modules						
Module number	D/s mod number	Ov/flow mod no	Slope (m/m)	Length (m)	Diameter (m)	Roughness (n)
3	4	103	0,030	15	0,45	0,013
4	8	104	0,032	94	0,45	0,013
7	8	0	0,030	15	0,45	0,013
8	0	0	0,030	100	0,45	0,013

Trapezoidal channel modules						
Module number	D/s mod number	Ov/flow mod no	Slope (m/m)	Length (m)	Dimensions widthxheight (m)	Roughness (n)
1	3	101	0,017	40,0	0,30x0,15	0,015
2	4	102	0,017	40,0	0,30x0,15	0,015
5	7	103	0,017	46,0	0,30x0,15	0,015
6	8	104	0,017	46,0	0,30x0,15	0,015

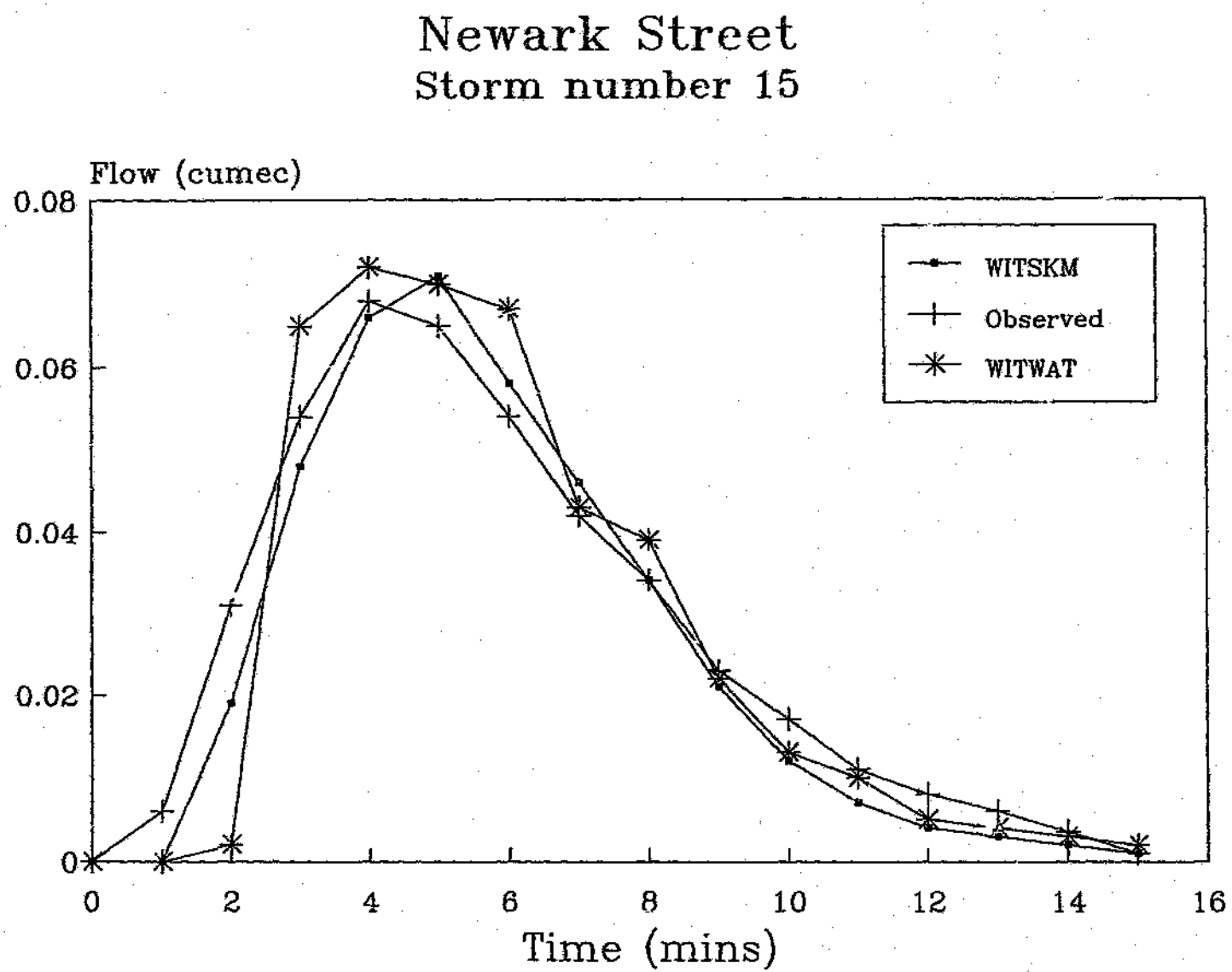


Figure 6.2. Comparison of observed and simulated hydrographs (Newark Street storm no 15)

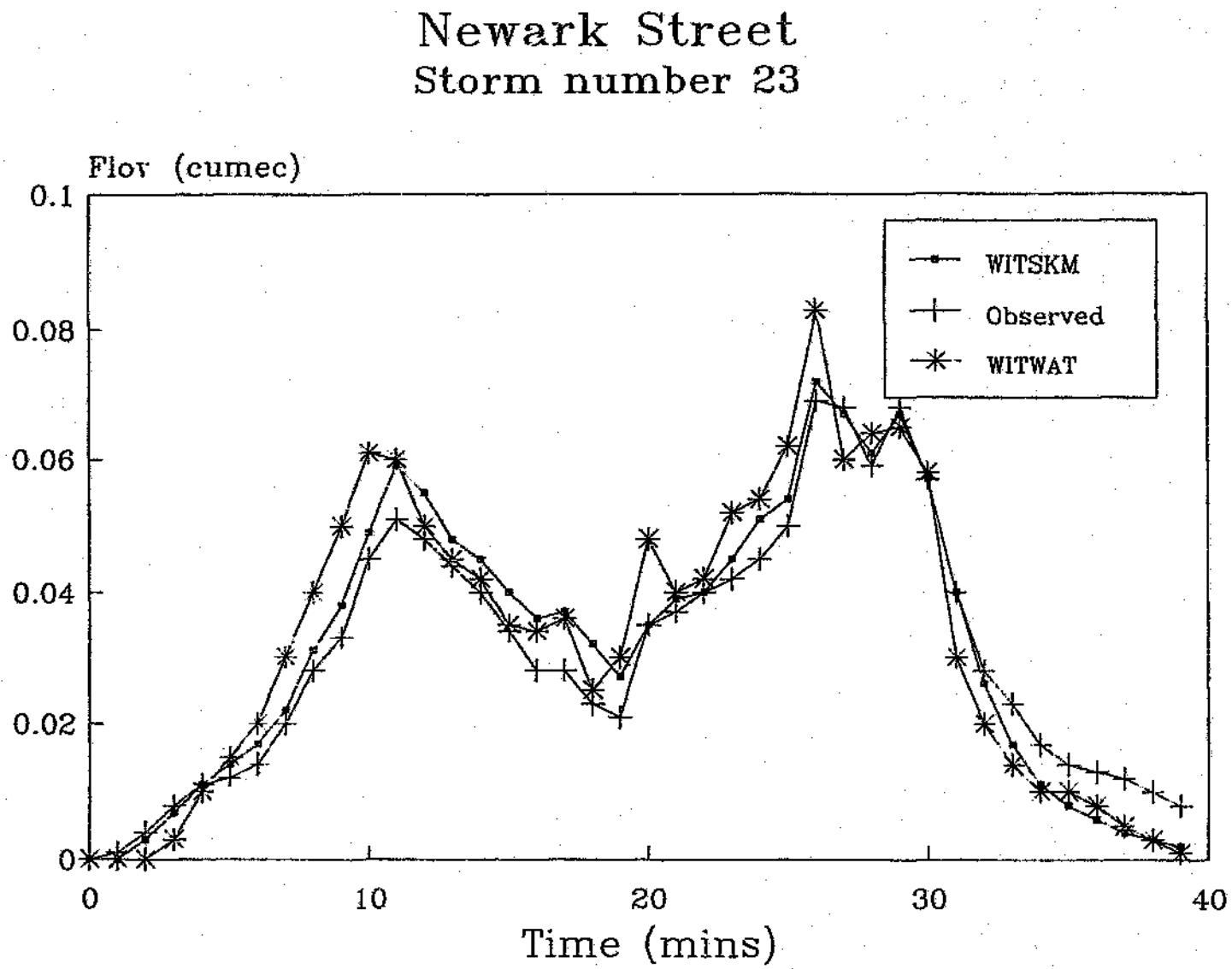


Figure 6.3. Comparison of observed and simulated hydrographs (Newark Street storm no 23)

Table 6.2: Goodness-of-fit Statistics for Newark Street

Storm number 15			
Parameter	WITSKM	WITWAT	Observed
Peak flowrate (m^3/s)	0,071	0,073	0,068
Ratio of peaks (sim/obs)	1,04	1,08	-
% error in simulated peak	4,4	7,8	-
Mean Flowrate (m^3/s)	0,034	0,035	0,035
Ratio of means	0,96	1,00	-
% error in simulated mean	-2,86	0,00	-
Volume of Flow (m^3)	22,80	24,00	23,70
Ratio of volumes	0,96	1,01	-
% error in simulated volume	-3,80	1,27	-
Sum of squared residuals $(m^3/s)^2$	0,0004	0,001	
Sum of absolute residuals (m^3/s)	0,055	0,08	
Coefficient of efficiency	0,93	0,72	
Proportional error of estimate	0,26	0,40	
Sum of absolute areas of divergence	2,60	3,00	

Storm number 23			
Parameter	WITSKM	WITWAT	Observed
Peak flowrate (m^3/s)	0,071	0,080	0,069
Ratio of peaks (sim/obs)	1,03	1,16	-
% error in simulated peak	2,9	15,9	-
Mean Flowrate (m^3/s)	0,036	0,036	0,034
Ratio of means	1,06	1,06	-
% error in simulated mean	5,88	5,88	-
Volume of Flow (m^3)	76,18	76,00	72,40
Ratio of volumes	1,05	1,05	-
% error in simulated volume	5,22	4,97	-
Sum of squared residuals $(m^3/s)^2$	0,0008	0,002	
Sum of absolute residuals (m^3/s)	0,140	0,20	
Coefficient of efficiency	0,92	0,86	
Proportional error of estimate	0,24	0,35	
Sum of absolute areas of divergence	7,90	8,00	

In a previous study on the effect of the level of discretization on the ability of WITWAT to model the runoff from the Sunninghill Park catchment, two storm events recorded on 7 and 9 January 1987 were used. The storm on the 7 January had a duration of 30 minutes, a maximum intensity of 113 mm/h, and a total depth of rainfall of 26,2mm. The previous significant rainfall event was 14 days before on the 23 December 1986. The storm of the 9 January 1987 had a duration of 60 mins, a maximum intensity of 100 mm/h, and a total rainfall depth of 34,2mm. The last rainfall event was that of the 7 January. The initial moisture conditions for the catchment could therefore be expected to be drier for the event of the 7 January.

These storm events together with the discretization and topographical data used for WITWAT (Fig 6.4 and table 6.3) were input into WITSKM. The Green-Ampt parameters were used to calibrate the model for the event of the 7 January 1987 and only the initial moisture content of the soil was varied in the calibration for the event of 9 January 1987. In choosing the Green-Ampt parameters, the guidelines provided by Rawls et al(1966) were used. These guidelines provide estimates of the Green-Ampt parameters based on the soil texture classification procedure of the Soil Conservation Service of the United States Department of Agriculture. The parameters provided by Rawls et al could however only be used for the modules without any impervious areas due to urbanization. For the modules effected by urbanization the saturated hydraulic conductivity was reduced according to the extent of the impervious cover.

Comparisons of the hydrographs produced using WITSKM and WITWAT with the observed hydrograph are presented in Figures 6.5 and 6.6. The statistics of goodness-of-fit are given in Table 6.4. From the plots and the statistics of goodness-of-fit the two models behave similarly. For the calibration for the storm of the 9 January, the initial moisture content was the only parameter varied and the same moisture content of 0,2 gave the best calibration.

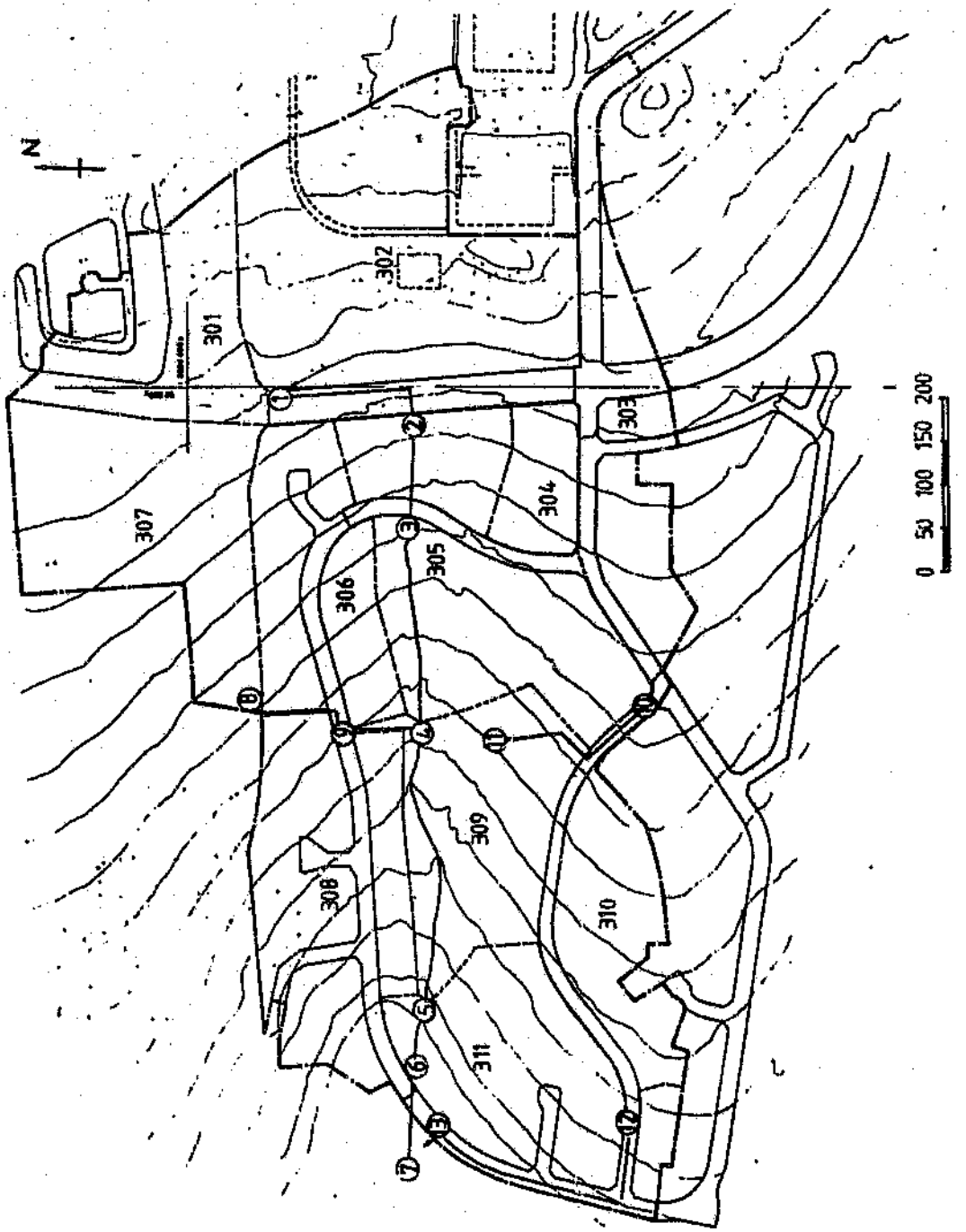


Figure 6.4 Discretization of Sunninghill Park Catchment

Table 6.3: Sunninghill Park module data

Overland Flow Modules					
Module number	D/s mod number	Length (m)	Width (m)	Slope (m/m)	Roughness (n)
301	1	81,0	592,0	0,028	0,156
302	2	152,0	777,0	0,048	0,216
303	10	120,0	399,0	0,038	0,140
304	3	38,0	391,0	0,068	0,020
305	4	105,0	750,0	0,061	0,216
306	4	92,0	509,0	0,067	0,147
307	8	160,0	420,0	0,039	0,121
308	6	74,0	938,0	0,069	0,172
309	5	76,0	694,0	0,069	0,249
310	12	88,0	472,0	0,042	0,163
311	6	81,0	821,0	0,056	0,183

Pipe Modules						
Module number	D/s mod number	Ov/flow mod no	Slope (m/m)	Length (m)	Diameter (m)	Roughness (n)
1	2	305	0,030	170,0	0,60	0,014
8	9	308	0,055	130,0	0,90	0,014
9	4	308	0,055	70,0	0,90	0,014
10	11	309	0,052	190,0	0,525	0,014
12	13	311	0,030	350,0	0,675	0,014
13	7	0	0,050	10,0	0,675	0,014
11	309	0	0,052	100,0	0,600	0,014

Trapezoidal channel modules						
Module number	D/s mod number	Ov/flow mod no	Slope (m/m)	Length (m)	Dimensions widthxheight (m)	Roughness (n)
2	3	305	0,067	120,0	4,80x2,00	0,014
3	4	309	0,055	235,0	6,00x2,50	0,040
4	5	309	0,056	320,0	6,00x3,00	0,040
5	6	311	0,030	70,0	6,00x3,50	0,040
6	7	0	0,054	70,0	3,00x4,50	0,014
7	0	0	0,054	10,0	3,00x5,00	0,014

Table 6.3 (Continued)

Green-Ampt infiltration parameters				
Module number	Hydraulic conductivity (mm/h)	Suction Head (m)	Porosity	Initial moisture content (vol/vol)
301	3,5	0,10	0,40	0,20
302	7,0	0,10	0,40	0,20
303	3,0	0,10	0,40	0,20
304	0,0	0,00	0,00	0,00
305	4,0	0,10	0,40	0,20
306	3,0	0,10	0,40	0,20
307	3,0	0,10	0,40	0,20
308	4,0	0,10	0,40	0,20
309	7,0	0,10	0,40	0,20
310	3,0	0,10	0,40	0,20
311	4,0	0,10	0,40	0,20

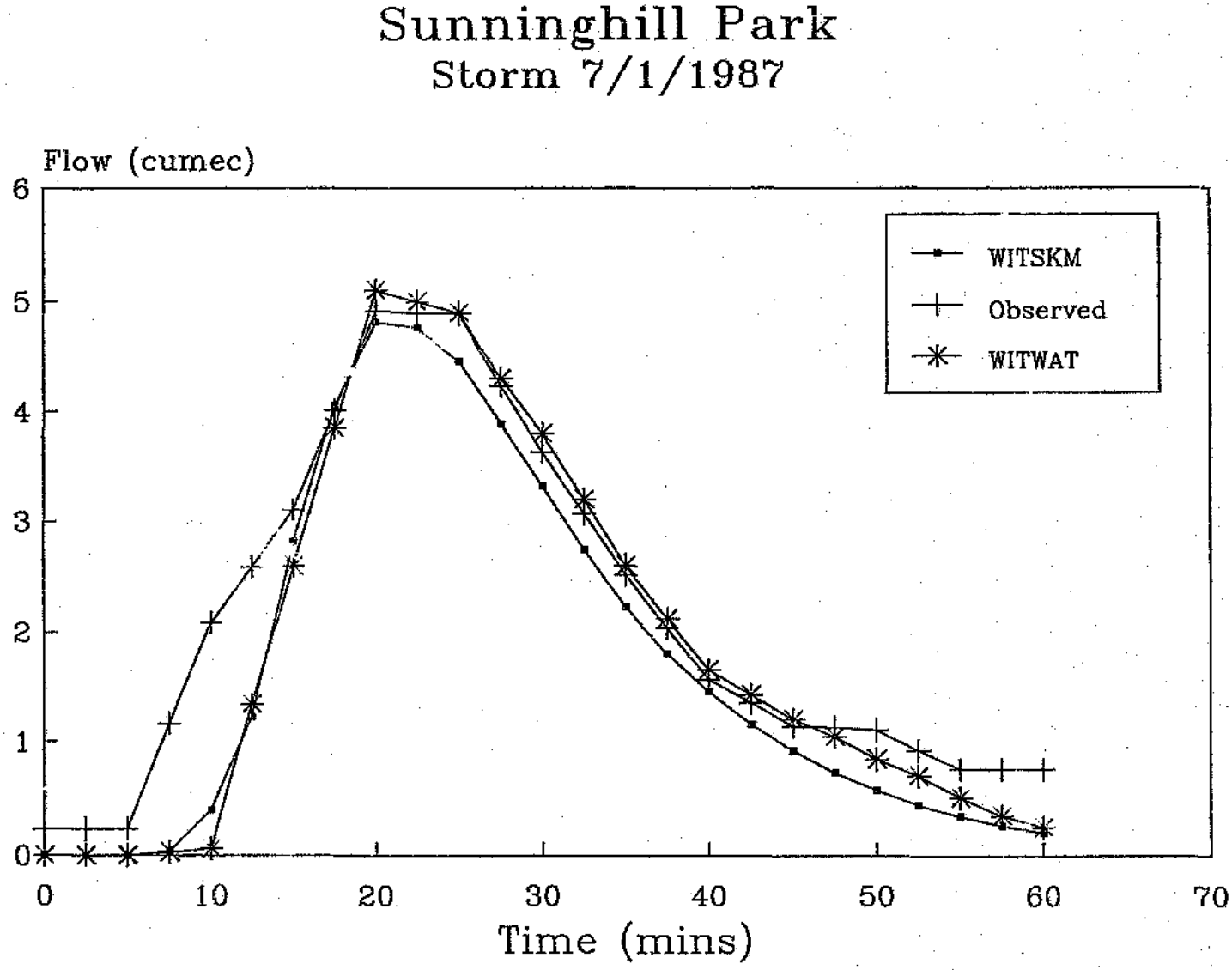


Figure 6.5 Comparison of Observed and Simulated Hydrographs (Sunninghill Park Storm 7-01-87)

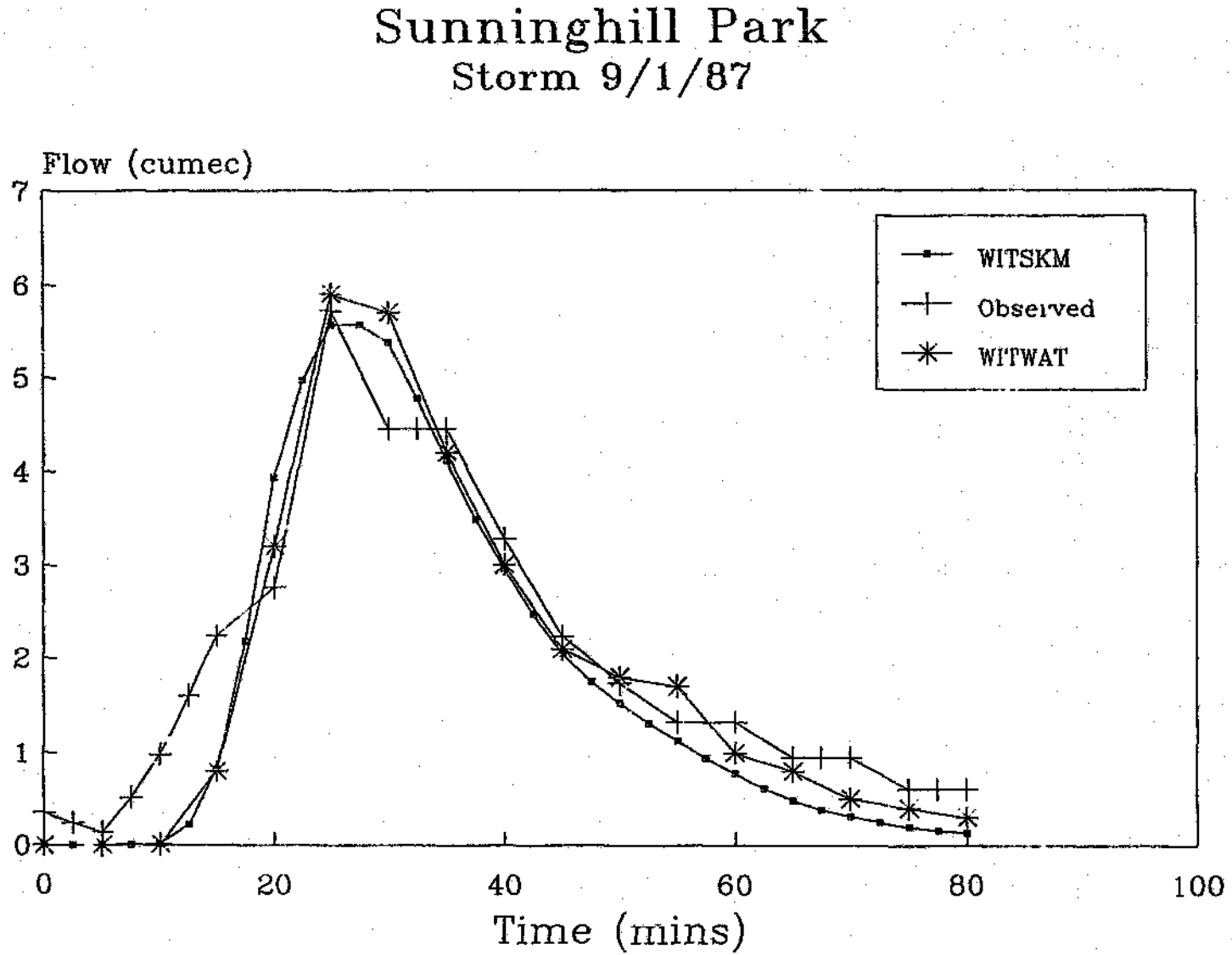


Figure 6.6 Comparison of Observed and Simulated Hydrographs (Sunninghill Park Storm 9-01-87)

Table 6.4: Goodness-of-fit Statistics Sunninghill Park

Storm 7 January 1987			
Parameter	WITSKM	WITWAT	Observed
Peak flowrate (m^3/s)	4,820	5,100	4,920
Ratio of peaks (sim/obs)	0,98	1,04	-
% error in simulated peak	-2,0	4,0	-
Mean Flowrate (m^3/s)	1,706	1,880	2,134
Ratio of means	0,80	0,88	-
% error in simulated mean	-20,0	-12,0	-
Volume of Flow (m^3)	6381,5	7012,7	7927,0
Ratio of volumes	0,81	0,88	-
% error in simulated volume	-19,5	-12,0	-
Sum of squared residuals $(m^3/s)^2$	8,420	8,160	
Sum of absolute residuals (m^3/s)	10,77	8,56	
Coefficient of efficiency	0,86	0,86	
Proportional error of estimate	0,53	0,50	
Sum of absolute areas of divergence	1545	1196	

Storm 9 January 1987			
Parameter	WITSKM	WITWAT	Observed
Peak flowrate (m^3/s)	0,071	0,080	0,069
Ratio of peaks (sim/obs)	1,03	1,16	-
% error in simulated peak	2,9	15,9	-
Mean Flowrate (m^3/s)	0,036	0,036	0,034
Ratio of means	1,06	1,06	-
% error in simulated mean	5,88	5,88	-
Volume of Flow (m^3)	76,18	76,00	72,40
Ratio of volumes	1,05	1,05	-
% error in simulated volume	5,22	4,97	-
Sum of squared residuals $(m^3/s)^2$	0,0008	0,002	
Sum of absolute residuals (m^3/s)	0,140	0,20	
Coefficient of efficiency	0,92	0,86	
Proportional error of estimate	0,24	0,35	
Sum of absolute areas of divergence	7,90	8,00	

6.4 Zululand Catchment W1M17

This Catchment has an area of 66,9 ha and is situated some 130 km north of Durban. Information on the catchments was obtained from Hope and Mulder (1979) and on the soil depths and types as well as infiltration parameters from Mulder (1984). Mulder (1984), after studying the rainfall-runoff characteristics of a number of nested catchments in the Zululand area, concluded that the runoff process in the area was essentially that of subsurface flow. With the details provided on the soils of the area, this catchment was used to evaluate the interflow and subsurface flow procedures of WITSKM. This catchment has been modelled by Green and Stephenson (1986) using WITWAT and difficulty was experienced in modelling the recession limb of the hydrographs. Different final infiltration rates f_c in the Horton infiltration equation also had to be used when modelling events having different initial moisture contents.

In applying WITSKM to the catchment, the overland approach of Horton and a system employing aquifers was used to model the subsurface flow. The infiltration parameters varied quite dramatically from one soil group to another and even within a particular soil group. However asuumptions were made and values that were considered to be representative were selected. The area is largely open veld with marsh areas along the main river courses of the catchment. Two catchment discretizations shown in figs 6.7 and 6.8 were used to model the catchment. The layout shown in fig 6.7 was that used by Green and Stephenson(1986) when using WITWAT to model the catchments. The discretization shown in fig 6.8 is an adjustment of that shown in fig 6.7 to make allowance for the marsh areas along the rivers. The infiltration tests on the alluvium in these areas show the final infiltration rate to be of the order of 650 mm/h and in the rest of the catchment a value of 20 mm/h was considered to be representative. For the subsurface flow case, a cascade of a single aquifer and an overland flow module was used to model the soil profile. The depths of the aquifer was taken as 2,0m in the marsh areas and 0,5m in the rest of the catchment. The module data is presented in Tables 6.5 and 6.6 for the two cases.

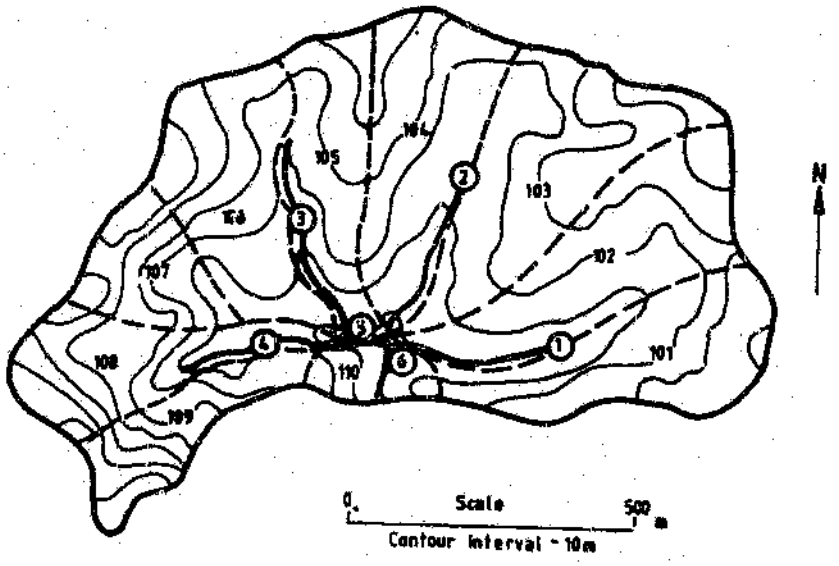


Figure 6.7. Discretization Zululand catchment-WITWAT

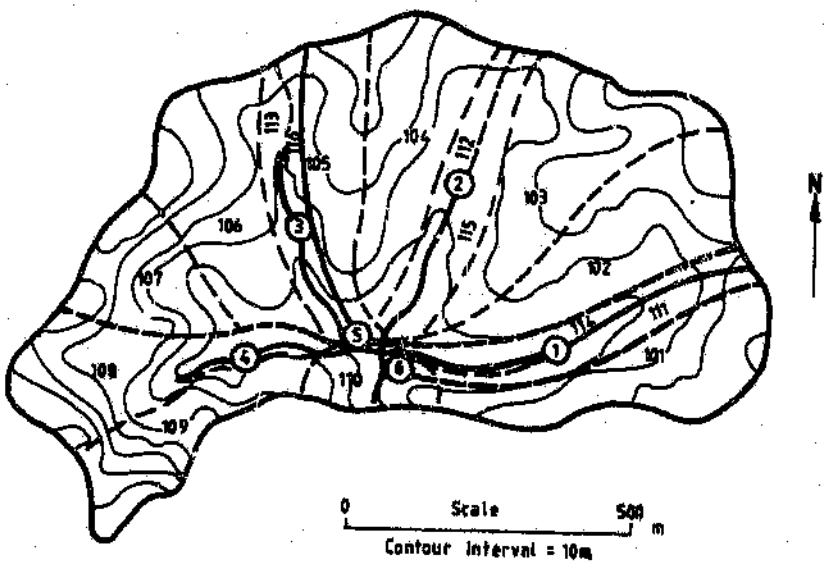


Figure 6.8. Discretization Zululand catchment-subsurface flow

Table 6.5 : Discretization Zululand Catchment - WITWAT

Overland Flow Modules					
Module number	D/s mod number	Length (m)	Width (m)	Slope (m/m)	Roughness (n)
101	1	227,0	387,0	0,132	0,240
102	1	280,0	314,0	0,128	0,240
103	2	550,0	203,0	0,064	0,240
104	2	250,0	352,0	0,100	0,240
105	3	400,0	170,0	0,075	0,240
106	3	410,0	180,0	0,098	0,240
107	4	200,0	185,0	0,200	0,240
108	4	300,0	180,0	0,200	0,240
109	4	150,0	313,0	0,171	0,240
110	5	75,0	186,0	0,100	0,240

Trapezoidal channel modules						
Module number	D/s mod number	Ov/flow mod no	Slope (m/m)	Length (m)	Dimensions widthxheight	Roughness (n)
1	6	0	0,025	350,0	2,00x2,00	0,055
2	6	0	0,040	300,0	1,50x2,50	0,055
3	5	0	0,050	250,0	2,00x3,00	0,055
4	5	0	0,040	100,0	2,00x3,50	0,055
5	6	0	0,041	75,0	2,00x4,50	0,055
6	0	0	0,035	100,0	2,00x5,00	0,055

Table 6.6 : Revised Discretization of Zululand Catchment

Overland Flow Modules					
Module number	D/s mod number	Length (m)	Width (m)	Slope (m/m)	Roughness (n)
101	111	167,0	387,0	0,132	0,240
102	114	210,0	314,0	0,128	0,240
103	115	420,0	203,0	0,064	0,240
104	112	200,0	352,0	0,100	0,240
105	116	300,0	170,0	0,075	0,240
106	113	310,0	180,0	0,098	0,240
107	4	200,0	185,0	0,200	0,240
108	4	300,0	180,0	0,200	0,240
109	4	150,0	313,0	0,171	0,240
110	6	75,0	186,0	0,100	0,240
111	1	60,0	387,0	0,132	0,240
112	2	50,0	352,0	0,100	0,240
113	3	100,0	180,0	0,098	0,240
114	1	70,0	314,0	0,128	0,240
115	2	130,0	203,0	0,064	0,240
116	3	100,0	170,0	0,075	0,240

Trapezoidal channel modules						
Module number	D/s mod number	Ov/flow mod no	Slope (m/m)	Length (m)	Dimensions widthxheight (m)	Roughness (n)
1	6	0	0,025	350,0	2,00x2,00	0,055
2	6	0	0,040	300,0	1,50x2,50	0,055
3	5	0	0,050	250,0	2,00x3,00	0,055
4	5	0	0,040	100,0	2,00x3,50	0,055
5	6	0	0,041	75,0	2,00x',50	0,055
6	0	0	0,035	100,0	2,00x5,00	0,055

Two storms were used viz a storm on the 7 February 1977 and the 9 November 1977. The storm of the 9 November occurred on a dry catchment while that of the 7 February 1977 occurred on a saturated catchment. The duration of the storm of the 7 February 1977 was 9 hours with a peak intensity of 53,3 mm/h while the storm of the 9 November 1977 had a duration of 1,5 hours and a peak intensity of 101,9 mm/h.

When employing the Horton approach of overland flow, WITSKM gave similar results, for both discretizations, to those achieved by Green and Stephenson (1986) using WITWAT. The infiltration parameters for the Green-Ampt model are given in Table 6.7 for the Horton and subsurface approaches. In the subsurface flow case the only parameters varied during calibration were the initial moisture content and the water table depth of the aquifer in the marsh areas along the river banks.

The hydrographs for the two events are compared graphically in figures 6.9 and 6.10 for both approaches shown in fig. 6.7 and 6.8. For the

Table 6.7 : Infiltration Parameters used for Zululand Catchment

Horton Approach					
Event	Perm (mm/h)	Suction (m)	Porosity	Initial moisture content	
7-02-77	2	0,01	0,4	0,395	
9-11-77	13	0,13	0,4	0,080	
Subsurface Flow					
module numbers 101 to 109					
Event	Perm (mm/h)	Suction (m)	Porosity	Initial Moisture Content	Depth W.T(m)
7-02-77	20	0,08	0,40	0,395	0
9-11-77	20	0,08	0,40	0,100	0
module numbers 110 to 116					
Event	Perm (mm/h)	Suction (m)	Porosity	Initial Moisture Content	Depth W.T(m)
7-02-77	650	0,05	0,45	0,44	1,90
9-11-77	650	0,05	0,45	0,38	1,45

event of the 9 November 1977, the hydrograph for the subsurface flow case modelled the recession limb of the hydrograph better than for the Horton approach. The results for the event of the 7 February 1977 were however similar for both the subsurface and Horton approaches

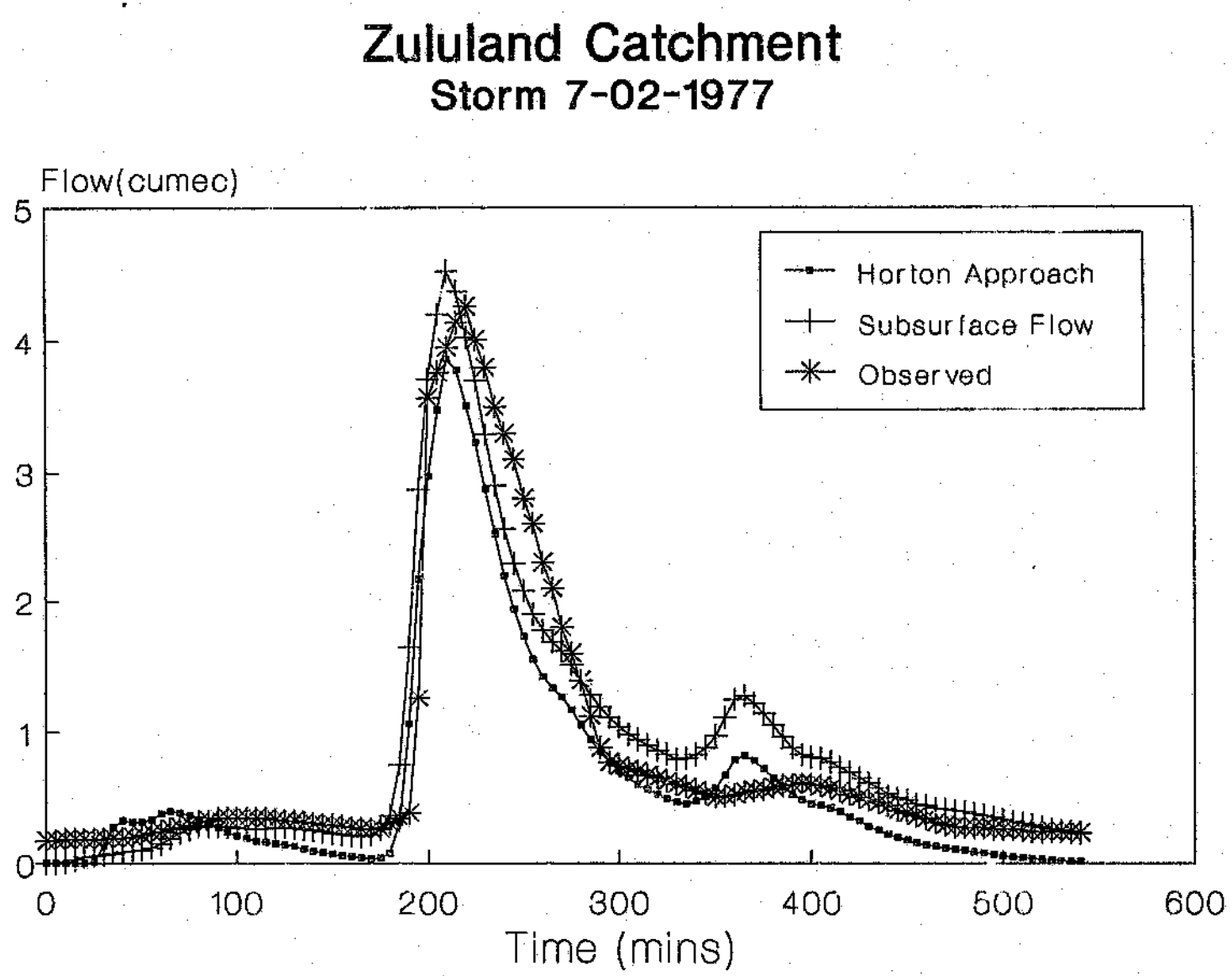


Figure 6.9 : Comparison hydrographs for Zululand Catchment

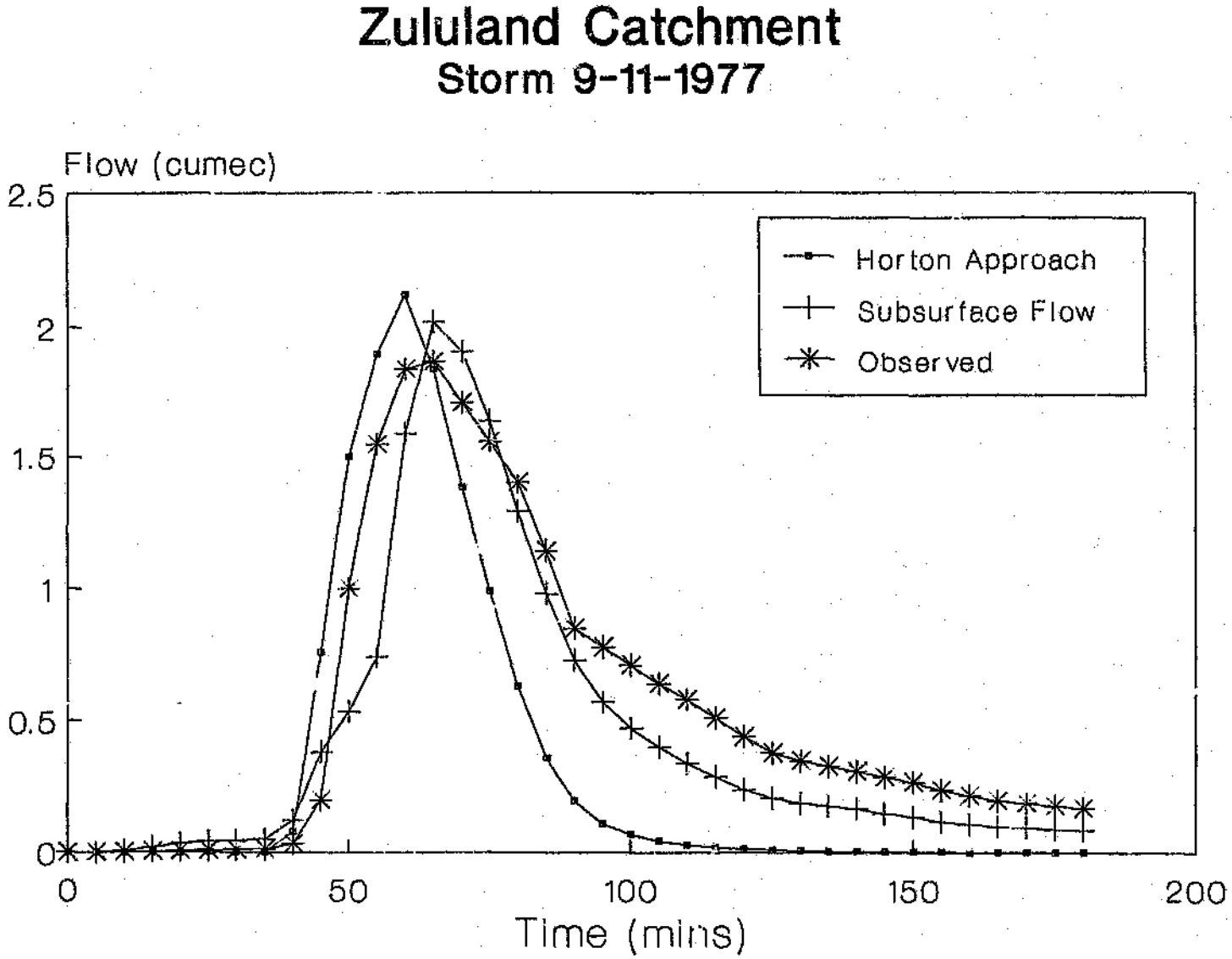


Figure 6.10 : Comparison of Hydrographs for Zululand Catchment

with the subsurface flow comparing with the observed hydrograph more closely for the low flows at the beginning and end of the events. However what is important is that the physical infiltration parameters for the soil were kept the same for both events for the subsurface flow case. Better calibrations could possibly be achieved if more detail on the distribution of the soil depths and infiltration parameters over the catchment was available. The catchment could then be more accurately discretized. A more appropriate model to model the subsurface or variable source area approach would be one that uses a fine grid or pixel system of discretization so that the large variation in the soil parameters over a catchment can be taken into account.

CHAPTER 7 APPLICATION OF WITSKM TO THE WATERVAL CATCHMENT

7.1 Introduction

To ascertain the effectiveness of the different stormwater management options such as dual drainage, use of flood plains, and detention storage ponds, a hypothetical town was laid out on the Waterval catchment north of Johannesburg. This catchment and the neighbouring Sunninghill catchment in Sandton have been monitored by the Water Systems Research Group of the University of the Witwatersrand. Both catchments have a catchment area of 75 ha. The Waterval catchment is at present undeveloped and three rainfall-runoff events have been abstracted from the catchment data for use in analysing the management options for the proposed town. By doing this the runoff generated by WITSKM for the different options tried can be compared to the measured runoff for the catchment in its natural state.

7.2 Description of Proposed Township

The proposed township was laid out along the lines of the Sunninghill development. The town therefore consists of houses, flats, and townhouse developments with a tarred road network. The infiltration and roughness parameters for the various land use types that were obtained when calibrating WITSKM for the Sunninghill catchment can be used on the Waterval town. The layout of the town and the details of the discretization, pipe and road networks are shown in Fig 7.1 and in Table 7.1. The level of the discretization used to model the catchment is coarse with the exact positions of the erven and access roads into the various land use areas not being detailed.

In order to accommodate the possible drainage options a park area that can be used as a flood plain has been included in the township layout. This area runs down the centre of the town and discharges into the natural outlet of the catchment. A natural or lined channel can be included in the park area for the simulation of the dual drainage

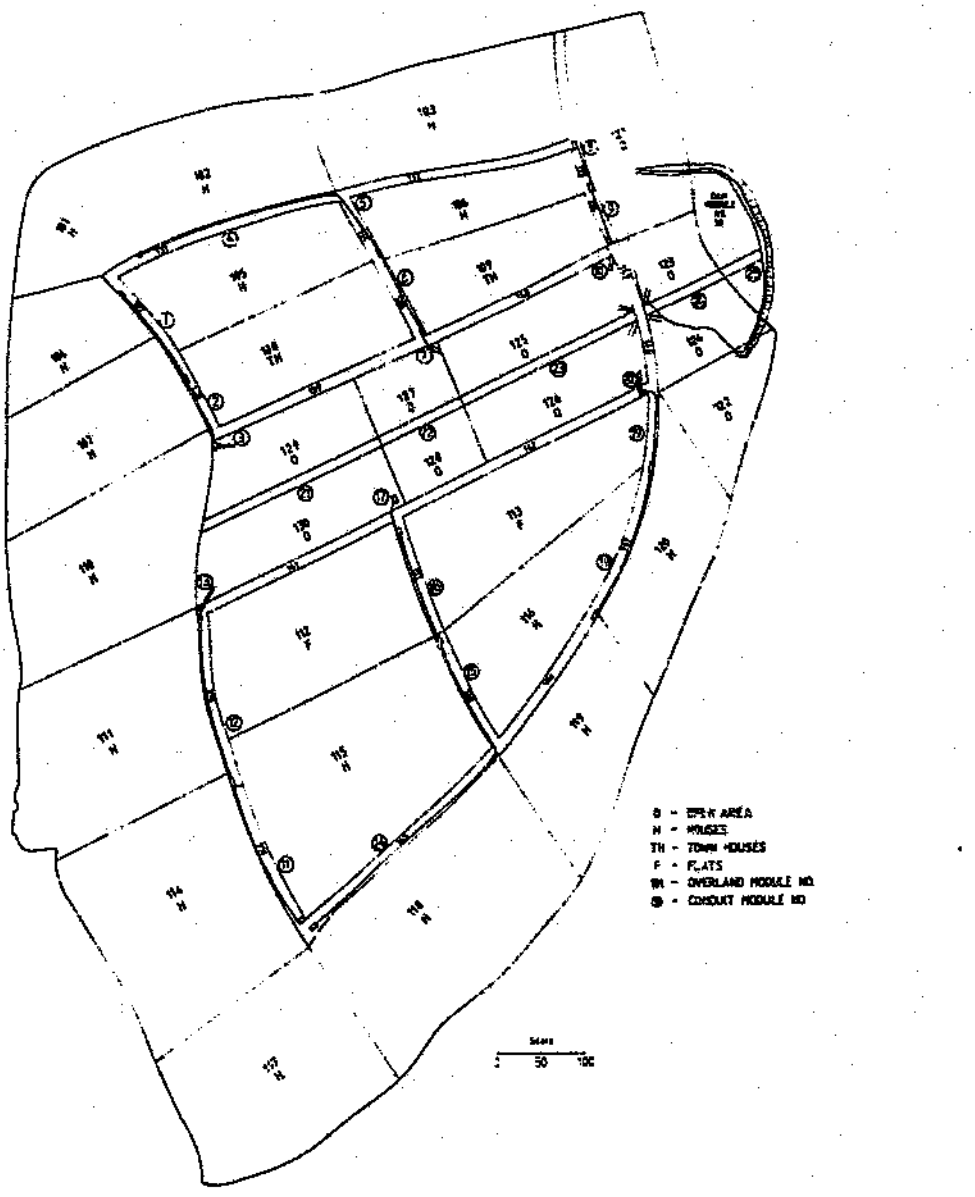


Figure 7.1 Discretization of Proposed Waterval Town

Table 7.1 : Module Numbers for the different Land Uses

Land-Use	Range of Module Numbers
Town Houses	108-109
Flats	112-113
Houses	101-107 110-111 114-120
Open Areas	121-130
Roads	131-151
Pipes	1-20
Channels	21-25

option. The pipe network can also drain into this area at various points along its length. No development has been planned at the outlet of the catchment to make provision for the inclusion of a detention storage pond and sport fields. The sport fields will act as temporary storage and soak-a-way for excess water. The infiltration and roughness parameters used in the model for the various land uses and conduit modules are summarised in Table 7.2

7.3 Storms used in Analysis

For the analysis of the different stormwater management options and for the comparison of the WITSKM output to the recorded output for the catchment in its natural state, three storm events were abstracted from the catchment data. The first event occurred on the 26 September 1987, the second on the 3 February 1987 and the third on the 21 March 1987. Rainfall events 1 and 2 have a triangular shape while event 3 could be considered uniform. All three of the events were recorded at all 5 of the raingauges that monitor the catchment. Thiessen polygons were used to produce an average rainfall record for the catchment for

Table 7.2: Infiltration and Roughness Parameters

Land-Use	Roughness (n)	Perm (mm/h)	Suction (m)	Porosity	Initial Moist Content
Houses	0,14	3,5	0,1	0,40	0,25
T/Houses	0,11	3,0	0,1	0,40	0,25
Flats	0,10	2,0	0,1	0,40	0,25
Open Areas	0,20	7,0	0,1	0,40	0,25
Roads	0,016	0,0	0,0	0,00	0,00
Pipes	0,014	-	-	-	-
Channel Lined	0,020	-	-	-	-
Channel unlined	0,050	-	-	-	-

input to WITSKM. Using the equations describing the I-D-F curves (Green and Stephenson, 1984) for the inland region, the recurrence intervals for the rainfall events were estimated and these together with other pertinent data is presented in Table 7.3. The actual rainfall and runoff data are presented in Appendix A.

Table 7.3 : Rainfall and Runoff Information for Recorded Storm Events

Rainfall					
Event	Peak Intensity (mm/h)	Duration (mins)	Volume (m^3)	Time to Peak (mins)	Rec Int
1	67	50	16962	25	2
2	142	60	36306	20	30
3	21,4	120	15962		1,4
Runoff					
Event	Peak Runoff (m^3/s)	Duration (mins)	Volume (m^3)	Time to Peak (mins)	
1	1,80	110	3723	15	
2	0,23	90	615	30	
3	0,00	-	0,0	-	

Examining the figures shown in Table 7.3, there is an apparent anomaly in that the 30 year recurrence interval event(event 2) produced a lower volume as well as runoff peak when compared to the 2 year event(event 1). The records in the data base were examined to determine the antecedent moisture conditions which could explain the apparent anomaly. For event 1, the total rainfall in the preceding 4 months was 70mm with no rainfall falling in May, June, or July. The situation was very different for event 2 with some 260 mm falling in the previous 3 months. The catchment before the event 2 can therefore be considered to be in a wet condition while the other event occurs after a dry winter season, thus the antecedent moisture conditions do not explain this anomaly. There are two further possible explanations which are:-

1 Measuring error due to faulty instruments for one or both of the events.

2 Gradings of the soil in the Waterval catchment showed that in the lower and middle regions of the catchment the clay content is of the order of 6-10%. This can cause a crust to form on the soil surface after a dry period. This would reduce the infiltration characteristics of the soil and could result in higher runoff volumes and peaks.

Regardless of which of the explanations is correct, the storm events will be used as recorded for comparison of the effectiveness of the different drainage options for the proposed town.

7.4 Drainage System Options

7.4.1 Introduction

There are a number of different possible layouts of town that could be considered for the Waterval catchment both from the town planning and stormwater drainage point of view. For the purposes being considered here, very little attention has been given to the town planning

aspects except to make provision for the stormwater drainage options being considered. In order to compare the runoff produced by WITSKM with the recorded hydrograph, the stormwater has all been channelled to the natural outlet of the catchment. Better systems may exist where the stormwater can be channelled out of the catchment at different points but these options will not be pursued here. The drainage systems chosen start with a completely linked system and then proceed through flood plain and dual drainage systems to the storage options.

7.4.2 Drainage Systems

System 1 : Fully linked drainage system

This option represents the worst in storm drainage design from the cost and runoff point of view. The pipe drainage network is designed to take all the flows from the roads and developed areas. The pipes are then linked directly to a lined channel running down the centre of the flood plain which discharges at the outlet of the catchment. The channel has also been designed so as not to surcharge onto the surrounding flood plains. The linking and roughness of the conduit network will cause the catchment to respond quickly to rainfall and the catchment will be susceptible to the shorter duration higher intensity storms. This type of drainage system will result in large pipe networks and expensive lined channels. As most of the runoff is confined to the pipe and channel network, the residents will not be frequently inconvenienced by flooded roads and park areas. However the question arises as to what size storm the drainage system should be designed for.

System 2 : Flood plain system

For this system the pipe and channel networks of system 1 were maintained, however the pipes were disconnected from the channel network and allowed to discharge onto the flood plains. In this way infiltration is encouraged and the runoff is spread over a larger and

rougher surface which results in smaller flow depths and hence greater retardation and temporary storage of runoff. To model the flood plain cum channel system, the park area (see fig 7.1) is subdivided into overland flow modules and channels. The runoff into a channel is from the upstream channel and from the overland flow modules immediately upstream. Referring to the township layout shown in Fig 7.1, the flow into channel 23 for example is from channel 22 and overland flow modules 127 and 128.

System 3 : Dual drainage system

A dual drainage system consists of a minor drainage system and a major drainage system. The minor system consists of the pipe and channel network, which is designed to cope with the lower recurrence interval events. The major drainage system, in this case the roads and flood plain, is designed to cater for the rarer higher recurrence interval events. In this way the size of the pipe and channel system can be reduced thereby lowering the costs of the drainage system. The question arises in designing these systems as to which event the minor system should be sized to cope with so as not to inconvenience the users to frequently. Event 1 is estimated to be a 2 year event and was used to size the minor drainage system. The pipe network was, as in the case of system 2, disconnected from the channel network. Unlined rougher channels were used and were sized so as not to surcharge for this event. For event 2 the pipes were made to surcharge onto the roads and the channels onto the adjacent overland flow modules.

System 4 : No pipe network system

A further system considered was to remove the pipe reticulation network entirely and use only the roads to remove the stormwater to the catchment outlet. The water was kept on the roads until road sections 148 and 149 from where the water was routed into channel 24. This system will be a low cost system but more inconvenient for the user.

System 5 : Temporary storage system

The drainage system of system 3 was altered so that the water surcharging from channel 24 could be stored temporarily on an open area such as a sports field. For this purpose overland flow modules 123 and 124 were given a flat slope of 0,001 with a roughness of 0,2 to act as temporary storage for the surcharged water. Being constructed at the bottom of the catchment, this option holds little advantage for the residents of the town. This option will however reduce the flood peaks that could be expected downstream which would be beneficial for the downstream communities.

System 6 : Detention storage system

As in system 5 the dual drainage option of system 3 was adapted to route the flows into a detention storage pond at the bottom of the catchment. The type of detention facility used had a 0,5m wide and 0,5m high culvert outlet and a spillway. The site at the bottom of the catchment could support a 7 m high dam wall giving a storage capacity of 62000 m^3. The storage depth relationship was estimated from a 1 in 2000 orthophoto of the catchment. The coefficients for the culvert outlet were calculated for the unsubmerged condition assuming inlet control and critical flow at the entrance. The equation governing such flow conditions is:-

$$Q = \frac{2}{3} C_v \, w \, \sqrt{\frac{2}{3} g} \, h^{3/2} \qquad (7.1)$$

where Q is the flow rate in m^3/s
w is the culvert width in m
h is the depth of water at culvert entrance in m
c_u is a discharge coefficient

For a coefficient c_u of 0,9 equation 7.1 becomes

$$Q = 0{,}77h^{3/2} \qquad (7.2)$$

For submerged flow through the culvert, the equation 7.3 describing flow through an orifice is used.

$$Q = C_s A \sqrt{2gh} \qquad (7.3)$$

where c_s coefficient of contraction
A is the area of the orifice opening in m^2
h is the depth of the water above the centre line of the opening in m

The coefficient of discharge c_s was taken as 0,6 which results in:-

$$Q = 0,66h^{\frac{1}{2}} \qquad (7.4)$$

for submerged culverts.

A spillway of width 20m was used at level 7m to handle the high flows. A spillway coefficient of 2,0 was used and the following equation results:

$$Q = 40h^{3,2} \qquad (7.5)$$

where h is the height above the spillway in m

7.5 Results of Simulations

A summary of the results of the simulations are presented in Table 7.4 and shown plotted in figs 7.2, 7.3 and 7.4.

Table 7.4 : Results of Simulations

Event 1			
System	Peak (m^3/s)	Time to peak (mins)	Volume (m^3)
1	3,9	40	5294
2 & 3	3,3	45	4634
4	3,5	40	5133
6	0,9	65	4634
obs	1,80	30	3723
Event 2			
System	Peak (m^3/s)	Time to peak (mins)	Volume (m^3)
1	11,9	30	22037
2	11,2	35	21594
3	10,4	40	21538
4	11,0	35	21892
5	7,8	45	21083
6	1,3	70	21538
obs	0,23	45	615
Event 3			
System	Peak (m^3/s)	Time to peak (mins)	Volume (m^3)
1	0,85	85	1258
2 & 3	0,72	95	469
4	0,78	95	998
obs	-	-	-

Event 1
26 September 1987

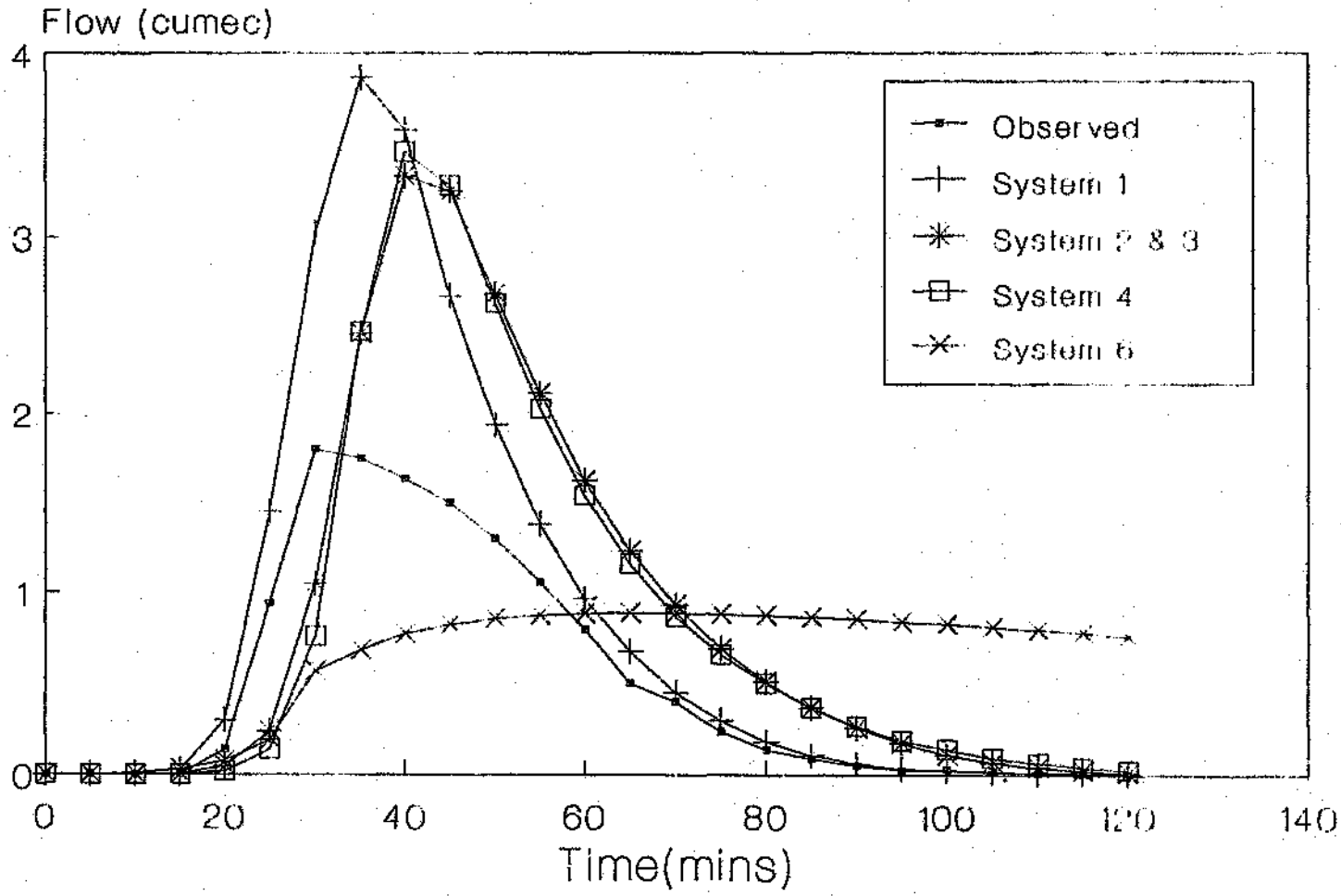


Figure 7.2 : Output Hydrographs for Drainage Systems

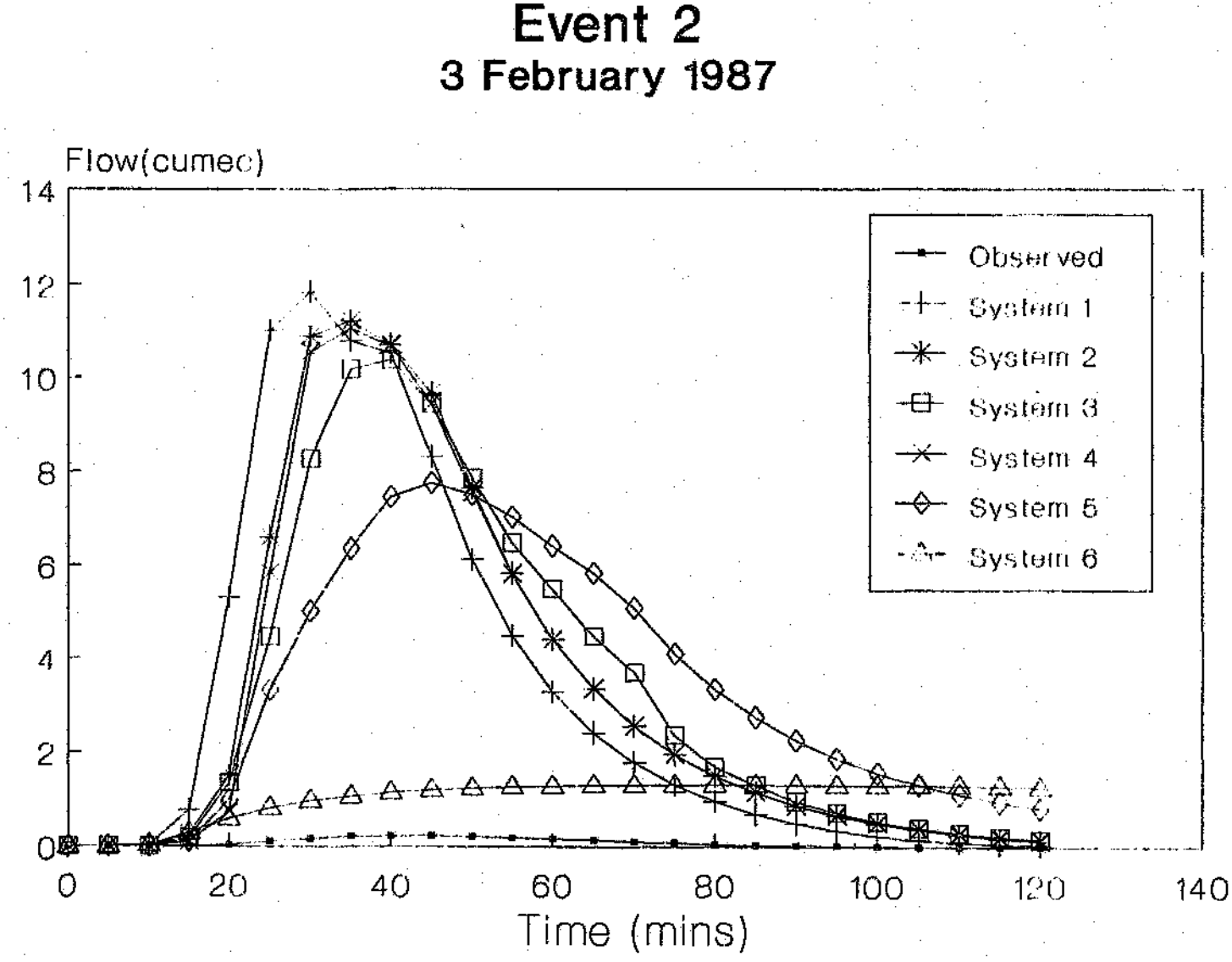


Figure 7.3 : Output Hydrographs for Drainage Systems

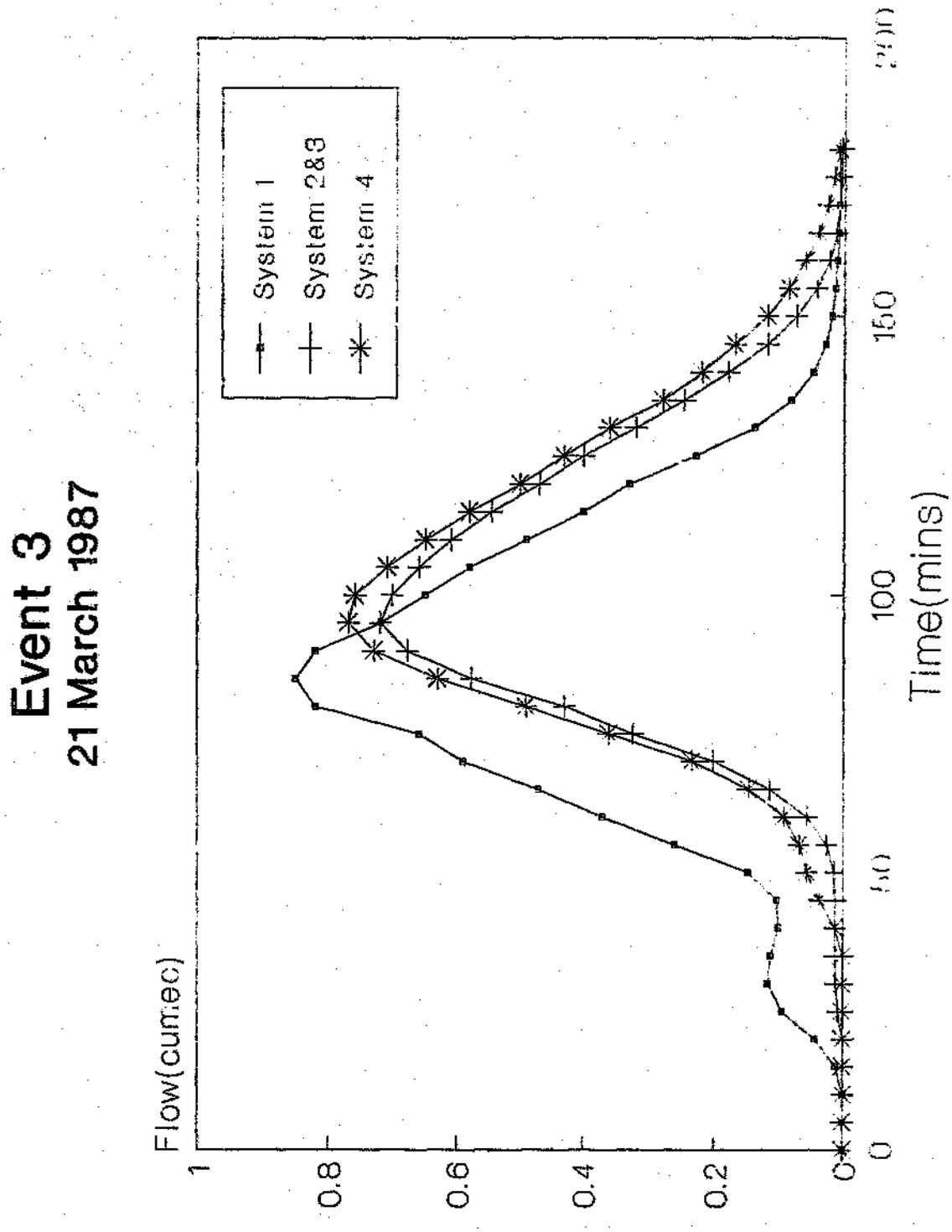


Figure 7.4 : Output Hydrographs for Drainage Systems

The largest peak and runoff volume was produced by drainage system 1, the all linked system. This is understandable due to the closed nature of the system. With the low roughness values of the pipes and channels the catchment will respond to short duration high intensity storms as is indicated by the comparatively low value of time to peak of 30 mins. The flood plain system did not reduce the flood peak and the volume substantially for the storm events. As the storm event of the 26-9-87 was used to size the pipe and channel system so as not to surcharge for this event, the flood plain and dual drainage option gave the same results. The dual drainage system caused a greater drop in peak than the flood plain option for event 2 when compared to system 1. This is due to the surcharge being routed onto the adjacent rougher flood plains. The greater width causing shallower depths of flow which together with the greater roughness results in lower runoff velocities. The volumes of runoff for drainage systems 3 and 2 are essentially the same. The no pipe system viz system 4 has a higher peak than system 3 and a shorter time to peak due to the ...ness of the road surface. The volume of runoff is also higher due to most of the runoff being confined to the impermeable roads. The remaining two systems use a means of temporary storage to hold back the flood waters for later release. System 5 using a flat and rough overland flow plane to retard the flow and temporarily store the water while the detention dam used a culvert outlet to choke the flow. For the storm event of the 3-02-87 the depth in the detention dam was 4,2m for the peak flow of 1,3 m^3/s. These two options were the most effective in reducing the flood peak.

Two aspects of the simulation results can be discussed. The one is the ability of the drainage system to limit the runoff peaks from the proposed town to a level that is acceptable to the downstream communities and drainage systems, be they the natural or installed stormwater drainage network. The other aspect is the design of the system for the convenience of the immediate community which it serves.

From the point of view of the downstream communities, the only effective method of limiting the flood was the detention storage pond system viz system 6. System 5 being the next best in reducing the flood peak. None of the other systems were able to reduce the runoff peaks significantly while none of the systems reduced the volume of runoff to the level of the recorded hydrograph for either of the events.

In considering the second aspect, both the costs of the system and the ability of the system to remove the runoff with an acceptable level of inconvenience to the residents of the town must be considered. The main factor of concern is the depths of flow that will occur on the roads and on the flood plain. The maximum depths are summarised in Table 7.5

Table 7.5 : Maximum Flow Depths(m) on Roads and Flood Plains

Event 2						
System	1	2	3	4	5	6
Roads	0,014	0,014	0,073	0,114	0,073	0,073
Flood Plain	0,026	0,095	0,222	0,147	0,246	0,026
Event 1						
System	1	2	3	4	5	6
Roads	0,009	0,009	0,009	0,059	0,009	0,009
Flood Plain	0,008	0,043	0,043	0,079	0,080	0,008
Event 3						
System	1	2 & 3	4			
Roads	0,002	0,002	0,022			
Flood Plain	0,002	0,024	0,036			

Drainage system 1 for both events causes the lowest depths on the roads and flood plain. However this scheme has the largest pipes with

the diameters ranging from 0,55m to 1,0m and channels (base width x height) from 1,0m x 1,0m at the top of the catchment to 2,0m x 1,0m at the bottom of the catchment based on the event of the 3-2-87. Thus this scheme although having the lowest depths will be the most costly. The pipe and channel network of system 3 were sized so as not to surcharge for the event of the 26-9-87. Thus the depths of flow for systems 2 and 3 for the event of the 26-9-87 are the same. The pipes for system 3 ranged in size from 0,35m to 0,65m and the channels (base widthxheight) from 1,0m x 0,3m to 2,0m x 0,6m. The drainage system is therefore cheaper for this option although the depth of flow on the roads due to surcharging pipes is greater at 0,073 m. The cheapest system will be the no pipe system of system 4 however this causes depths of 0,114m on road section 148 at the bottom of the catchment. The large depth on the flood plain of 0,222m for system 3 and 0,114m for system 4 occur on module 123 onto which pipe 10 discharges and channel 24 surcharges. The storage of water on a nearby field, as is done in system 5, is probably the most cost effective system in terms of the reduction of the runoff peak. The detention dam although effective in reducing the flood peak will prove to be the most expensive option and plays no role in improving the drainage system for the residents of the town.

CHAPTER 8 CONCLUSIONS AND RECOMMENDATIONS

The following conclusions and recommendations can be made as a result of this study:-

1. This version of WITSKM is an improvement of the original version with an improved routing algorithm which solves the instability problems experienced in the original version. The detention storage module has also been improved with the inclusion of the ability to model outlets. The program, during calibration, compared favourably with WITWAT, which has been used extensively in practice. However WITSKM is more powerful for planning and comparing stormwater management options for an urban catchment due to its greater flexibility in rerouting overflows from conduits and the inclusion of a storage module. WITSKM however still has shortcomings in that the design mode of WITWAT has not been included so the conduit network or minor system cannot be sized automatically for a particular storm event. This feature would be very useful in designing dual drainage systems. The ability to generate a particular recurrence interval rainfall event having a specified distribution has also not been included in this version of WITSKM. These capabilities should be added to produce a more complete planning and design tool for urban drainage systems.

2. The use of the aquifer module to model subsurface flow should be tested on more catchments where sufficient data on the soil profiles and types is available.

3. The future of WITSKM should be in its application to flood events and flood control for large catchments of the order of 100-1000 km^2. The program should then be improved to cope with the spatial distribution of rainfall over a catchment.

4. The Muskingum-Cunge routing algorithm has been found both during the test runs and the analysis of the drainage systems for the proposed Waterval town to be stable and superior to the original routing algorithm in WITSKM. However due to the short lengths of conduit and relatively steep slopes, kinematic routing is unecessary and a time shift method would have sufficed. This was found to be particularly true in the case of pipes which are normally smooth. However if WITSKM was to be applied to large rural catchments where the channels are long and relatively rough, a routing method such as Muskingum-Cunge which models the attenuation of a flood wave in a channel would be necessary.

5. With the WITWAT and WITSKM models the simulation of the rainfall-runoff process for urban catchments in terms of water quantity has been successfully modelled. More attention should now be given to the modelling of the quality of the runoff that can be expected from urban catchments.

REFERENCES

Abbot, M.B. , "Computational Hydraulics," Pitman Publ. Ltd. London, 1979.

Chow, V.T., "Open channel hydraulics," McGraw-Hill, New York, 1959.

Constantinides, C.A., "Two Dimensional kinematic modelling of the rainfall-runoff process," Water Systems Research Programme, University of the Witwatersrand, Report No. 1/1982, February 1982.

Courant, R., Friederichs, K.O., and Lewy, H., "Uber die partiellen differenzengleichungen der mathematischen physik," math. Ann. 100, 1928

Cunge, J.A., "On the subject of a flood propagation computation method (Muskingum method).," Journal of hydraulic research, vol 7, no 2, 1969.

Cunge, J.A., Holly, F.M., and Verwey, A., "Practical aspects of computational river hydraulics," Pitman, Bath, Britain, 1980

Green, I.R.A, "WITWAT stormwater drainage program-Version II," Water Systems Research Programme, University of the Witwatersrand, Report No. 2/1984, May 1984.

Green, I.R.A, and Stephenson, D.," Urban hydrology and drainage: WITWAT stormwater drainage program,"Water Systems Research Programme, University of the Witwatersrand, Report No 2/1984, May 1984.

Green, I.R.A, and Stephenson, D., "Urban hydrology and drainage : Comparisons of urban drainage models for use in South Africa," Water Systems Research Programme, University of the Witwatersrand, Report no. 3/86, May 1986.

Green, W.H., and Ampt, G.A., "Studies on soil physics, 1, The flow of air and water through soils," Journal of Agr. Soc., 4(1), 1911.

Henderson, F.M, "Open channel flow," Macmillan, New York, 1966.

Henderson, F.M., and Wooding R.A., "Overland flow and groundwater flow from a steady rainfall of finite duration," Jnl. of Geophysical Research, Vol. 69, No. 8, April 1964.

Holden, A.P, Personal communication , University of the Witwatersrand ,1989

Holden, A.P, and Stephenson, D., "Improved four-point solution of the kinematic equations," Journal of Hydraulics Research, Vol 26, No 4, 1988,pp 413-423.

Hope, A.S., and Mulder, C.J., "Hydrological investigations of small catchments in the Natal coastal belt and the role of physiography and land-use in the rainfall-runoff process," University of Zululand, 1979.

Huber, W.C., Heaney, J.P., Nix, S.J., Dickenson, R.E., and Polman, D.J., "Stormwater management model, User's Manual, Version iii ," Dept. of Environmental Engineering Sciences, University of Florida, Gainesville, Florida, U.S.A, May 1982.

James, W., and Robinson, M.A., "Time and space resolution for continuous dynamic storm and runoff model studies," Proceedings of the conference on hydraulics and hydrology in the small computer age, Florida,U.S.A, August 1985.

Kirkby, M.J., "Hillslope hydrology," John Wiley and sons, Britain, 1978.

Kolovopoulos, P., "Modelling the effect of urbanization on storm flow in the Braamfontein Spruit," Water Systems Research Programme, University of the Witwatersrand, Report No. 11/1986, June 1986.

Kolovopolous, P., " Free surface water level computation," Doctoral thesis, Dept. Civil Engineering, University of the Witwatersrand, 1988.

Li, R.M., and Rogers, J.J., "Short course - rivers 76 program," Colorado State University, U.S.A, August 1976.

Lighthill, F.R.S, and Whitham, C.B., "On kinematic Waves, 1. Flood movement in long rivers," Proc. Royal Society, London, Ser. A, Vol 229, May 1955.

Manley, R.E., "Calibration of hydrological model using optimization techniques," ASCE, Jnl. hydraulics division, February 1978.

Mulder, G.J., "Die invloed van infiltrasie op stormwaterafloop in die opvanggebied van die Ntuzerivier (Zoeloelandse kusstrook)," Water Research Commission report no. 66/4/84, September 1984.

Nash, J.E., and Sutcliffe, J.V., "River flow forecasting through conceptual models. Part 1-A discussion of principles," Jnl. of Hydrology, Vol 10, 1970.

Ponce, V.M, "Diffusion wave modelling of catchment dynamics," ASCE, Journal of hydraulics division, vol 112, no 8, August 1986.

Ponce, V.M, and Yevjevich, V., "Muskingum-Cunge method with variable parameters," ASCE, Journal of hydraulics division, vol 104, no 12, Dec 1978

Preissmann, A., "Propagation des intumescences dans les canaux et les riviers," 1'e Congres l'Association Francaise de Calcule, Grenoble, France, 1961.

Rawitz, E., Engman, E.T., and Cline, G.D., "Use of mass balance method for examining the role of soils in controlling watershed performance," Water resources research, 6(4), 1970.

Rawls, W.J., Brakensiek, D.L., and Saxton, K.E., "Soil water characteristics," American Society of Agric. Eng., St Josephs, Michigan, U.S.A, paper 81-2510, 1981.

Stephenson, D., "Application of a stormwater management program," Transactions of South African Institute of Civil Engineers, June 1989.

Stephenson, D., "Direct optimization of Muskingum routing coefficients," Jnl. of Hydrology, Vol 41, 1979.

Stephenson, D., and Meadows, M.E., "Kinematic hydrology and modelling," Elsevier, Amsterdam, 1986.

Sternberg, J.D., "Unsteady flow in drain pipes," M.Sc. thesis, Dept. of Civil Engineering, University of the Witwatersrand, February 1989.

Terstriep, M.L., and Stall, J.B., "The Illinois urban drainage area simulator, ILLUDAS," Illinois State Water Survey, Urbana, Bulletin 58, 1974.

Tischendorf, W.G., "Tracing stormflow to varying source area in small forested watershed in south eastern Piedmont," PhD dissertation, Univ Georgia, Athens, Georgia, 1969.

Wilson,E.M., "Engineering Hydrology," Macmillan, London, 1974

APPENDICES

APPENDIX A OBSERVED STORM EVENTS

A.1 Newark Street

A.1.1 Storm 15

Time interval between hyetograph ordinates (mm/h) = 1 minute

0	77	107	107	91	107	46
61	15					

Time interval between hydrograph ordinates (m^3/s) = 1 minute

0	0,006	0,031	0,054	0,068	0,065	0,054
0,042	0,034	0,023	0,017	0,011	0,008	0,006
0,003						

A.1.2 Storm 23

Time interval between hyetograph ordinates (mm/h) = 1 minute

31	31	15	15	31	31	46
61	61	91	91	61	61	61
46	46	61	15	46	77	46
61	77	77	77	137	61	91
107	46	31	15	0	15	

Time interval between hydrograph ordinates (m^3/s) = 1 minute

0	0,001	0,004	0,003	0,011	0,012	0,014
0,02	0,028	0,033	0,045	0,051	0,048	0,044
0,04	0,034	0,028	0,028	0,023	0,021	0,035
0,037	0,04	0,042	0,045	0,05	0,069	0,068
0,059	0,068	0,057	0,04	0,028	0,023	0,017
0,014	0,013	0,012	0,01	0,008		

A.2 Sunninghill Park

A.2.1 Storm 7 January 1987

Time interval between hyetograph ordinates (mm/h) = 5 minutes

0	7,2	112,8	103,2	60,0	24,0	7,2

Time interval between hydrograph ordinates (m^3/s) = 5 minutes

0,23	0,23	2,09	3,10	4,92	4,89	3,63
2,52	1,56	1,14	0,76	0,76	0,76	0,76

A.2.2 Storm 9 January 1987

Time interval between hyetograph ordinates (mm/h) = 5 minutes

0	7,2	62,4	98,4	100,8	60,0	36,0
9,6	7,2	2,4	9,6	4,8	0	0
4,8	2,4	4,8				

Time interval between hydrograph ordinates (m^3/s) = 5 minutes

0,34	0,14	0,98	2,25	2,76	5,72	4,45
4,45	3,28	2,24	1,74	1,33	1,33	0,95
0,95	0,61	0,61				

A.3 Zululand W1M17

A.3.1 Storm 7 February 1977

Time interval between hyetograph ordinates (mm/h) = 15 minutes

1,7	18,8	0,3	7,6	1,9	0,7	0,6
2,4	0,3	0,3	0,3	10,5	53,3	32,6
11,4	3,4	4,0	6,2	1,7	2,3	2,9
2,4	7,6	13,0	1,7	0,7	4,0	1,2
0,5	0,8	1,3	1,3	0,6	0,2	

Observed hydrograph

Time (mins)	Observed flow (m^3/s)
0	0
20	0,18
43	0,19
91	0,34
114	0,34
174	0,25
190	0,38
198	1,81
198	3,5
219	4,3
292	0,78
351	0,5
397	0,62
478	0,28
568	0,21
643	0,15

A.3.2 Storm 9 November 1977

Time interval between hyetograph ordinates (mm/h) = 15 minutes

0	18,8	101,9	54,7	14,3	0,4

Observed hydrograph

Time (mins)	Observed flow (m^3/s)
0	0
15	0,005
39	0,014
43	0,09
46	0,25
52	1,37
62	1,96
83	1,31
88	0,88
124	0,38
167	0,19
234	0,10
264	0,08

A.4 Waterval catchment

A.4.1 Storm 3 February 1987

Time interval between hyetograph ordinates (mm/h) = 5 minutes

1,2	26,3	97,4	141,8	136,4	63,9	43,0
56,0	11,0	3,0	0,8	0,2		

Time interval between hydrograph ordinates (m^3/s) = 5 minutes

0	0	0	0	0,04	0,11	0,17
0,22	0,22	0,23	0,21	0,19	0,17	0,14
0,11	0,09	0,05	0,04	0,03		

A.4.2 Storm 21 March 1987

Time interval between hyetograph ordinates (mm/h) = 5 minutes

1,6	2,7	7,2	17,4	17,3	10,9	8,0
7,2	10,8	17,3	21,4	16,5	15,8	17,6
13,5	19,5	11,2	9,7	6,5	7,0	6,9
4,2	3,3	4,0				

The rainfall event of 21 March 1987 did not cause runoff

A.4.3 Storm 26 September 1987

Time interval between hyetograph ordinates (mm/h) = 5 minutes

0,3	10,6	23,3	49,2	67,3	61,3	41,6
16,2	0	1,6				

Time interval between hydrograph ordinates (m^3/s) = 5 minutes

0	0	0	0	0,14	0,94	1,8
1,75	1,63	1,5	1,3	1,1	0,8	0,5
0,4	0,24	0,14	0,09	0,05	0,03	0,03
0,02	0,01	0,01				

APPENDIX B GOODNESS-OF-FIT CRITERIA

B.1 General

There are a variety of goodness-of-fit techniques that can be employed to examine how output from a hydrological simulation model conforms to the corresponding observed data. The techniques can range from subjective visual methods to purely objective techniques where the fit is measured by a value output by a mathematical function of the difference between modelled and observed values. The criteria used depends very much on the purpose for which the model was developed and the aims of the user. Peak flows maybe all that is of interest to the design engineer. However a comparison of observed and simulated runoff volumes or even hydrograph shapes could be important if storage is contemplated.

A number of goodness-of-fit criteria are discussed in Green and Stephenson (1986) and the recommended criteria used in their report have been used in this study.

B.2 Criteria Used

B.2.1 Graphical Plots

Graphs of observed and simulated hydrographs for each of the storm events for each of the catchments will be presented. These, although subjective, make for an immediate qualitative assessment of the goodness-of-fit of a run. Errors in shape, volumes, and peak flow rate are immediately obvious. A graphical plot also gives an idea of the capabilities of the program.

B.2.2 Peak Flowrate, Volume, and Mean Flowrate

WITSKM is essentially a single event model and therefore the peak runoff rates are one of the most important outputs. A direct

comparison of peaks will therefore be of value as will the time to peak. The difference is highlighted by calculating the ratio of the simulated to observed peak flow rates and the percentage error in the simulated peak. Similarly the percentage error and the ratio of simulated to observed are calculated for the volume and mean flow rates.

B.2.3 Dimensional, Ordinate Dependent Shape Factors

One of the most commonly used factors for assessing the goodness-of-fit as regards the shape of a hydrograph is the sum of squares criterion defined by:

$$G = \sum_{i=1}^{n} [\, q_o(t) - q_s(t) \,]_i^2$$

where
- G is the objective function to be minimized
- $q_o(t)$ is the observed flowrate at time t
- $q_s(t)$ is the simulated flowrate at time t
- n is the number of ordinates used in the comparison.

A further criterion which reduces the effect of large magnitude outliers and the effect of residuals whose values are less than unity , was suggested by Stephenson (1979). This criteric.. is defined by:

$$G = \sum_{i=1}^{n} / \, q_o - q_s \, /_i$$

Although the volumes of the observed and simulated hydrographs may agree closely, the shapes of the hydrographs maybe cosiderably different. A statistic which highlights this difference is the sum of the absolute areas of divergence between the two hydrographs, given by:

$$A = \sum_{i=1}^{n} / \frac{(\text{residual})_i + (\text{residual})_{i+1}}{2} \, dt \, /_i$$

where A is the sum of absolute area of divergence between the two hydrographs

dt is the time step

The above criteria all depend on the dimensions used as well as the number of ordinates describing the hydrographs. This makes comparison between storm events of different durations for a particular catchment difficult as well as comparisons between calibrations on different catchments. However for the purposes of this investigation the ability of WITSKM to calibrate for each event for each catchment is being investigated and compared to WITWAT so these statistics will suffice.

B.2.4 Dimensionless Measures of Fit

Nash and Sutcliffe (1970) proposed a dimensionless coefficient of model efficiency in the form

$$E = \frac{F_o^2 - F^2}{F_o^2}$$

where E is the efficiency of the model

$$F^2 = \sum_{i=1}^{n} [\, q_o(t) - q_s(t) \,]_i^2$$

$$F_o^2 = \sum_{i=1}^{n} [\, q_o(t) - q \,]_i^2$$

where q is the mean of the observed flows

A further statistic proposed by Manley (1978) is the proportional error of estimate (PEE) which is given by

$$PEE = \left[\frac{1}{n} \sum_{i=1}^{n} \left[\frac{q_o(t) - q_s(t)}{q_o(t)} \right]_i^2 \right]^{1/2}$$

APPENDIX C USER MANUAL

C.1 Introduction

WITSKM has been written in the BASIC programming language for IBM compatible PC 's using the DOS operating system. The system has been written for a computer system having a hard drive (c: drive), 640 Kb RAM, and an EGA card and colour screen. The program can be adapted for a dual floppy drive system and monochrome Hercules card if required. The program consists of an editor, computation, calibration, and connectivity graphics subprograms. The program operation is controlled from the editor subprogram by calling the other subprograms into memory using the chain command. After the computation, calibration, or connectivity graphics subprograms have run, the control is returned to the appropriate menu in the editor subprogram. The user controls the operation of the program using a series of five menus. These are the startup, primary, edit, output, and connectivity menus. Each of the menus displays a number of options on the screen. The arrow keys are used to move the cursor opposite the required option and the ENTER key is then used to select the option. Each of the menus will be discussed in section C.4 of this Appendix.

WITSKM simulates some of the more pertinent processes involved in rainfall-runoff. To use the model successfully, the implications of the assumptions made and theory used in modelling these processes is required. In addition the correct level of catchment discretization and values of the models parameters must be used. Guidelines on parameter estimation are given in this manual (section C.6). However the best way of gaining a feel for the parameters to use for different land-use and vegetation cover types is to calibrate the model against recorded data for different catchments. It was for this reason that the calibration subprogram was included in WITSKM. The catchment data included in this report and in Green and Stephenson (1986) can be used for this purpose.

C.2 Loading and Starting WITSKM

For a computer system having a hard drive, a directory should be created using DOS on the c: drive of the hard disk for storing the program, data and results files. Once the directory has been created copy the contents of the disk into the directory. For dual floppy drive systems insert the program disk into drive a:. Drive B: is used for the storage on a floppy disk of data and computation results. A floppy disk should always be installed in drive B:. To start the program, type WITSKM and enter. The message Chaining editor subprogram should appear on the screen.

C.3 Modules and Connectivity

The module types available in WITSKM are:

1. Overland flow modules
2. Circular pipes
3. Trapezoidal channels
4. Detention/retention storage dams
5. Compound channels
6. Aquifers

The numbers 1 - 6 above are used in the program to identify the module types. The appropriate number is entered during the data input procedure described in C.5. Each module is identified by a user supplied module number. Any number between 1 and 400 maybe used. The module numbers are used to determine the connectivity for the routing of flows from one module to the next. This is done by specifying the downstream module number and in the case of channels and pipes the module number of the module to which overflows must be routed is also entered. This feature is used for the modelling of dual drainage. A 0 is used for the downstream module number of the last module discharging from the catchment. The modules can be entered in any order and the program will determine the connectivity and the order in

which the modules must be calculated. If an error in the connectivity has been made due to entering the downstream or overflow module numbers incorrectly, a message will be written to the screen to this effect and control is returned to the primary menu.

For overland flow modules and aquifers, the module number of the underlying aquifer must also be specified. If information on the soils is available as regards depths, water table levels and soil properties then interflow and subsurface flow can be modelled if required. The various soil horizons can be modelled by a stack of aquifer modules underlying the overland flow module with the infiltration parameters being entered with the aquifer data. Each stack of aquifers is limited to three aquifers and the overland flow module. For the lowest aquifer the stack a 0 is entered for the infiltration module number. If subsurface flow is not to be modelled then a 0 is entered for the infiltration module number when entering the overland flow module data. The program then automatically creates a dummy aquifer having a module number from 900 upwards. The infiltration parameters for this aquifer are entered with the overland flow module data.

The maximium number of each of the types of modules that can be entered is given below:

Overland flow modules	- 70
Pipe modules	- 20
Trapezoidal channel modules	- 20
Storage modules	- 3
Compound channel modules	- 5
Aquifer modules	- 70

The total number of modules however must not exceed 140 and the maximum number of time steps allowed is 100. The program can handle 10 modules flowing onto a particular module i.e a maximum of 10 upstream modules are allowed for any particular module. Similarly in modelling dual drainage a maximium of 6 modules can overflow onto any module.

C.4 Menus

C.4.1 Startup Menu

The startup menu is the first menu displayed on the screen after starting the program by entering WITSKM at the DOS prompt. The menu is reproduced below (Figure 1) and the options available are discussed.

```
                    W I T S K M

              Kinematic Modular Flow Model

You may ....

   -►  enter new data or
       edit existing data
       duplicate a data file
       print a data file
       run the program with an existing data file
       quit

Move -► up or down to select option then press spacebar or ↵ to enter
```

Figure 1 : Startup menu

Enter new data : If a new data set is to be entered from the beginning then select this option from the menu. To enter the new data, the user answers a series of questions and fills in annotated tables. The details of the procedure for the entering of new data and the data required for the different modules is presented in section C.5. Once the data for the final module has been entered, a zero is

entered for the module number and control is returned to the primary menu. It is recommended that after the data is entered the file data option in the primary menu be used to save the data.

Edit existing data : If this option is selected the program will ask for the name of the data file. All data files used with WITSKM have the extension .DAT. When entering the filename this extension is NOT entered. For example the Sunninghill data file maybe called SUN71.DAT however SUN71 will be entered. Once the filename has been entered, the data is read from the file and control of the program transferred to the primary menu.

Duplicate a data file : This option is useful if an existing data file can be edited for use for another catchment or for analysing another management option. The program asks:

Enter source filename

The filename is entered remembering to leave off the .DAT extension. The data is then read from the file and the program asks :

Enter destination filename

Once the filename has been entered (excluding the .DAT extension) the data is written to the disk under the new data filename. The control of the program is then returned to the primary menu.

Print a data file : This option allows a data file to be printed out on a printer. The printout routine has been written for the IBM proprinter but has been found to work on the Star printer as well. The program prompts the user for the filename of the data to be printed out. Once printed out, control is transferred to the primary menu.

Run the program with an existing data file : This allows the user to run the program immediately if no editing of the data is required. Once this option has been selected and the name of the data file is entered, the computation subprogram is loaded. Once the computations have been completed the control of the program is returned to the output menu.

Quit : This option terminates the program and returns to DOS.

C.4.2 Primary Menu

W I T S K M

You may ....

-► edit data

file data

create a duplicate data file

output data to printer

run the program

output connectivity

quit

Move -► up or down to select option then press spacebar or ←┘ to enter

Figure 2 : Primary menu

Edit data : This option passes control to the editing menu which is described in section C.4.3.

File data : This option allows for the saving of data to the disk. In the case of a dual floppy drive system, the data is written to the B: drive. If the data has not been saved to disk before, the program will ask for a filename for the data. Once this has been entered, the

data is written to the disk. If the data has previously been saved, the data is written to the disk under the original filename. If a new filename for the edited data is required, the data must first be saved under the original filename and then the "create a duplicate data file" option in the primary menu can be used to duplicate the data under another filename. It is recommended that the new file is created before any data editing is undertaken.

Create a duplicate data file : This option is the similar to the duplicate file option in the startup menu. The difference being that only the destination filename is required as the source file is the data file presently being worked with. Once the destination filename has been entered, the data is read from the disk and subsequently written to the disk under the new filename. The control is then returned to the primary menu and any future editing will be carried out on the new data file.

Output data to printer : This option is the same as the "print a data file option" in the startup menu.

Run the program : This option will load the computation subprogram for the calculation of the runoff for the catchment. Before this option is excercised, the data should be saved and the connectivity should also be checked using the output connectivity option on this menu.

Output connectivity : Selection of this option transfers control to the connectivity menu shown below.

```
You may ....
   ->  Output connectivity table to printer
       Output connectivity to screen
       Quit
```

Figure 3 : Connectivity menu

Output connectivity table to printer : This option will output to the printer all the modules and give, for each module, the module numbers of the modules immediately upstream. A similar table is output giving, for each module, the module numbers of the modules which overflow onto any particular module.

Output connectivity to screen : Selection of this option will load the graphics connectivity subprogram. This program draws on the screen the module layout. The module numbers are shown for each module and the following symbols are used to represent the different module types.

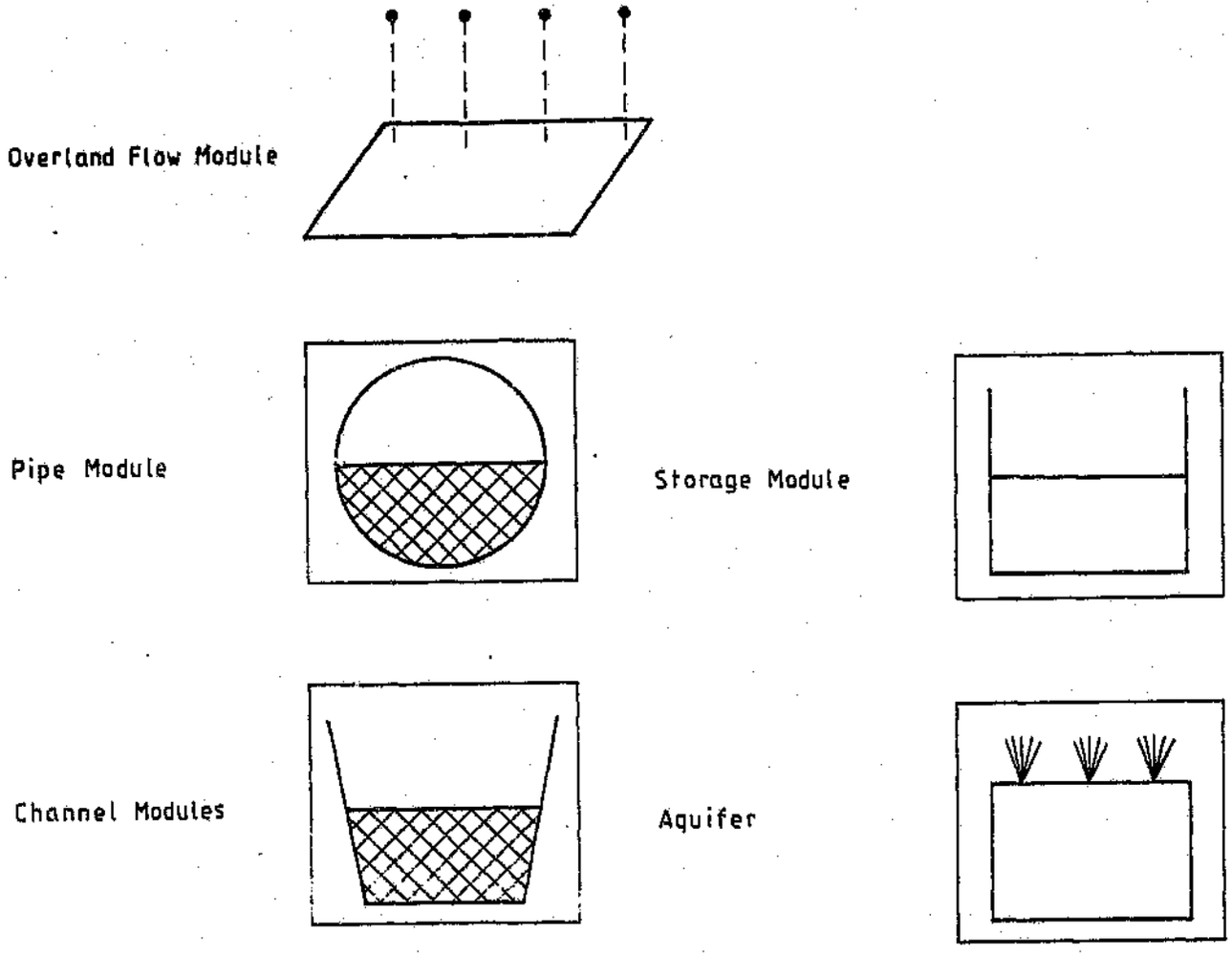


Figure 4 : Symbols used for connectivity output to screen

The graphics presentation does not allow for the checking of the overflow modules. The output to printer option can be used for this purpose.

Quit : This option is used to return to the primary menu.

C.4.3 Edit Menu

WITSKM DATA EDIT

You may ....

→ browse through the data

update a specific module

insert a module

delete a module

quit

Move → up or down to select option then press spacebar or ↵ to enter

Figure 5 : Edit menu

Browse through the data : This option allows the user to browse through the data. The spacebar is used to move from one set of data to the next. Any particular set of data can be edited using the UP and DOWN arrow keys. Once the data item has been selected the new value is typed in and the ENTER or ARROW keys can be used to enter the data

value. All the module data including the data set title and rainfall data can be edited using the browse option. The ESC key can be used at any time to exit browse mode and return to the edit menu.

The browse option is in fact the only way of editing the time step and rainfall data. When editing the computation time step data, the number of rainfall intensity values is automatically adjusted for each time step. The system used however requires that the new value for the timestep is a multiple of the original value. For example if the timestep was originally 5 minutes and the storm duration 1 hr, there will be 12 rainfall intensity values. If the time step is halved to 2,5 minutes, the rainfall intensity values for each of the 5 minute time intervals is doubled to give values for the 2,5 minute intervals thereby giving the required 24 intensity values.

Update a specific module : If only a specific module is required to be updated then this option is selected. The program prompts the user for the module number of the module to be updated. The data for that module is then displayed on the screen and the arrow keys can be used as in browsing the data, to select the data items to be updated. Once the edit has been completed the SPACE BAR is pressed to return to the edit menu.

Insert a module : This option is used if a module is to be added. The procedure followed is the same as that described in entering new data in section C.5.

Delete a module : Upon selecting this option the program prompts the user for the module number to be deleted. Once the module number has been entered the module is deleted.

Quit : This option returns control of the program to the primary menu.

C.4.4 Output Menu

WITSKM - Output mode

You may ....

-► Print output to printer

Plot output to screen

Call calibration program

Output results to a file

quit

Move -► up or down to select · ıs spacebar or ◄┘ to enter

Figure 6 : Output menu

Print output to printer : Selecting this option allows the hydrograph for any module to be output to the printer. Any number of modules maybe output merely by entering the appropriate module numbers. In addition at the head of the printout will be printed the name of the data file, the title of the data set, the rainfall and time step information followed by the module and hydrograph data. For each of the modules output, the module number, the volume in m^3 of the outflow hydrograph are printed out. For the overland flow modules the rainfall volume over that particular catchment is also printed out while for the conduits the capacity (cumec) of the conduit when flowing full is output. For all the modules the time, inflow hydrograph to, and the outflow hydrograph from the modules are printed out. For the pipe, channel, and aquifer modules in addition to the above hydrograph data, the overflows from the module are output if its capacity is exceeded. In the case of storage modules, the depth (m) of the water in the dam is also output. Once the required module data has been output a 0 is entered to return control to the output menu.

Plot output to screen : This option allows the inflow and outflow hydrographs to be plotted to the screen. If the DOS graphics is installed a screen dump of the plots maybe done. On the screen is displayed the inflow and outflow hydrographs together with the peak inflow and outflow rates and the volume under the outflow hydrograph. To display the hydrograph for a module, the program prompts the user for the module number. Once entered the hydrographs are displayed on the screen for that module. Any key is then pressed to return to the prompt for the next module to be displayed. Once completed a 0 is entered to return to the output menu.

Call calibration program : Choosing this option will load in the calibration subprogram for comparison of observed and simulated output. The program asks for the name of the data file containing the observed data. This data file must be in a particular format as shown below:-

```
n
time 1, observed flow 1
time 2, observed flow 2
  "          "2
time n, observed flow n
```

where n is the number of observed data points

The program then asks2 for the module number of the module with which to compare the observed data. Plots are then displayed on the screen of the observed and simulated hydrographs. The observed hydrograph being the broken line. By pressing any key the goodness-of-fit criteria as described in Appendix B are displayed on the screen. By pressing any key again the output menu is again displayed.

Output of results to a file : This option allows computation results to be output to a disk. The data is output in ASCII so the data file can be used as input to graphics packages such as Statgraphics to produce output for reports. The program asks for a name for the data file to save the data to. The .DAT extension is automatically added to the filename by the program so the filename entered must exclude the extension. This is followed by a prompt for the module number whose output hydrograph is to be saved to disk. The two questions are then repeated until all the hydrographs to be saved to disk have been output. A zero (0) is entered to return to the output menu.

C.5 Data Entry

A series of prompts and annotated tables are used to facilitate the inputting of new data. The annotated tables are those used to display the module data on the screen when using the browse option of the edit menu. For a prompt the appropriate data is typed and then entered using the ENTER key. Upon striking the enter key the next prompt or table is displayed on the screen. For the annotated tables, ARROW keys are used to move the cursor through the table and the default zero values can be changed as in editing using the browse or update of a specific module options of the edit menu. The first section of the data input procedure is to enter the data set title, time data, and the rainfall intensity data. This is then followed by the module data. To end the data input procedure a zero is entered at the prompt for the module number. Control is then returned to the primary menu where the data should be saved to file using the "file data" option. The following tables and prompts are used by the program to input the data:

1. Title Entry

W I T S K M

Enter a title describing the data set ....

Title line

Figure 7 : Prompt for entry of title

A single line of up to 72 characters can be entered as a title to describe the data or simulation run (Fig. 7). Once the title has been typed the ENTER key is used to input the title and move to the next table.

2. Entry of time data

The input of this data is by means of a table (Fig. 8):

W I T S K M

Time interval (hours) ..... : 0.000

Simulation duration (hours) ..... : 0.00

Rainfall duration (hours) ..... : 0.00

Enter new data else press space bar for next screen

Figure 8 : Entry table for time data

The UP and DOWN arrow keys are used to move through the table. Once the correct data has been entered the SPACEBAR is used to display the next screen (for information on the computation time step, and simulation duration see C.6)

3. Entry of Rainfall Intensities

The entry of this data is by means of a table (Fig. 9). Based on the time step and rainfall duration the program will display the correct number of rainfall intensities on the screen. The ARROW keys are used to move through the table and enter the required rainfall data in mm/hr. Once entered the SPACEBAR is pressed to get the next screen which is the entry of the module data.

WITSKM - Rain entry mode

Rainfall Intensities in mm/hr :

0.0	0.0	0.0	0.0	0.0	0.0

Enter new data else press space bar for next screen

Figure 9 : Entry table for rainfall intensities

4. Entry of Connectivity and Module Type Data

The program uses a series of prompts for the entry of the module number, downstream module number, and module type. The program also prompts for the infiltration module number in the case of the overland flow and aquifer modules or the overflow module number in the case of pipes and channels. The module number for a particular module must be unique and be an integer between 1 and 400. The type of module is identified by one of the following codes:

Code	Module type
1	Overland flow plane
2	Pipe
3	Trapezoidal channel
4	Storage basin
5	Compound channel
6	Aquifer

Once all the information describing the connectivity of the modules and module type has been entered, tables are used for the entry of the remaining module data.

5. Entry of Module Data

Overland flow data : Two tables (Figs 10 and 11) are used to enter the overland flow plane data depending on the infiltration module number entered. If a zero is entered then a dummy aquifer is created and the infiltration parameters for the dummy aquifer are entered with the module data as shown below.

WITSKM - Data Entry Mode

Module number	:	1
Downstream module	:	2
Module type	:	1
Width of catchment (m)	:	0
Length of catchment(m)	:	0
Manning n of catchment	:	0.0000
Slope of catchment (m/m)	:	0.0000
Permeability (mm/h)	:	0.0
Suction head (m)	:	0.000
Moisture content (Fraction by volume)	:	0.000
Porosity	:	0.000

Enter new data else press space bar for next screen

Figure 10 : Entry table for overland data without aquifer

The slopes, lengths, and widths of the catchments are estimated from topographical maps of the area being modelled. The infiltration and roughness parameters are estimated according to the guidelines given in section C.6. If the infiltration module number is greater than zero then the infiltration data is entered with the underlying aquifer and only the following data as shown below (Fig 11) need be entered

WITSKM - Data Entry Mode

Module number	:	2
Downstream module	:	3
Module type	:	1
Width of catchment (m)	:	0
Length of catchment(m)	:	0
Manning n of catchment	:	0.0000
Slope of catchment (m/m)	:	0.0000
Parallel module for infiltration	:	6

Enter new data else press space bar for next screen

Figure 11 : Entry table for overland data with aquifer

Pipe data : The data required for the pipe module are shown below (Fig 12).

WITSKM - Data Entry Mode

Module number	:	2
Downstream module	:	3
Module type	:	2
Pipe length (m)	:	0
Slope (m/m)	:	0.000
Roughness n	:	0.000
Pipe diameter (m)	:	0.00
Parallel module for overflows	:	0

Enter new data else press space bar for next screen

Figure 12 : Entry table for pipe data

Trapezoidal channel data :

WITSKM - Data Entry Mode

Module number	:	4
Downstream module	:	5
Module type	:	3
Length of trapezoidal channel (m)	:	0
Bed slope (m/m)	:	0.000
Roughness n	:	0.000
Base width of trapezoidal channel (m)	:	0.00
LH side slope (horiz/vert)	:	0.00
RH side slope (horiz/vert)	:	0.00
Maximum flow depth (m)	:	0.00
Parallel module for overflows	:	0

Enter new data else press space bar for next screen

Figure 13 : Entry table for trapezoidal channel

To describe the shape of this type of channel the base width, the left and right hand side slopes and the maximum flow depth in the channel are required. If a rectangular channel is to be modelled then the side slopes would be entered as zeroes. The program determines the flow that the channel can convey at the maximum flow depth. For any inflows to the channel that exceed this flow rate (capacity of the channel), the excess flow is rerouted to the overflow module as specified by the overflow module number.

Storage basin data : In modelling detention/retention storage basins provision has been made for storage facilities to have a bottom outlet if required. The table displayed on the screen initially is the table for a storage facility having a bottom outlet (Fig 14). The Yes for outlet in the table can be changed to No and the table is changed to that for a facility without a bottom outlet (Fig 15). The levels and water depths in the storage basin are all expressed relative to the same origin which is normally taken as the bottom of the storage basin at the

dam wall. A definition sketch and description of the equations describing the flow through the outlet and spillway are given in Chapter 3.

```
                    WITSKM - Data Entry Mode

Module number ..............................  :      5
Downstream module ..........................  :      6
Module type ................................  :      4
Coeff a in stor(m^3)=a*(depth(m))^b ........  :    0.0
Coeff b in stor(m^3)=a*(depth(m))^b.........  :  0.000
Spillway level (m) .........................  :   0.00
Outlet .....................................  :    Yes
Outlet invert level (m) ....................  :   0.00
Coeff c for unsubmerged outlet c*dep^d .....  :   0.00
Coeff d for unsubmerged outlet c*dep^d .....  :   0.00
Coeff e for submerged outlet e*dep^f .......  :   0.00
Coeff f for submerged outlet e*dep^f .......  :   0.00
Coeff g for spillway  g*dep^h ..............  :    0.0
Coeff h for spillway  g*dep^h ..............  :  0.000
Depth or diam (m) of outlet ................  :  0.000
Initial water level in dam (m) .............  :   0.00


          Enter new data else press space bar for next screen
```

Figure 14 : Entry table for storage basin with outlet

```
                    WITSKM - Data Entry Mode

Module number ..............................  :      5
Downstream module ..........................  :      6
Module type ................................  :      4
Coeff a in stor(m^3)=a*(depth(m))^b ........  :    0.0
Coeff b in stor(m^3)=a*(depth(m))^b.........  :  0.000
Spillway level (m) .........................  :   0.00
Outlet .....................................  :     No
Coeff g for spillway  g*dep^h ..............  :    0.0
Coeff h for spillway  g*dep^h ..............  :  0.000
Initial water level in dam (m) .............  :   0.00


          Enter new data else press space bar for next screen
```

Figure 15 : Entry table storage basin without outlet

Compound channels : Compound channels are modelled using the separate channel method. The type of channel that can be modelled is that having a main channel flanked by flood plains (Fig. 16). The model cannot cope with 2 main channels as shown in figure 17. To apply this method the channel is divided into channel segments based on considerations of roughness and channel geometry. The maximum number of channel segments that can be input is 5. To describe the shape of the channel, the coordinates of the channel points are entered. The first and last points entered must have the same elevation. The coordinate system used requires the origin to be on the left of the channel section as shown in figure 16. The X axis gives the horizontal position of the points while the y axis gives the elevation of the points. A maximum of 10 points can be entered to describe the channel cross section. In addition the program requires the point numbers at the boundaries of the channel segments. Three tables are used to enter the required input and are presented in figures 18, 19, and 20 for the channel given in figure 16.

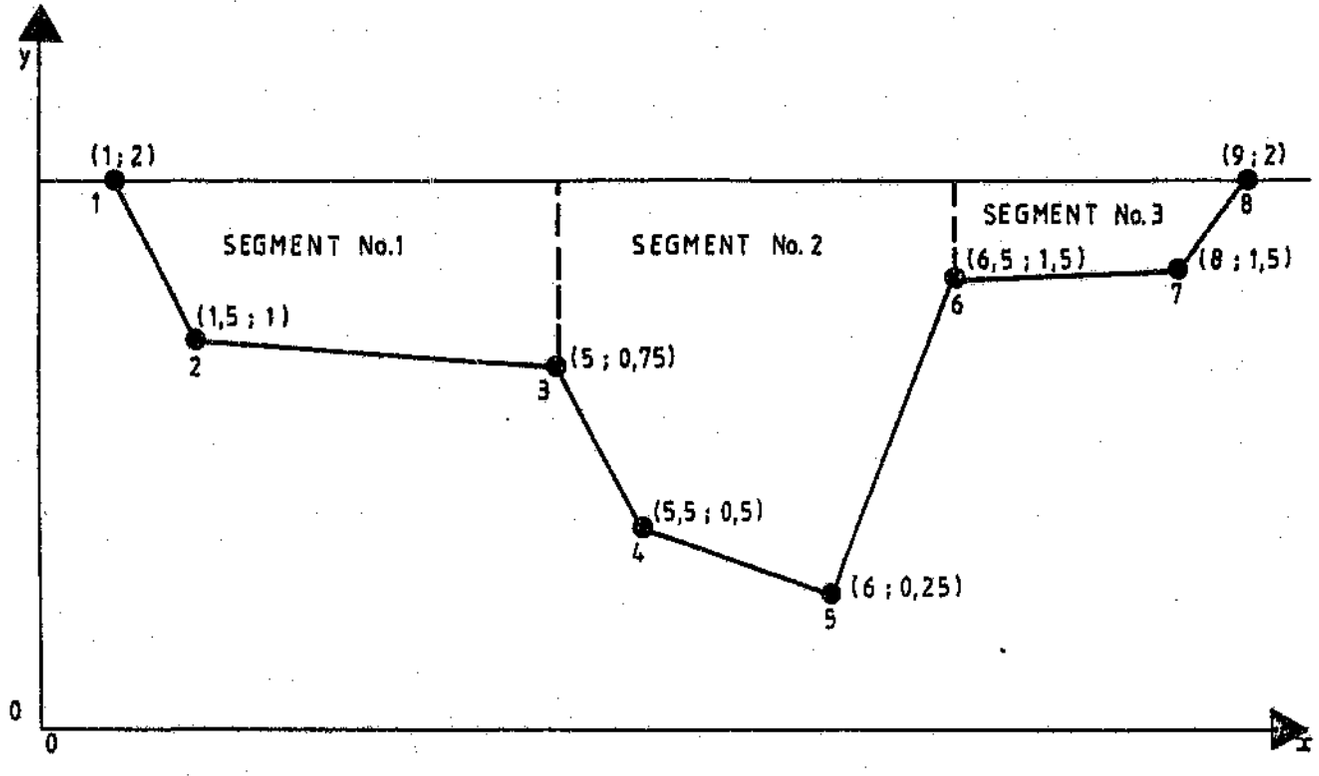


Figure 16 : Typical compound channel section

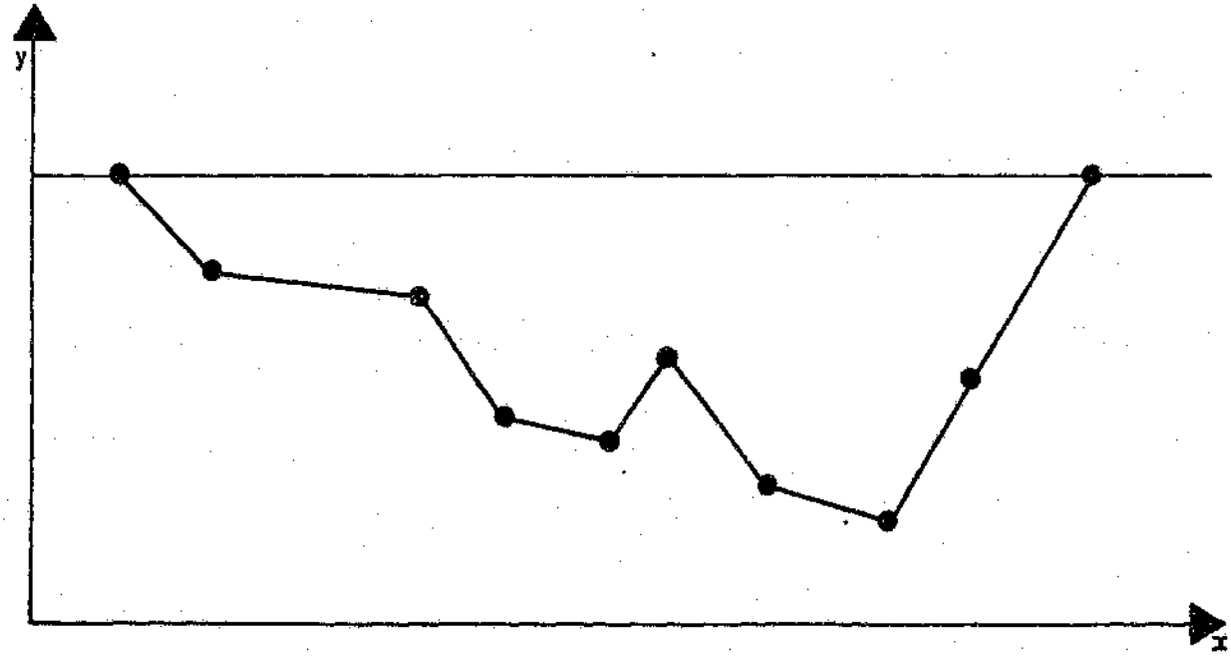


Figure 17 : Compound channel section having 2 main channels

WITSKM - Data Entry Mode

Module number	:	1
Downstream module	:	2
Module type	:	5
Length of compound channel (m)	:	500.00
Bed slope (m/m)	:	0.010
No. of points describing channel	:	8
No. of channel segments	:	3
Parallel module for overflows	:	0

Enter new data else press space bar for next screen

Figure 18 : Table for compound channel data entry

WITSKM - Data Entry Mode

Enter points describing channel section :

X coordinate (m)	Y coordinate (m)
1.00	2.00
1.50	1.00
5.00	0.75
5.50	0.50
6.00	0.25
6.50	1.50
8.00	1.50
9.00	2.00

Enter new data else press space bar for next screen

Figure 19 : Table for entry of channel section points

WITSKM - Data Entry Mode

Enter point numbers for channel segments :

Segment no.	Left point	Right point	Roughness (n)
1	1	3	0.055
2	3	6	0.040
3	6	8	0.055

Enter new data else press space bar for next screen

Figure 20 : Table for the entry of segment roughnesses and boundary points

Aquifers : The data required for aquifers is given below in figure 21.

WITSKM - Data Entry Mode

Module number	:	4
Downstream module	:	6
Module type	:	6
Width of aquifer (m)	:	0
Length of aquifer(m)	:	0
Depth of aquifer (m)	:	0.00
Slope (m/m)	:	0.000
Height of water table (m)	:	0.000
Moisture content(fraction by volume)	:	0.000
Porosity	:	0.000
Suction head (m)	:	0.000
Permeability (mm/h)	:	0.00
Parallel module for infiltration	:	0

Enter new data else press space bar for next screen

C.6. Guidelines for Parameter Estimation

C.6.1 General

The guidelines given in this section are based on the assumption that the user is familiar with the problem to be solved, the processes involved and the solution methodologies. The decision whether aquifer modules are to be used for the modelling of interflow and subsurface flow will depend on the objective for which the model is to be used and the amount of data on the depths and properties of the soil layers of the catchment. For most urbanized catchments the modelling of interflow and subsurface will not be necessary and the Horton type approach will suffice.

C.6.2 Level of Discretization

The level of discretization i.e the number of subcatchments or modules into which the catchment is divided depends largely on the information available to the user and the objectiye of the simulation. Factors to be considered in discretizing are sub-watershed boundaries, the location of the natural drainage channels, areas having similar topographical/land-use characteristics and the major pipe and channel networks. In discretizing the catchment into overland flow modules, it must also be remembered that the subcatchments are modelled as a sloping rectangular plane. This implies that the shape of the modules should be as close to rectangular as possible.

C.6.3 Guidelines for Choice of Time Data.

In using WITSKM the following guidelines are given in choosing the computation time step:

1. The time step should be chosen such that there are at least 5 time steps to the hydrograph peak. This results in consistent results as any further reduction in the time step does not usually cause a change in peak or a substantial improvement in the hydrograph shape.
2. The simulation duration should be about twice the rain storm duration.
3. In choosing the time step, the user must bear in mind the smaller the time step the longer the computation time will be.
4. The Muskingum-Cunge routing method has been found to be robust and produces results for most choices of computation time step. However the scheme has been found to produce the best results if the time and spatial increments are such that the Courant number Cr = 1. The Courant number is given by

$$Cr = cdt/dx \qquad C.1$$

where c is the wave celerity, and dt and dx are the computation time and spatial increments respectively.

C.6.4 Hydraulic Resistance to Overland Flow

The values for Manning's roughness coefficient are not as well known for overland flow as for channel flow. However summarized below in Table C.1 are typical Mannings "n" values as presented in Green and Stephenson (1984). When an overland flow module is a mixture of cover types an average value for Mannings n must be estimated based on the area of each cover type.

Table C.1 : Typical Manning Roughnesses for Overland Flow

Surface	Manning Roughness
Concrete or Smooth asphalt	0,012 - 0,013
Bare sand, rough asphalt or concrete paving	0,014 - 0,016
Bare loam soil, packed clay	0,033
Sparse vegetation	0,053 - 0,013
Short veld grass	0,20
Dense turf	0,35
Dense shrubbery and forest litter	0,40 - 0,48

C.6.4 Infiltration Parameters

The estimation of the infiltration parameters for the Green-Ampt infiltration equation remains more of an art than a science. Only by calibration against recorded data can parameters be chosen with confidence for different land-use types for ungauged catchments. Rawls et al (1983) analysed soil data that had been collected on some 5000 soil horizons in the U.S.A and was able to determine the Green-Ampt parameters based on the soil texture classification as proposed by the United States Department of Agriculture. In Table C.2, a summary of the parameters presented by Rawls et al is given for different classes of soil. The values presented in brackets for the porosity and the

suction head give the range of values obtained for these parameters. The hydraulic conductivity given in Table C.2 is half of the saturated hydraulic conductivity. This is the value of K recommended by Bouwers (1966) and Constantinides (1982) for use in the Green-Ampt formulation.

The figures provided in Table C.2 serve only as guidelines and are applicable to undeveloped areas. If a catchment has been urbanized, the hydraulic conductivity K estimated from Table C.2 according to soil type, must be adjusted to allow for the impervious areas. In deciding the amount to reduce the hydraulic conductivity by the degree that the impervious areas are connected together must be taken into

Table C.2 : Green-Ampt Parameters for different soil classifications

Soil Class	Porosity	Suction Head (m)	Hydraulic Conductivity (mm/h)
sand	0,42 (0,35-0,48)	0,05 (0,01-0,25)	118
loamy sand	0,40 (0,33-0,47)	0,06 (0,01-0,27)	30
sandy loam	0,41 (0,28-0,54)	0,11 (0,03-0,45)	11
loam	0,43 (0,33-0,53)	0,09 (0,01-0,59)	3,4
Silt loam	0,49 (0,39-0,58)	0,17 (0,03-0,95)	6,5
clay loam	0,31 (0,23-0,5)	0,21 (0,05-0,91)	1
sandy clay	0,32 (0,21-0,44)	0,24 (0,04-1,40)	0,6
silty clay	0,42 (0,33-0,51)	0,29 (0,06-1,39)	0,5
clay	0,39 (0,27-0,50)	0,32 (0,06-1,60)	0,3

consideration. Given in Chapter 6 of the report the adjusted parameters for sandy loam for the Sunninghill catchment land-use type are pesented.

APPENDIX D

PROGRAM LISTINGS

```
TYPE modconnectivity
     modno AS INTEGER
     ofl AS INTEGER
     typ AS INTEGER
     dsmod AS INTEGER
     infmod AS INTEGER
     p1 AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
    lng AS SINGL
    wid AS SINGLE
END TYPE

TYPE aquimod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wtl AS SINGLE
   sorp AS SINGLE
   perm AS SINGLE
   por AS SINGLE
   imc AS SINGLE
   cap AS SINGLE
   volume AS SINGLE
   yprev AS SINGLE
END TYPE

TYPE pipemod
    slo AS SINGLE
    diam AS SINGLE
    lng AS SINGLE
    man AS SINGLE
    cap AS SINGLE
    min AS SINGLE
END TYPE

TYPE trapmod
     slo AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     wid AS SINGLE
     ss1 AS SINGLE
     ss2 AS SINGLE
     mdep AS SINGLE
     cap AS SINGLE
END TYPE
TYPE stormod
    cl AS SINGLE
    sl AS SINGLE
    a AS SINGLE
    b AS SINGLE
    stlev AS SINGLE
    typ AS INTEGER
    ccu AS SINGLE
    cu AS SINGLE
    ccs AS SINGLE
    cs AS SINGLE
    csp AS SINGLE
    sp AS SINGLE
    depth AS SINGLE
    crit1 AS SINGLE
    crit2 AS SINGLE
    crit3 AS SINGLE
```

APPENDIX D

PROGRAM LISTINGS

EDITOR SUBPROGRAM

```
TYPE modconnectivity
     modno AS INTEGER
     ofl AS INTEGER
     typ AS INTEGER
     dsmod AS INTEGER
     infmod AS INTEGER
     pl AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
    lng AS SINGLE
    wid AS SINGLE
END TYPE

TYPE aquimod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wtl AS SINGLE
   sorp AS SINGLE
   perm AS SINGLE
   por AS SINGLE
   imc AS SINGLE
   cap AS SINGLE
   volume AS SINGLE
   yprev AS SINGLE
END TYPE

TYPE pipemod
    slo AS SINGLE
    diam AS SINGLE
    lng AS SINGLE
    man AS SINGLE
    cap AS SINGLE
    min AS SINGLE
END TYPE

TYPE trapmod
     slo AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     wid AS SINGLE
     ss1 AS SINGLE
     ss2 AS SINGLE
     mdep AS SINGLE
     cap AS SINGLE
END TYPE
TYPE stormod
    cl AS SINGLE
    sl AS SINGLE
    a AS SINGLE
    b AS SINGLE
    stlev AS SINGLE
    typ AS INTEGER
    ccu AS SINGLE
    cu AS SINGLE
    ccs AS SINGLE
    cs AS SINGLE
    csp AS SINGLE
    sp AS SINGLE
    depth AS SINGLE
    crit1 AS SINGLE
    crit2 AS SINGLE
    crit3 AS SINGLE
```

```
    prevstor AS SINGLE
END TYPE

TYPE compmod
    slo AS SINGLE
    lng AS SINGLE
    nosegs AS INTEGER
    nopts AS INTEGER
END TYPE

COMMON modu() AS modconnectivity, pipe() AS pipemod, storage() AS stormod
COMMON overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON qin(), qout(), unit(), con(), ovflo(), order(), nomod%, noit%, tint
COMMON expo, over(), rain(), pi
COMMON prompt1$, flag.file%, tcode%
COMMON title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON sim.time, rain.time, hyeto.number%, newmod%, flag.insert%
COMMON compchan() AS compmod, x(), y(), n(), segno()

REM $DYNAMIC

REDIM qin(140, 100), qout(140, 100), modu(140)  AS modconnectivity
REDIM overlnd(70) AS overmod, aqui(70) AS aquimod, pipe(30) AS pipemod
REDIM unit(140, 4), con(140, 12), ovflo(140, 6), order(2, 140)
REDIM over(140, 100), rain(100)
REDIM trapchan(25) AS trapmod, storage(3) AS stormod
REDIM compchan(5) AS compmod, x(5, 10), y(5, 10), n(5, 5), segno(5, 5, 2)

'Program to start up WITSKM

CLS

newmod% = 899
expo = 5 / 3
pi = 3.14159

p1% = 0: p2% = 0: p3% = 0: p4% = 0: p5% = 0: p6% = 0

prompt1$ = " Enter new data else press space bar for next screen "

flag.file% = 0


tcode% = 1

LOCATE 11, 28
COLOR 0, 2
PRINT "Chaining editor subprogram"

CHAIN "editorf"

END
```

```
DECLARE SUB conmenu (title2$, choice1%)
DECLARE SUB fileoutput ()
DECLARE SUB modconnect (print.connec%)
DECLARE SUB firstmenu (title1$, title3$, choice%)
DECLARE SUB secondmenu (title1$, title2$, choice%)
DECLARE SUB editmenu (title1$, choice1%)
DECLARE SUB progend ()
DECLARE SUB outputmenu (title1$, choice1%)
DECLARE SUB selectmenu (opt$(), NUMBEROPTIONS!, choice%)
DECLARE SUB duplicate ()
DECLARE SUB readfile ()
DECLARE SUB writefile ()
DECLARE SUB insertmodule ()
DECLARE SUB deletemodule ()
DECLARE SUB updatemodule ()
DECLARE SUB browse ()
DECLARE SUB raininput ()
DECLARE SUB TimestepInfo ()
DECLARE SUB TitleEnter ()
DECLARE SUB inputeditor ()
DECLARE SUB graphpl ()
DECLARE SUB printout ()
DECLARE SUB data.echo ()
TYPE modconnectivity
     modno AS INTEGER
     ofl AS INTEGER
     typ AS INTEGER
     dsmod AS INTEGER
     infmod AS INTEGER
     pl AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
    lng AS SINGLE
    wid AS SINGLE
END TYPE

TYPE aquimod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wtl AS SINGLE
   sorp AS SINGLE
   perm AS SINGLE
   por AS SINGLE
   imc AS SINGLE
   cap AS SINGLE
   volume AS SINGLE
   yprev AS SINGLE
END TYPE

TYPE pipemod
     slo AS SINGLE
    diam AS SINGLE
    lng AS SINGLE
    man AS SINGLE
    cap AS SINGLE
    min AS SINGLE
END TYPE

TYPE trapmod
     slo AS SINGLE
     lng AS SINGLE
     man AS SINGLE
```

```
     wid AS SINGLE
     ss1 AS SINGLE
     ss2 AS SINGLE
     mdep AS SINGLE
     cap AS SINGLE
END TYPE

TYPE stormod
    cl AS SINGLE
    sl AS SINGLE
    a AS SINGLE
    b AS SINGLE
    stlev AS SINGLE
    typ AS INTEGER
    ccu AS SINGLE
    cu AS SINGLE
    ccs AS SINGLE
    cs AS SINGLE
    csp AS SINGLE
    sp AS SINGLE
    depth AS SINGLE
    crit1 AS SINGLE
    crit2 AS SINGLE
    crit3 AS SINGLE
    prevstor AS SINGLE
END TYPE

TYPE compmod
    slo AS SINGLE
    lng AS SINGLE
    nosegs AS INTEGER
    nopts AS INTEGER
END TYPE


COMMON SHARED modu() AS modconnectivity, pipe() AS pipemod, storage() AS
stormod
COMMON SHARED overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON SHARED qin(), qout(), unit(), con(), ovflo(), order(), nomod%, noit%,
tint
COMMON SHARED expo, over(), rain(), pi
COMMON SHARED prompt1$, flag.file%, tcode%
COMMON SHARED title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON SHARED sim.time, rain.time, hyeto.number%, newmod%, flag.insert%
COMMON SHARED compchan() AS compmod, x(), y(), n(), segno()

screentitle1$ = " W I T S K M "
screentitle3$ = " Move " + CHR$(45) + CHR$(16) + " up or down to select option
then press spacebar or " + CHR$(17) + CHR$(196) + CHR$(217) + " to enter "


DO

  SELECT CASE tcode%

      CASE 1
         CALL firstmenu(screentitle1$, screentitle3$, choice%)

         SELECT CASE choice%

          CASE 1
            CALL TitleEnter
            CALL TimestepInfo
            hyeto.number% = CINT(rain.time / tint)
            CALL raininput

            CLS
```

```
      COLOR 0, 2
      LOCATE 1, 28
      PRINT "WITSKM - Data Entry Mode"

      CALL inputeditor
      tcode% = 3

   CASE 2
      CALL readfile
      tcode% = 3

   CASE 3
      CALL duplicate
      tcode% = 3

   CASE 4
      CALL readfile
      CALL modconnect(0)
      IF flag.esc% = 1 THEN
          flag.esc% = 0
      ELSE
          CALL data.echo
      END IF
      tcode% = 3

   CASE 5
      CALL readfile
      CLS
      LOCATE 11, 25
      COLOR 0, 2
      PRINT "Chaining computation subprogram"
      COLOR 3, 0
      CHAIN "witcom4f"

   CASE 6
      CALL progend

  END SELECT

CASE 2
      flag% = 0

      DO
         CALL outputmenu(screentitle3$, choice2%)

         SELECT CASE choice2%
             CASE 1
                 CALL printout
             CASE 2
                 CALL graphpl
             CASE 3
                 CLS
                 LOCATE 11, 25
                 COLOR 0, 2
                 PRINT "Chaining calibration subprogram"
                 COLOR 3, 0
                 CHAIN "calibf"
             CASE 4
                 CALL fileoutput

             CASE 5
                flag% = 1
         END SELECT

      LOOP UNTIL flag% = 1

      tcode% = 3
```

```
CASE 3
      edit.finish2% = 0

      DO

         CALL secondmenu(screentitle1$, screentitle3$, choice%)

         SELECT CASE choice%

           CASE 1

              edit.finish% = 0

              DO

                CALL editmenu(screentitle3$, choice1%)

                SELECT CASE choice1%
                  CASE 1
                     CALL browse
                     IF flag.esc% = 1 THEN
                        flag.esc% = 0
                     END IF
                  CASE 2
                     CALL insertmodule
                     IF flag.esc% = 1 THEN
                        flag.esc% = 0
                     END IF
                  CASE 3
                     CALL updatemodule
                     IF flag.esc% = 1 THEN
                        flag.esc% = 0
                     END IF
                  CASE 4
                     CALL deletemodule
                  CASE 5
                     edit.finish% = 1
               END SELECT

              LOOP UNTIL edit.finish% = 1

           CASE 2
              CALL writefile
           CASE 3
              CALL duplicate
           CASE 4
              CALL modconnect(0)
              CALL data.echo
           CASE 5
              CLS
              LOCATE 11, 25
              COLOR 0, 2
              PRINT "Chaining computation subprogram"
              COLOR 3, 0
              CHAIN "witcom4f"
           CASE 6

              edit.finish% = 0

              DO

                CALL conmenu(screentitle3$, choice1%)

                SELECT CASE choice1%
                  CASE 1
                     CALL modconnect(1)
                  CASE 2
```

```
                    CHAIN "planlasf"
                CASE 3
                    edit.finish% = 1
            END SELECT

        LOOP UNTIL edit.finish% = 1

    CASE 7
        CALL progend
        edit.finish2% = 1
END SELECT

LOOP UNTIL edit.finish2% = 1

END SELECT

LOOP UNTIL tcode% = 0

END

SUB conmenu (title2$, choice1%)

    REDIM opt$(3)

    COLOR 7, 0
    CLS
    COLOR 0, 2: LOCATE 1, 32
    PRINT "Connectivity Menu"

    COLOR 7, 0
    NUMBEROPTIONS = 3
    opt$(1) = "Output connectivity table to printer"
    opt$(2) = "Output connectivity to screen"
    opt$(3) = "Quit"

    COLOR 3, 0
    LOCATE , , 0, 6, 7
    LOCATE 7, 7: PRINT "You may ...."

    FOR 1% = 1 TO NUMBEROPTIONS
        LOCATE 7 + 2 * 1%, 15: PRINT opt$(1%)
    NEXT 1%

    LOCATE 23, 4: COLOR 0, 2
    PRINT title2$
    COLOR 6, 0
    LOCATE 9, 10: COLOR 7, 0
    PRINT CHR$(45) + CHR$(16)
    LOCATE 9, 15
    PRINT opt$(1)

    CALL selectmenu(opt$(), NUMBEROPTIONS, choice1%)

    ERASE opt$


END SUB

SUB editmenu (title1$, choice1%)

     REM $DYNAMIC

     REDIM opt$(5)

     COLOR 3, 0
     CLS
     COLOR 0, 2: LOCATE 1, 30
```

```
    PRINT " WITSKM DATA EDIT "

    COLOR 0, 2
    LOCATE 23, 4
    PRINT title1$

    NUMBEROPTIONS = 5
    opt$(1) = "browse through the data"
    opt$(2) = "update a specific module"
    opt$(3) = "insert a module"
    opt$(4) = "delete a module"
    opt$(5) = "quit"

    COLOR 3, 0
    LOCATE , , 0, 6, 7
    LOCATE 7, 7: PRINT "You may ...."

    FOR i% = 1 TO NUMBEROPTIONS
        LOCATE 7 + 2 * i%, 15: PRINT opt$(i%)
    NEXT i%

    LOCATE 23, 4: COLOR 0, 2
    PRINT title3$
    COLOR 6, 0
    LOCATE 9, 10: COLOR 7, 0
    PRINT CHR$(45) + CHR$(16)
    LOCATE 9, 15
    PRINT opt$(1)

    CALL selectmenu(opt$(), NUMBEROPTIONS, choice1%)

    ERASE opt$

END SUB

REM $STATIC
SUB firstmenu (title1$, title3$, choice%)

    REM $DYNAMIC

    REDIM opt$(6)

    COLOR 7, 0: CLS
    COLOR 0, 2: LOCATE 1, 32: PRINT title1$
    COLOR 3, 0: LOCATE 4, 25: PRINT " Kinematic Modular Flow Model "
    NUMBEROPTIONS = 6
    opt$(1) = "enter new data or"
    opt$(2) = "edit existing data"
    opt$(3) = "duplicate a data file"
    opt$(4) = "p[illegible]t a data file"
    opt$(5) = "[illegible]n the program with an existing data file"
    opt$(6) = "[illegible]t"

    COLOR 3, 0
    LOCATE , , 0, 6, 7
    LOCATE 7, 7: PRINT "You may ...."

    FOR i% = 1 TO NUMBEROPTIONS
        LOCATE 7 + 2 * i%, 15: PRINT opt$(i%)
    NEXT i%

    LOCATE 23, 4: COLOR 0, 2
    PRINT title3$
    COLOR 6, 0
    LOCATE 9, 10: COLOR 7, 0
    PRINT CHR$(45) + CHR$(16)
    LOCATE 9, 15
```

```
    PRINT opt$(1)

    CALL selectmenu(opt$(), NUMBEROPTIONS, choice%)

    ERASE opt$

END SUB

REM $STATIC
SUB outputmenu (title1$, choice1%)

    REDIM opt$(5)

    SCREEN 0: WIDTH 80
    COLOR 3, 0: CLS
    COLOR 0, 2
    LOCATE 1, 28: PRINT " WITSKM - Output mode "

    COLOR 0, 2
    LOCATE 23, 4
    PRINT title1$

    NUMBEROPTIONS = 5

    opt$(1) = "Print output to printer"
    opt$(2) = "Plot output to screen"
    opt$(3) = "Call calibration program"
    opt$(4) = "Output results to a file"
    opt$(5) = "quit"

    COLOR 3, 0
    LOCATE , , 0, 6, 7
    LOCATE 7, 7: PRINT "You may ...."

    FOR 1% = 1 TO NUMBEROPTIONS
        LOCATE 7 + 2 * 1%, 15: PRINT opt$(1%)
    NEXT 1%

    COLOR 6, 0
    LOCATE 9, 10: COLOR 7, 0
    PRINT CHR$(45) + CHR$(16)
    LOCATE 9, 15
    PRINT opt$(1)

    CALL selectmenu(opt$(), NUMBEROPTIONS, choice1%)

    ERASE opt$

END SUB

SUB progend

    COLOR 7, 0: CLS
    COLOR 0, 2
    LOCATE 10, 32
    PRINT " Program ended ": COLOR 7, 0: LOCATE 20, 1
    END

END SUB

SUB secondmenu (title1$, title2$, choice%)

    REDIM opt$(7)

    COLOR 7, 0
    CLS
    COLOR 0, 2: LOCATE 1, 32
```

```
    PRINT title1$

    COLOR 7, 0
    NUMBEROPTIONS = 7
    opt$(1) = "edit data"
    opt$(2) = "file data"
    opt$(3) = "create a duplicate data file"
    opt$(4) = "output data to printer"
    opt$(5) = "run the program"
    opt$(6) = "output connectivity"
    opt$(7) = "quit"

    COLOR 3, 0
    LOCATE , , 0, 6, 7
    LOCATE 7, 7: PRINT "You may ...."

    FOR 1% = 1 TO NUMBEROPTIONS
        LOCATE 7 + 2 * 1%, 15: PRINT opt$(1%)
    NEXT 1%

    LOCATE 23, 4: COLOR 0, 2
    PRINT title2$
    COLOR 6, 0
    LOCATE 9, 10: COLOR 7, 0
    PRINT CHR$(45) + CHR$(16)
    LOCATE 9, 15
    PRINT opt$(1)

    CALL selectmenu(opt$(), NUMBEROPTIONS, choice%)

    ERASE opt$

END SUB

SUB selectmenu (opt$(), NUMBEROPTIONS, choice%)

     choice% = 1

8130 a$ = INKEY$: IF a$ = "" THEN 8130
     IF a$ = CHR$(27) THEN
        flag.esc = 1
        choice% = 7
        GOTO 8490' escape
     END IF
     IF a$ = CHR$(13) OR a$ = CHR$(32) THEN 8490
     IF ASC(MID$(a$, 1, 1)) <> 0 THEN 8130
     '
     ' Cursor down
     '
     IF ASC(MID$(a$, 2, 1)) = 80 OR ASC(MID$(a$, 2, 1)) = 89 THEN 8200 ELSE
8330
8200 cursor = CSRLIN + 1
     choice% = (cursor - 7) / 2
     LOCATE cursor - 2, 10: PRINT SPACE$(2)
     LOCATE cursor - 2, 15: COLOR 3, 0: PRINT opt$(choice% - 1)
     IF choice% <= NUMBEROPTIONS THEN 8250 ELSE 8280
8250 LOCATE cursor, 10: COLOR 7, 0: PRINT CHR$(45) + CHR$(16)
     LOCATE cursor, 15: PRINT opt$(choice%)
     GOTO 8130
8280 ' Wrap around - bottom to top
     LOCATE 9, 10: COLOR 7, 0: PRINT CHR$(45) + CHR$(16): LOCATE 9, 15: PRINT
opt$(1)
     choice% = 1
     GOTO 8130
     '
8330 ' Cursor up
     '
```

```
     IF ASC(MID$(a$, 2, 1)) = 72 OR ASC(MID$(a$, 2, 1)) = 91 THEN 8360 ELSE
8130
8360 cursor = CSRLIN - 3
     choice% = (cursor - 7) / 2
     'IF I<1 THEN I=1:GOTO 1110
     LOCATE cursor + 2, 10: PRINT SPACE$(2)
     LOCATE cursor + 2, 15: COLOR 3, 0: PRINT opt$(choice% + 1)
     IF choice% >= 1 THEN 8420 ELSE 8450
8420 LOCATE cursor, 10: COLOR 7, 0: PRINT CHR$(45) + CHR$(16)
     LOCATE cursor, 15: PRINT opt$(choice%)
     GOTO 8130
8450 ' Wrap around - top to bottom
     LOCATE NUMBEROPTIONS * 2 + 7, 10: COLOR 7, 0: PRINT CHR$(45) + CHR$(16):
LOCATE NUMBEROPTIONS * 2 + 7, 15: PRINT opt$(NUMBEROPTIONS)
     choice% = NUMBEROPTIONS
     GOTO 8130
8490 LOCATE 23, 1: PRINT SPACE$(79)
'Selected option entered ... value for L returned to menu


END SUB

DECLARE FUNCTION getfilename$ (heading$)
DECLARE SUB clearscreen1 ()
DECLARE SUB build.file.index (flag.index%)
DECLARE SUB screeneditor (NBLOCK%, NCOLS%, nfield%, nvalues%, counter%,
row!(), col!(), a$, change%, M%, mm%, FLAGNEXT%, FLAGCURSOR%, FLAGCR%,
flag.string%)
DECLARE SUB readfile ()
DECLARE SUB datain ()
DECLARE SUB dataout ()
DECLARE SUB TitleEdit ()
DECLARE SUB TitleEnter ()
DECLARE SUB TimestepInfo ()
DECLARE SUB raininput ()
DECLARE SUB inputeditor ()
DECLARE SUB trapchanzero (p%)
DECLARE SUB aquizero (p%)
DECLARE SUB moduzero (p%)
DECLARE SUB overlndzero (p%)
DECLARE SUB pipezero (p%)
DECLARE SUB storagezero (p%)
DECLARE SUB compzero (p%)
DECLARE SUB ChangeModType (n%)
DECLARE SUB clearscreen ()
DECLARE SUB overlandinput (i%, newtype%)
DECLARE SUB pipeinput (i%, newtype%)
DECLARE SUB trapinput (i%, newtype%)
DECLARE SUB storinput (i%, newtype%)
DECLARE SUB compinput (i%, newtype%)
DECLARE SUB aquifinput (i%, newtype%)
TYPE modconnectivity
     modno AS INTEGER
     ofl AS INTEGER
     typ AS INTEGER
     dsmod AS INTEGER
     infmod AS INTEGER
     pl AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
    lng AS SINGLE
    wid AS SINGLE
END TYPE
```

```
TYPE aquimod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wtl AS SINGLE
   sorp AS SINGLE
   perm AS SINGLE
   por AS SINGLE
   imc AS SINGLE
   cap AS SINGLE
   volume AS SINGLE
   yprev AS SINGLE
END TYPE

TYPE pipemod
    slo AS SINGLE
    diam AS SINGLE
    lng AS SINGLE
    man AS SINGLE
    cap AS SINGLE
    min AS SINGLE
END TYPE

TYPE trapmod
     slo AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     wid AS SINGLE
     ss1 AS SINGLE
     ss2 AS SINGLE
     mdep AS SINGLE
     cap AS SINGLE
END TYPE

TYPE stormod
    cl AS SINGLE
    sl AS SINGLE
    a AS SINGLE
    b AS SINGLE
    stlev AS SINGLE
    typ AS INTEGER
    ccu AS SINGLE
    cu AS SINGLE
    ccs AS SINGLE
    cs AS SINGLE
    csp AS SINGLE
    sp AS SINGLE
    depth AS SINGLE
    crit1 AS SINGLE
    crit2 AS SINGLE
    crit3 AS SINGLE
    prevstor AS SINGLE
END TYPE

TYPE compmod
    slo AS SINGLE
    lng AS SINGLE
    nosegs AS INTEGER
    nopts AS INTEGER
END TYPE


COMMON SHARED modu() AS modconnectivity, pipe() AS pipemod, storage() AS
stormod
COMMON SHARED overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON SHARED qin(), qout(), unit(), con(), ovflo(), order(), nomod%, noit%,
```

```
tint
COMMON SHARED expo, over(), rain(), pi
COMMON SHARED prompt1$, flag.file%, tcode%
COMMON SHARED title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON SHARED sim.time, rain.time, hyeto.number%, newmod%, flag.insert%
COMMON SHARED compchan() AS compmod, x(), y(), n(), segno()


handler:

      SELECT CASE ERR

           CASE 53, 76

               BEEP: BEEP: PRINT : PRINT : PRINT "File "; file$; " does not
exist ... press any key"
               WHILE INKEY$ = "": WEND
               file$ = getfilename$(heading$)
               RESUME

           CASE 64, 52

               BEEP
               BEEP
               PRINT
               PRINT
               PRINT "Bad filename (limited to 8 characters) ... press any
key"
               WHILE INKEY$ = "": WEND
               file$ = getfilename$(heading$)
               RESUME

           CASE ELSE
               ON ERROR GOTO 0
      END SELECT

END

SUB browse

        CLS

        TitleEdit

        IF flag.esc% = 1 THEN
            flag.esc% = 0
            EXIT SUB
        END IF

        TimestepInfo

        IF flag.esc% = 1 THEN
           flag.esc% = 0
           EXIT SUB
        END IF

        raininput

        IF flag.esc% = 1 THEN
           flag.esc% = 0
           EXIT SUB
        END IF

        CLS

        COLOR 0, 2
        LOCATE 1, 28
```

```
PRINT "WITSKM - Browse Mode"


COLOR 3, 0

FOR i% = 1 TO nomod%

   CALL clearscreen
   flag.change% = 0

DO
   IF flag.change% = 1 THEN

       SELECT CASE modu(i%).typ
          CASE 1
              p1% = p1% + 1
              modu(i%).p1 = p1%

          CASE 2
              p2% = p2% + 1
              modu(i%).p1 = p2%

          CASE 3
              p3% = p3% + 1
              modu(i%).p1 = p3%

          CASE 4
              p4% = p4% + 1
              modu(i%).p1 = p4%

          CASE 5
              p5% = p5% + 1
              modu(i%).p1 = p5%

          CASE 6
              p6% = p6% + 1
              modu(i%).p1 = p6%

       END SELECT
    END IF

   flag.change% = 0

   IF modu(i%).modno < 900 THEN

      SELECT CASE modu(i%).typ

         CASE 1
            newtype% = 1
            overlandinput i%, newtype%

            IF newtype% <> 1 THEN
               n% = i%
               ChangeModType n%
               modu(i%).infmod = 0
               modu(i%).ofl = 0
               modu(i%).typ = newtype%
               newtype% = 1
               flag.change% = 1
            END IF

         CASE 2
            newtype% = 2

            pipeinput i%, newtype%
```

```
     IF newtype% <> 2 THEN
        n% = i%
        ChangeModType n%
        modu(i%).infmod = 0
        modu(i%).ofl = 0
        modu(i%).typ = newtype%
        newtype% = 2
        flag.change% = 1
     END IF

CASE 3
     newtype% = 3
     trapinput i%, newtype%

     IF newtype% <> 3 THEN
        n% = i%
        ChangeModType n%
        modu(i%).infmod = 0
        modu(i%).ofl = 0
        modu(i%).typ = newtype%
        newtype% = 3
        flag.change% = 1
     END IF

CASE 4
     newtype% = 4

     storinput i%, newtype%

     IF newtype% <> 4 THEN
        n% = i%
        ChangeModType n%
        modu(i%).infmod = 0
        modu(i%).ofl = 0
        modu(i%).typ = newtype%
        newtype% = 4
        flag.change% = 1
     END IF

CASE 5
     newtype% = 5

     compinput i%, newtype%

     IF newtype% <> 5 THEN
        n% = i%
        ChangeModType n%
        modu(i%).infmod = 0
        modu(i%).ofl = 0
        modu(i%).typ = newtype%
        newtype% = 5
        flag.change% = 1
     END IF

CASE 6
     newtype% = 6

     aquifinput i%, newtype%

     IF newtype% <> 6 THEN
        n% = i%
        ChangeModType n%
        modu(i%).infmod = 0
        modu(i%).ofl = 0
        modu(i%).typ = newtype%
        newtype% = 6
        flag.change% = 1
```

```
                END IF

            END SELECT

            IF flag.esc% = 1 THEN
               flag.esc% = 0
               EXIT SUB
            END IF

         END IF

      LOOP WHILE flag.change% = 1

      NEXT i%

END SUB

SUB ChangeModType (n%)

    clearscreen

    SELECT CASE modu(n%).typ

       CASE 1
          p% = modu(n%).p1
          overlndzero p%

          FOR k% = p% + 1 TO p1%

             SWAP overlnd(k% - 1), overlnd(k%)

             FOR t% = 1 TO nomod%
               IF modu(t%).typ = 1 THEN
                 IF modu(t%).p1 = k% THEN
                     modu(t%).p1 = k% - 1
                 END IF
               END IF
             NEXT t%

          NEXT k%

          p1% = p1% - 1

          IF modu(n%).infmod >= 900 THEN

                a% = modu(n% + 1).p1
                aquizero a%
                moduzero n% + 1

                FOR z% = a% + 1 TO p6%
                     SWAP aqui(z% - 1), aqui(z%)

                     FOR t% = 1 TO nomod%
                         IF modu(t%).typ = 6 THEN
                             IF modu(t%).p1 = z% THEN
                                 modu(t%).p1 = z% - 1
                             END IF
                         END IF
                     NEXT t%

                NEXT z%

                FOR z% = n% + 2 TO nomod%
                     modu(z% - 1).modno = modu(z%).modno
                     modu(z% - 1).dsmod = modu(z%).dsmod
                     modu(z% - 1).ofl = modu(z%).ofl
```

```
            modu(z% - 1).typ = modu(z%).typ
            modu(z% - 1).infmod = modu(z%).infmod
            modu(z% - 1).p1 = modu(z%).p1
        NEXT z%

        nomod% = nomod% - 1

        p6% = p6% - 1

    END IF

CASE 2

    p% = modu(n%).p1
    pipezero p%

    FOR k% = p% + 1 TO p2%
       SWAP pipe(k% - 1), pipe(k%)

       FOR t% = 1 TO nomod%
          IF modu(t%).typ = 2 THEN
            IF modu(t%).p1 = k% THEN
               modu(t%).p1 = k% - 1
            END IF
          END IF
       NEXT t%

    NEXT k%

    p2% = p2% - 1

CASE 3
    p% = modu(n%).p1
    trapchanzero p%

    FOR k% = p% + 1 TO p3%
       SWAP trapchan(k% - 1), trapchan(k%)

       FOR t% = 1 TO nomod%
           IF modu(t%).typ = 3 THEN
              IF modu(t%).p1 = k% THEN
                  modu(t%).p1 = k% - 1
              END IF
           END IF
       NEXT t%

    NEXT k%

    p3% = p3% - 1

CASE 4
    p% = modu(n%).p1
    storagezero p%

    FOR k% = p% + 1 TO p4%
       SWAP storage(k% - 1), storage(k%)

       FOR t% = 1 TO nomod%
         IF modu(t%).typ = 4 THEN
            IF modu(t%).p1 = k% THEN
               modu(t%).p1 = k% - 1
            END IF
         END IF
       NEXT t%

    NEXT k%
```

```
        p4% = p4% - 1

      CASE 5
        a% = modu(n%).p1
        compzero a%

        FOR z% = a% + 1 TO p5%
           SWAP compchan(z% - 1), compchan(z%)

           FOR t% = 1 TO nomod%
             IF modu(t%).typ = 5 THEN
                IF modu(t%).p1 = z% THEN
                   modu(t%).p1 = z% - 1
                END IF
             END IF
           NEXT t%

        NEXT z%

        p5% = p5% - 1


      CASE 6
        a% = modu(n%).p1
        aquizero a%

        FOR z% = a% + 1 TO p6%
           SWAP aqui(z% - 1), aqui(z%)

           FOR t% = 1 TO nomod%
             IF modu(t%).typ = 6 THEN
                IF modu(t%).p1 = z% THEN
                   modu(t%).p1 = z% - 1
                END IF
             END IF
           NEXT t%

        NEXT z%

        p6% = p6% - 1


    END SELECT

END SUB

SUB clearscreen1

      LOCATE 4, 28
      PRINT SPACE$(20)

      FOR jj% = 5 TO 15
         LOCATE jj%, 1
         PRINT SPACE$(79)
      NEXT jj%

END SUB

SUB datain

      ON ERROR GOTO handler

      OPEN file$ FOR INPUT AS #1

      LOCATE 6, 1: PRINT "Reading data from file "
      LOCATE 6, 24: PRINT file$
```

```
      INPUT #1, title$
      INPUT #1, tint, sim.time, rain.time, hyeto.number%, noit%, nomod%
      INPUT #1, p1%, p2%, p3%, p4%, p5%, p6%, newmod%

      FOR i% = 1 TO hyeto.number%
         INPUT #1, rain(i%)
      NEXT i%

      FOR i% = 1 TO nomod%
         INPUT #1, modu(i%).modno
         INPUT    #1,    modu(i%).dsmod,    modu(i%).of1,    modu(i%).typ,
modu(i%).infmod, modu(i%).p1

         SELECT CASE modu(i%).typ

           CASE 1
             p% = modu(i%).p1
             INPUT #1, overlnd(p%).man, overlnd(p%).slo, overlnd(p%).lng
             INPUT #1, overlnd(p%).wid

           CASE 2
             p% = modu(i%).p1
             INPUT #1, pipe(p%).lng, pipe(p%).diam, pipe(p%).slo, pipe(p%).man

           CASE 3
             p% = modu(i%).p1
             INPUT #1, trapchan(p%).slo, trapchan(p%).man, trapchan(p%).lng
             INPUT #1, trapchan(p%).wid, trapchan(p%).ss1, trapchan(p%).ss2,
trapchan(p%).mdep

           CASE 4
             p% = modu(i%).p1
             INPUT   #1,   storage(p%).typ,   storage(p%).a,   storage(p%).b,
storage(p%).c1
             INPUT  #1,  storage(p%).ccu,  storage(p%).cu,  storage(p%).ccs,
storage(p%).cs
             INPUT #1, storage(p%).depth
             INPUT  #1,  storage(p%).s1,  storage(p%).csp,  storage(p%).sp,
storage(p%).stlev

           CASE 5
             p% = modu(i%).p1
               INPUT    #1,    compchan(p%).slo,    compchan(p%).lng,
compchan(p%).nosegs, compchan(p%).nopts

             FOR j% = 1 TO compchan(p%).nopts
                  INPUT #1, x(p%, j%), y(p%, j%)
             NEXT j%

             FOR j% = 1 TO compchan(p%).nosegs
                  INPUT #1, segno(p%, j%, 1), segno(p%, j%, 2), n(p%, j%)
             NEXT j%

           CASE 6
             p% = modu(i%).p1
             INPUT    #1,    aqui(p%).slo,    aqui(p%).wid,    aqui(p%).lng,
aqui(p%).depth
             INPUT #1, aqui(p%).wt1
             INPUT    #1,    aqui(p%).sorp,    aqui(p%).perm,    aqui(p%).imc,
aqui(p%).por

         END SELECT

      NEXT i%

  CLOSE #1
```

```
END SUB

SUB dataout

      ON ERROR GOTO handler

      OPEN file$ FOR OUTPUT AS #1

      WRITE #1, title$
      WRITE #1, tint, sim.time, rain.time, hyeto.number%, noit%, nomod%
      WRITE #1, p1%, p2%, p3%, p4%, p5%, p6%, newmod%

      FOR i% = 1 TO hyeto.number%
           WRITE #1, rain(i%)
      NEXT i%

      FOR i% = 1 TO nomod%
           WRITE #1, modu(i%).modno
           WRITE    #1,    modu(i%).dsmod,    modu(i%).ofl,    modu(i%).typ,
modu(i%).infmod, modu(i%).p1

           SELECT CASE modu(i%).typ

              CASE 1
                p% = modu(i%).p1
                WRITE #1, overlnd(p%).man, overlnd(p%).slo, overlnd(p%).lng
                WRITE #1, overlnd(p%).wid

              CASE 2
                p% = modu(i%).p1
                WRITE   #1,   pipe(p%).lng,   pipe(p%).diam,   pipe(p%).slo,
pipe(p%).man

              CASE 3
                p% = modu(i%).p1
                WRITE #1, trapchan(p%).slo, trapchan(p%).man, trapchan(p%).lng
                 WRITE    #1,    trapchan(p%).wid,    trapchan(p%).ss1,
trapchan(p%).ss2, trapchan(p%).mdep

              CASE 4
                p% = modu(i%).p1
                WRITE  #1,  storage(p%).typ,  storage(p%).a,  storage(p%).b,
storage(p%).c1
                WRITE #1, storage(p%).ccu, storage(p%).cu, storage(p%).ccs,
storage(p%).cs
                WRITE #1, storage(p%).depth
                WRITE #1, storage(p%).sl, storage(p%).csp, storage(p%).sp,
storage(p%).stlev

              CASE 5
                p% = modu(i%).p1
                 WRITE    #1,    compchan(p%).slo,    compchan(p%).lng,
compchan(p%).nosegs, compchan(p%).nopts

                FOR j% = 1 TO compchan(p%).nopts
                    WRITE #1, x(p%, j%), y(p%, j%)
                NEXT j%

                FOR j% = 1 TO compchan(p%).nosegs
                    WRITE #1, segno(p%, j%, 1), segno(p%, j%, 2), n(p%, j%)
                NEXT j%

              CASE 6
                p% = modu(i%).p1
                WRITE   #1,   aqui(p%).slo,   aqui(p%).wid,   aqui(p%).lng,
aqui(p%).depth
                WRITE #1, aqui(p%).wtl
```

```
                WRITE   #1,   aqui(p%).sorp,   aqui(p%).perm,   aqui(p%).imc,
aqui(p%).por

          END SELECT

       NEXT i%

       CLOSE #1

END SUB

SUB deletemodule

   flag.change% = 0

   DO

      COLOR 3, 0
      CLS
      COLOR 0, 2
      LOCATE 1, 28: PRINT " WITSKM - Delete mode "
      COLOR 3, 0
      LOCATE 4, 1
      PRINT "Enter module number to be deleted (0 to return)"
      LOCATE 4, 49
      COLOR 7, 0: INPUT "", nn%
      COLOR 3, 0

      IF nn% = 0 THEN EXIT SUB

      FOR n% = 1 TO nomod%
          IF modu(n%).modno = nn% THEN
             ChangeModType n%
             moduzero n%

             FOR q% = n% + 1 TO nomod%
                 SWAP modu(q% - 1), modu(q%)
             NEXT q%

             nomod% = nomod% - 1

             EXIT SUB
          END IF

      NEXT n%

      BEEP: BEEP: PRINT : PRINT
      PRINT "Module "; nn%; " does not exist ... press any key"
      WHILE INKEY$ = "": WEND

   LOOP UNTIL flag.change% = 1

END SUB

SUB duplicate

      clearscreen


      IF flag.file% = 0 THEN

          heading$ = "Enter source filename"
          file$ = getfilename$(heading$)
          flag.file% = 1
          COLOR 3, 0
```

```
    END IF

    LOCATE 8, 1
    PRINT "Reading file "; file$
    datain

    heading$ = "Enter destination filename"
    file$ = getfilename$(heading$)
    LOCATE 8, 1
    PRINT "Writing to file "; file$
    dataout

END SUB

FUNCTION getfilename$ (heading$)

    clearscreen1
    LOCATE 4, 1
    PRINT heading$
    COLOR 7, 0
    LOCATE 4, 28
    INPUT "", file1$: COLOR 3, 0
    getfilename$ = UCASE$(file1$)
    getfilename$ = "b:" + file1$ + ".DAT"

END FUNCTION

SUB insertmodule

  flag.change% = 0

  DO

    COLOR 3, 0
    CLS
    COLOR 0, 2
    LOCATE 1, 28: PRINT " WITSKM - Update mode ": COLOR 3, 0
    LOCATE 4, 1
    PRINT "Enter module number to be updated (0 to return)"
    COLOR 7, 0
    LOCATE 4, 50
    INPUT "", nn%
    COLOR 3, 0

    IF nn% = 0 THEN EXIT SUB

    FOR i% = 1 TO nomod%
        IF modu(i%).modno = nn% THEN

            flag.change% = 0

            DO
              IF flag.change% = 1 THEN

                SELECT CASE modu(i%).typ
                   CASE 1
                       p1% = p1% + 1
                       modu(i%).p1 = p1%

                   CASE 2
                       p2% = p2% + 1
                       modu(i%).p1 = p2%

                   CASE 3
                       p3% = p3% + 1
                       modu(i%).p1 = p3%
```

```
        CASE 4
            p4% = p4% + 1
            modu(i%).p1 = p4%

        CASE 5
            p5% = p5% + 1
            modu(i%).p1 = p5%

        CASE 6
            p6% = p6% + 1
            modu(i%).p1 = p6%

    END SELECT
END IF

flag.change% = 0

change% = 0

SELECT CASE modu(i%).typ

   CASE 1
      newtype% = 1
      overlandinput i%, newtype%

      IF newtype% <> 1 THEN
         n% = i%
         ChangeModType n%
         modu(i%).infmod = 0
         modu(i%).ofl = 0
         modu(i%).typ = newtype%
         newtype% = 1
         flag.change% = 1
         change% = 1
      END IF

   CASE 2
      newtype% = 2

      pipeinput i%, newtype%

       IF newtype% <> 2 THEN
         n% = i%
         ChangeModType n%
         modu(i%).infmod = 0
         modu(i%).ofl = 0
         modu(i%).typ = newtype%
         newtype% = 2
         flag.change% = 1
         change% = 1
      END IF

   CASE 3
      newtype% = 3
      trapinput i%, newtype%

      IF newtype% <> 3 THEN
         n% = i%
         ChangeModType n%
         modu(i%).infmod = 0
         modu(i%).ofl = 0
         modu(i%).typ = newtype%
         newtype% = 3
         flag.change% = 1
         change% = 1
      END IF
```

```
                CASE 4
                   newtype% = 4

                   storinput i%, newtype%

                   IF newtype% <> 4 THEN
                      n% = i%
                      ChangeModType n%
                      modu(i%).infmod = 0
                      modu(i%).ofl = 0
                      modu(i%).typ = newtype%
                      newtype% = 4
                      flag.change% = 1
                      change% = 1
                   END IF

                CASE 5
                   newtype% = 5

                   compinput i%, newtype%

                   IF newtype% <> 5 THEN
                      n% = i%
                      ChangeModType n%
                      modu(i%).infmod = 0
                      modu(i%).ofl = 0
                      modu(i%).typ = newtype%
                      newtype% = 5
                      flag.change% = 1
                      change% = 1
                   END IF

                CASE 6
                   newtype% = 6

                   aquifinput i%, newtype%

                   IF newtype% <> 6 THEN
                      n% = i%
                      ChangeModType n%
                      modu(i%).infmod = 0
                      modu(i%).ofl = 0
                      modu(i%).typ = newtype%
                      newtype% = 6
                      flag.change% = 1
                      change% = 1
                   END IF


             END SELECT

          LOOP UNTIL change% = 0

          EXIT SUB

       END IF

    NEXT i%

    BEEP: BEEP: PRINT : PRINT
    PRINT "Module "; nn%; " does not exist ... press any key"
    WHILE INKEY$ = "": WEND

 LOOP UNTIL flag.change% = 1

END SUB
```

```
SUB readfile

      clearscreen
      heading$ = "Enter data filename"
      file$ = getfilename$(heading$)

      datain
      flag.file% = 1
      LOCATE 4, 1
      PRINT SPACE$(79)
      LOCATE 6, 1
      PRINT SPACE$(79)

END SUB

SUB TitleEdit

      REDIM row(1), col(1)


      clearscreen

      COLOR 0, 2
      LOCATE 23, 13
      PRINT prompt1$


      COLOR 3, 0
      LOCATE 4, 1
      PRINT "Title"
      COLOR 3, 4
      LOCATE 6, 1
      PRINT title$
      COLOR 3, 0

      nvalues% = 1
      NBLOCK% = 1
      NCOLS% = 1
      nfield% = 79
      counter% = 1
      M% = 1

      row(M%) = 6
      col(M%) = 1

      flag.string% = 1

      CALL screeneditor(NBLOCK%, NCOLS%, nfield%, nvalues%, counter%, row(),
col(), a$, change%, M%, mm%, FLAGNEXT%, FLAGCURSOR%, FLAGCR%, flag.string%)

      flag.string% = 0

      IF flag.esc% = 1 THEN
         EXIT SUB
      END IF

      FLAGNEXT% = 0
      FLAGCR% = 0
      FLAGCURSOR% = 0
      IF change% <> 0 THEN title$ = a$

END SUB

SUB updatemodule

       COLOR 3, 0
```

```
        CLS

        COLOR 0, 2
        LOCATE 1, 28
        PRINT "WITSKM- Insert mode :"
        COLOR 3, 0
        flag.insert% = 1
        inputeditor
        flag.insert% = 0

END SUB

SUB writefile

     clearscreen

     IF flag.file% = 0 THEN
         COLOR 3, 0
         heading$ = "Enter data filename"
         file$ = getfilename$(heading$)
         flag.file% = 1
      END IF

      LOCATE 6, 1: PRINT "Writing data to file "
      LOCATE 6, 24: PRINT file$
      dataout
      LOCATE 4, 1: PRINT SPACE$(79)
      LOCATE 6, 1: PRINT SPACE$(79)


END SUB

DECLARE SUB compzero (p5%)
DECLARE SUB aquifinput (i%, newtype%)
DECLARE SUB compinput (i%, newtype%)
DECLARE SUB aquizero (p6%)
DECLARE SUB moduzero (i%)
DECLARE SUB overlndzero (p1%)
DECLARE SUB pipezero (p2%)
DECLARE SUB trapchanzero (p3%)
DECLARE SUB storagezero (p4%)
DECLARE SUB trapinput (i%, newtype%)
DECLARE SUB storinput (i%, newtype%)
DECLARE SUB modulechange (ans$, i%)
DECLARE SUB pipeinput (i%, newtype%)
DECLARE SUB overlandinput (i%, newtype%)
DECLARE SUB modulenumberinfo (i%)
DECLARE SUB screeneditor (NBLOCK%, NCOLS%, nfield%, nvalues%, counter%, row(),
col(), a$, CHANGE%, M%, mm%, FLAGNEXT%, FLAGCURSOR%, FLAGCR%, flag.string%)
DECLARE SUB leftarrow (M%, row!(), col!(), counter%)
DECLARE SUB uparrow (counter%, M%, row!(), col!(), NBLOCK%, NCOLS%)
DECLARE SUB rightarrow (M%, nvalues%, counter%, row!(), col!())
DECLARE SUB downarrow (counter%, NBLOCK%, M%, row!(), col!(), nvalues%,
NCOLS%)
DECLARE SUB clearscreen ()
TYPE modconnectivity
     modno AS INTEGER
     ofl AS INTEGER
     typ AS INTEGER
     dsmod AS INTEGER
     infmod AS INTEGER
     pl AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
```

```
    lng AS SINGLE
    wid AS SINGLE
END TYPE

TYPE aquimod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wtl AS SINGLE
   sorp AS SINGLE
   perm AS SINGLE
   por AS SINGLE
   imc AS SINGLE
   cap AS SINGLE
   volume AS SINGLE
   yprev AS SINGLE
END TYPE

TYPE pipemod
    slo AS SINGLE
    diam AS SINGLE
    lng AS SINGLE
    man AS SINGLE
    cap AS SINGLE
    min AS SINGLE
END TYPE

TYPE trapmod
    slo AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     wid AS SINGLE
     ss1 AS SINGLE
     ss2 AS SINGLE
     mdep AS SINGLE
     cap AS SINGLE
END TYPE

TYPE stormod
    cl AS SINGLE
    sl AS SINGLE
    a AS SINGLE
    b AS SINGLE
    stlev AS SINGLE
    typ AS INTEGER
    ccu AS SINGLE
    cu AS SINGLE
    ccs AS SINGLE
    cs AS SINGLE
    csp AS SINGLE
    sp AS SINGLE
    depth AS SINGLE
    crit1 AS SINGLE
    crit2 AS SINGLE
    crit3 AS SINGLE
    prevstor AS SINGLE
END TYPE

TYPE compmod
    slo AS SINGLE
    lng AS SINGLE
    nosegs AS INTEGER
    nopts AS INTEGER
END TYPE
```

```
COMMON SHARED modu() AS modconnectivity, pipe() AS pipemod, storage() AS
stormod
COMMON SHARED overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON SHARED qin(), qout(), unit(), con(), ovflo(), order(), nomod%, noit%,
tint
COMMON SHARED expo, over(), rain(), pi
COMMON SHARED prompt1$, flag.file%, tcode%
COMMON SHARED title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON SHARED sim.time, rain.time, hyeto.number%, newmod%, flag.insert%
COMMON SHARED compchan() AS compmod, x(), y(), n(), segno()

END

SUB aquifinput (i%, newtype%)

    REDIM row(13), col(13)

    CALL clearscreen

    COLOR 0, 2
    LOCATE 23, 13
    PRINT prompt1$

    COLOR 3, 0
    LOCATE 3, 1: PRINT "Module number ..... :"
    COLOR 3, 4
    LOCATE 3, 45: PRINT USING "######"; modu(i%).modno
    COLOR 3, 0
    LOCATE 4, 1: PRINT "Downstream module ..... :"
    COLOR 3, 4
    LOCATE 4, 45: PRINT USING "######"; modu(i%).dsmod
    COLOR 3, 0
    LOCATE 5, 1: PRINT "Module type ..... :"
    COLOR 3, 4
    LOCATE 5, 45: PRINT USING "######"; modu(i%).typ

    pnt% = modu(i%).p1
    COLOR 3, 0
    LOCATE 6, 1: PRINT "Width of aquifer (m) ..... :"
    COLOR 3, 4
    LOCATE 6, 45: PRINT USING "###.###"; aqui(pnt%).wid
    COLOR 3, 0
    LOCATE 7, 1: PRINT "Length of aquifer(m) ..... :"
    COLOR 3, 4
    LOCATE 7, 45: PRINT USING "######"; aqui(pnt%).lng
    COLOR 3, 0
    LOCATE 8, 1: PRINT "Depth of aquifer (m) ..... :"
    COLOR 3, 4
    LOCATE 8, 45: PRINT USING "###.##"; aqui(pnt%).depth
    COLOR 3, 0
    LOCATE 9, 1: PRINT "Slope (m/m) ..... :"
    COLOR 3, 4
    LOCATE 9, 45: PRINT USING "##.###"; aqui(pnt%).slo
    COLOR 3, 0
    LOCATE 10, 1: PRINT "Height of water table (m) ..... :"
    COLOR 3, 4
    LOCATE 10, 45: PRINT USING "##.###"; aqui(pnt%).wtl
    COLOR 3, 0
    LOCATE 11, 1: PRINT "Moisture content(fraction by volume) .... :"
    COLOR 3, 4
    LOCATE 11, 45: PRINT USING "##.###"; aqui(pnt%).imc
    COLOR 3, 0
    LOCATE 12, 1: PRINT "Porosity ..... :"
    COLOR 3, 4
    LOCATE 12, 45: PRINT USING "##.###"; aqui(pnt%).por
    COLOR 3, 0
    LOCATE 13, 1: PRINT "Suction head (m) ..... :"
```

```
    COLOR 3, 4
    LOCATE 13, 45: PRINT USING "##.###"; aqui(pnt%).sorp
    COLOR 3, 0
    LOCATE 14, 1: PRINT "Permeability (mm/h) ..................... :"
    COLOR 3, 4
    LOCATE 14, 45: PRINT USING "###.##"; aqui(pnt%).perm
    COLOR 3, 0
    LOCATE 15, 1: PRINT "Parallel module for infiltration ........ :"
    COLOR 3, 4
    LOCATE 15, 45: PRINT USING "######"; modu(i%).infmod

     nvalues% = 13
     NBLOCK% = 1
     NCOLS% = 1
     nfield% = 6
     counter% = 1
     FOR M% = 1 TO nvalues%
        row(M%) = 2 + M%
        col(M%) = 45
     NEXT M%
     M% = 1

     flag.end% = 0

  DO
      CALL screeneditor(NBLOCK%, NCOLS%, nfield%, nvalues%, counter%, row(),
col(), a$, CHANGE%, M%, mm%, FLAGNEXT%, FLAGCURSOR%, FLAGCR%, flag.string%)

       IF flag.esc% = 1 THEN
            EXIT SUB
       END IF

      IF CHANGE% = 1 THEN
         IF mm% = 1 THEN modu(i%).modno = VAL(a$)
         IF mm% = 2 THEN modu(i%).dsmod = VAL(a$)
         IF mm% = 3 THEN
              newtype% = VAL(a$)
              EXIT SUB
         END IF

         IF mm% = 4 THEN aqui(pnt%).wid = VAL(a$)
         IF mm% = 5 THEN aqui(pnt%).lng = VAL(a$)
         IF mm% = 6 THEN aqui(pnt%).depth = VAL(a$)
         IF mm% = 7 THEN aqui(pnt%).slo = VAL(a$)
         IF mm% = 8 THEN aqui(pnt%).wtl = VAL(a$)
         IF mm% = 9 THEN aqui(pnt%).imc = VAL(a$)
         IF mm% = 10 THEN aqui(pnt%).por = VAL(a$)
         IF mm% = 11 THEN aqui(pnt%).sorp = VAL(a$)
         IF mm% = 12 THEN aqui(pnt%).perm = VAL(a$)
         IF mm% = 13 THEN modu(i%).infmod = VAL(a$)

         COLOR 7, 1
         LOCATE row(M%) + counter% - 1, col(M%)

         IF mm% <= 5 OR mm% = 13 THEN
              PRINT USING "######"; VAL(a$)
         ELSEIF mm% = 8 OR mm% = 6 OR mm% = 12 THEN
              PRINT USING "###.##"; VAL(a$)
         ELSE
              PRINT USING "##.###"; VAL(a$)
         END IF
      END IF

      IF FLAGNEXT% = 1 THEN
           FLAGNEXT% = 0' reset
           EXIT SUB
      ELSEIF FLAGCURSOR% = 1 THEN
```

```
          CALL uparrow(counter%, M%, row(), col(), NBLOCK%, NCOLS%)
          FLAGCURSOR% = 0
      ELSEIF FLAGCURSOR% = 4 THEN
          CALL downarrow(counter%, NBLOCK%, M%, row(), col(), nvalues%,
NCOLS%)
          FLAGCURSOR% = 0
      ELSEIF FLAGCR% = 0 THEN
          CALL downarrow(counter%, NBLOCK%, M%, row(), col(), nvalues%,
NCOLS%)
      END IF

  LOOP UNTIL flag.end% = 1

END SUB

SUB aquizero (p%)

    aqui(p%).depth = 0
    aqui(p%).perm = 0
    aqui(p%).sorp = 0
    aqui(p%).imc = 0
    aqui(p%).yprev = 0
    aqui(p%).slo = 0
    aqui(p%).wid = 0
    aqui(p%).lng = 0
    aqui(p%).wtl = 0
    aqui(p%).por = 0
    aqui(p%).cap = 0
    aqui(p%).volume = 0

END SUB

SUB clearscreen

      COLOR 3, 0
      FOR JJ% = 2 TO 23
         LOCATE JJ%, 1: PRINT SPACE$(79)
      NEXT JJ%

END SUB

SUB compinput (i%, newtype%)

    REDIM row(20), col(20)

    CALL clearscreen

    COLOR 0, 2
    LOCATE 23, 13
    PRINT prompt1$

    COLOR 3, 0
    LOCATE 3, 1: PRINT "Module number ..... :"
    COLOR 3, 4
    LOCATE 3, 45: PRINT USING "######"; modu(i%).modno
    COLOR 3, 0
    LOCATE 4, 1: PRINT "Downstream module ..... :"
    COLOR 3, 4
    LOCATE 4, 45: PRINT USING "######"; modu(i%).dsmod
    COLOR 3, 0
    LOCATE 5, 1: PRINT "Module type ..... :"
    COLOR 3, 4
    LOCATE 5, 45: PRINT USING "######"; modu(i%).typ

    pnt% = modu(i%).p1

    COLOR 3, 0: LOCATE 6, 1: PRINT "Length of compound channel (m) .....
```

```
:"
    COLOR 3, 4: LOCATE 6, 45: PRINT USING "######"; compchan(pnt%).lng
    COLOR 3, 0: LOCATE 7, 1: PRINT "Bed slope (m/m) ......................
:"
    COLOR 3, 4: LOCATE 7, 45: PRINT USING "##.###"; compchan(pnt%).slo
    COLOR 3, 0: LOCATE 8, 1: PRINT "No. of points describing channel .....
:"
    COLOR 3, 4: LOCATE 8, 45: PRINT USING "##.###"; compchan(pnt%).nopts
    COLOR 3, 0: LOCATE 9, 1: PRINT "No. of channel segments ..............
:"
    COLOR 3, 4: LOCATE 9, 45: PRINT USING "###.##"; compchan(pnt%).nosegs
    COLOR 3, 0: LOCATE 10, 1: PRINT "Parallel module for overflows ........
:"
    COLOR 3, 4: LOCATE 10, 45: PRINT USING "######"; modu(i%).ofl

    nvalues% = 8
    NBLOCK% = 1
    NCOLS% = 1
    nfield% = 6
    counter% = 1
    FOR M% = 1 TO nvalues%
       row(M%) = 2 + M%
       col(M%) = 45
    NEXT M%
    M% = 1

    flag.end% = 0

    DO

       CALL screeneditor(NBLOCK%, NCOLS%, nfield%, nvalues%, counter%, row(),
col(), a$, CHANGE%, M%, mm%, FLAGNEXT%, FLAGCURSOR%, FLAGCR%, flag.string%)

       IF flag.esc% = 1 THEN
           EXIT SUB
       END IF

       IF CHANGE% = 1 THEN
           IF mm% = 1 THEN modu(i%).modno = VAL(a$)
           IF mm% = 2 THEN modu(i%).dsmod = VAL(a$)
           IF mm% = 3 THEN
              newtype% = VAL(a$)
              EXIT SUB
           END IF
           IF mm% = 4 THEN compchan(pnt%).lng = VAL(a$)
           IF mm% = 5 THEN compchan(pnt%).slo = VAL(a$)
           IF mm% = 6 THEN compchan(pnt%).nopts = VAL(a$)
           IF mm% = 7 THEN compchan(pnt%).nosegs = VAL(a$)
           IF mm% = 8 THEN modu(i%).ofl = VAL(a$)

           COLOR 7, 1
           LOCATE row(M%) + counter% - 1, col(M%)

           IF mm% = 5 THEN
               PRINT USING "##.###"; VAL(a$)
           ELSEIF mm% = 4 THEN
               PRINT USING "###.##"; VAL(a$)
           ELSE
              PRINT USING "######"; VAL(a$)
           END IF

       END IF

       IF FLAGNEXT% = 1 THEN
          FLAGNEXT% = 0' reset
          GOTO screen1
       ELSEIF FLAGCURSOR% = 1 THEN
```

```
         CALL uparrow(counter%, M%, row(), col(), NBLOCK%, NCOLS%)
         FLAGCURSOR% = 0
      ELSEIF FLAGCURSOR% = 4 THEN
         CALL downarrow(counter%, NBLOCK%, M%, row(), col(), nvalues%,
NCOLS%)
         FLAGCURSOR% = 0
      ELSEIF FLAGCR% = 0 THEN
         CALL downarrow(counter%, NBLOCK%, M%, row(), col(), nvalues%,
NCOLS%)
      END IF

   LOOP UNTIL flag.end% = 1

screen1:    CALL clearscreen

   COLOR 3, 0
   LOCATE 3, 1: PRINT "Enter points describing channel section :"
   LOCATE 5, 10
   PRINT "X coordinate (m)"
   LOCATE 5, 30
   PRINT "Y coordinate (m)"

   COLOR 0, 2
   LOCATE 23, 13
   PRINT prompt1$

   COLOR 3, 4
   count% = 1
   add% = 1
   add2% = 1

   FOR M% = 1 TO 2 * compchan(pnt%).nopts

      IF count% = M% THEN
         row(M%) = 6 + add%
         col(M%) = 16
         count% = count% + 2
         LOCATE row(M%), col(M%)
         PRINT USING "####.#"; x(pnt%, add%)
         add% = add% + 1
      ELSE
         row(M%) = 6 + add2%
         col(M%) = 34
         LOCATE row(M%), col(M%)
         PRINT USING "####.#"; y(pnt%, add2%)
         add2% = add2% + 1
      END IF

   NEXT M%


   nvalues% = 2 * compchan(pnt%).nopts
   NBLOCK% = 1
   NCOLS% = 2
   nfield% = 6
   counter% = 1
   M% = 1

   flag.end% = 0

   DO

      CALL screeneditor(NBLOCK%, NCOLS%, nfield%, nvalues%, counter%, row(),
col(), a$, CHANGE%, M%, mm%, FLAGNEXT%, FLAGCURSOR%, FLAGCR%, flag.string%)

      IF flag.esc% = 1 THEN
          EXIT SUB
```

```
        END IF

        IF CHANGE% = 1 THEN

           count% = 1
           add% = 1
           add2% = 1

           FOR k% = 1 TO 2 * compchan(pnt%).nopts


              IF count% = k% THEN
                 IF M% = k% THEN
                    x(pnt%, add%) = VAL(a$)
                 ELSE
                    add% = add% + 1
                    count% = count% + 2
                 END IF
              ELSE
                 IF M% = k% THEN
                    y(pnt%, add2%) = VAL(a$)
                 ELSE
                    add2% = add2% + 1
                 END IF
              END IF

            NEXT k%

        END IF

        COLOR 7, 1
        LOCATE row(M%) + counter% - 1, col(M%)

        PRINT USING "###.##"; VAL(a$)

        IF FLAGNEXT% = 1 THEN
           FLAGNEXT% = 0' reset
           GOTO screen2
        ELSEIF FLAGCURSOR% = 1 THEN
          CALL uparrow(counter%, M%, row(), col(), NBLOCK%, NCOLS%)
          FLAGCURSOR% = 0
        ELSEIF FLAGCURSOR% = 2 THEN
          CALL leftarrow(M%, row(), col(), counter%)
          FLAGCURSOR% = 0
        ELSEIF FLAGCURSOR% = 3 THEN
          CALL rightarrow(M%, nvalues%, counter%, row(), col())
          FLAGCURSOR% = 0
        ELSEIF FLAGCURSOR% = 4 THEN
          CALL downarrow(counter%, NBLOCK%, M%, row(), col(), nvalues%,
NCOLS%)
          FLAGCURSOR% = 0
        END IF

    LOOP UNTIL flag.end% = 1

screen2:    CALL clearscreen

    LOCATE 3, 1: PRINT "Enter point numbers for channel segments :"

    LOCATE 5, 1
    PRINT "Segment no."
    LOCATE 5, 14
    PRINT "Left point"
    LOCATE 5, 26
    PRINT "Right point"
    LOCATE 5, 38
    PRINT "Roughness (n)"
```

```
COLOR 0, 2
LOCATE 23, 13
PRINT prompt1$

COLOR 3, 0
FOR k% = 1 TO compchan(pnt%).nosegs
    LOCATE 7 + k% - 1, 5
    PRINT USING "###"; k%
NEXT k%

count% = 1
count1% = 2
add% = 1
add1% = 1
add2% = 1

COLOR 3, 4
FOR M% = 1 TO 3 * compchan(pnt%).nosegs

   IF count% = M% THEN
      row(M%) = 6 + add%
      col(M%) = 16
      count% = count% + 3
      LOCATE row(M%), col(M%)
      PRINT USING "######"; segno(pnt%, add%, 1)
      add% = add% + 1
   ELSEIF count1% = M% THEN
      row(M%) = 6 + add1%
      col(M%) = 28
      LOCATE row(M%), col(M%)
      PRINT USING "######"; segno(pnt%, add1%, 2)
      add1% = add1% + 1
      count1% = count1% + 3
   ELSE
      row(M%) = 6 + add2%
      col(M%) = 40
      LOCATE row(M%), col(M%)
      PRINT USING "##.###"; n(pnt%, add2%)
      add2% = add2% + 1
   END IF

NEXT M%

nvalues% = 3 * compchan(pnt%).nosegs
NBLOCK% = 1
NCOLS% = 3
nfield% = 6
counter% = 1
M% = 1

flag.end% = 0

DO

   CALL screeneditor(NBLOCK%, NCOLS%, nfield%, nvalues%, counter%, row(),
col(), a$, CHANGE%, M%, mm%, FLAGNEXT%, FLAGCURSOR%, FLAGCR%, flag.string%)

   IF flag.esc% = 1 THEN
       EXIT SUB
   END IF

   IF CHANGE% = 1 THEN

      count% = 1
      count1% = 2
      add% = 1
      add1% = 1
```

```
            add2% = 1

            COLOR 7, 1
            LOCATE row(M%) + counter% - 1, col(M%)

            FOR k% = 1 TO 3 * compchan(pnt%).nosegs

               IF count% = k% THEN
                  IF M% = k% THEN
                     segno(pnt%, add%, 1) = VAL(a$)
                     PRINT USING "######"; VAL(a$)
                  ELSE
                     add% = add% + 1
                     count% = count% + 3
                  END IF
               ELSEIF count1% = k% THEN
                  IF M% = k% THEN
                     segno(pnt%, add1%, 2) = VAL(a$)
                     PRINT USING "######"; VAL(a$)
                  ELSE
                     add1% = add1% + 1
                     count1% = count1% + 3
                  END IF
               ELSE
                  IF M% = k% THEN
                     n(pnt%, add2%) = VAL(a$)
                     PRINT USING "##.###"; VAL(a$)
                  ELSE
                     add2% = add2% + 1
                  END IF
               END IF

            NEXT k%

         END IF


         IF FLAGNEXT% = 1 THEN
            FLAGNEXT% = 0' reset
            EXIT SUB
         ELSEIF FLAGCURSOR% = 1 THEN
           CALL uparrow(counter%, M%, row( , col(), NBLOCK%, NCOLS%)
           FLAGCURSOR% = 0
         ELSEIF FLAGCURSOR% = 2 THEN
           CALL leftarrow(M%, row(), col(), counter%)
           FLAGCURSOR% = 0
         ELSEIF FLAGCURSOR% = 3 THEN
           CALL rightarrow(M%, nvalues%, counter%, row(), col())
           FLAGCURSOR% = 0
         ELSEIF FLAGCURSOR% = 4 THEN
           CALL downarrow(counter%, NBLOCK%, M%, row(), col(), nvalues%,
NCOLS%)
           FLAGCURSOR% = 0
         END IF

    LOOP UNTIL flag.end% = 1


END SUB

SUB compzero (p%)

     FOR j% = 1 TO compchan(p%).nopts
         x(p%, j%) = 0
         y(p%, j%) = 0
     NEXT j%
```

```
      FOR j% = 1 TO compchan(p%).nosegs
          segno(p%, j%, 1) = 0
          segno(p%, j%, 2) = 0
          n(p%, j%) = 0
      NEXT j%

      compchan(p%).slo = 0
      compchan(p%).lng = 0
      compchan(p%).nosegs = 0
      compchan(p%).nopts = 0

END SUB

SUB downarrow (counter%, NBLOCK%, M%, row(), col(), nvalues%, NCOLS%)

      counter% = counter% + 1
      IF counter% > NBLOCK% THEN
         M.TEMP% = M%: M% = M% + NCOLS%
         counter% = 1 ' re-set counter
         IF M% > nvalues% THEN
           M% = M.TEMP%: counter% = NBLOCK%
           LOCATE row(M%) + NBLOCK% - 1, col(M%): COLOR 7, 0
         ELSE
           LOCATE row(M%), col(M%): COLOR 7, 0
         END IF
      ELSE
         LOCATE row(M%) + counter% - 1, col(M%)
         COLOR 7, 0
      END IF

END SUB

SUB inputeditor

       IF flag.insert% = 1 THEN
          i% = nomod% + 1
       ELSE
          i% = 1
       END IF

       flag.out% = 0

       DO

          CALL modulenumberinfo(i%)

          IF modu(i%).modno = 0 THEN
             nomod% = i% - 1
             EXIT DO
          END IF

        DO

        . typechange% = 0

          SELECT CASE modu(i%).typ
             CASE 1
                p1% = p1% + 1
                modu(i%).p1 = p1%
                IF modu(i%).infmod = 0 THEN
                    p6% = p6% + 1
                    i% = i% + 1
                    newmod% = newmod% + 1
                    modu(i%).modno = newmod%
                    modu(i% - 1).infmod = newmod%
                    modu(i%).typ = 6
                    modu(i%).p1 = p6%
```

```
        modu(i%).dsmod = 0
        aqui(p6%).depth = 9999
        newtype% = 1
        CALL overlandinput(i% - 1, newtype%)

        IF newtype% <> 1 THEN
           CALL aquizero(p6%)
           p6% = p6% - 1
           CALL moduzero(i%)
           i% = i% - 1
           CALL overlndzero(p1%)
           p1% = p1% - 1
           typechange% = 1
           modu(i%).typ = newtype%
           newtype% = 1
        END IF

    ELSE
        newtype% = 1
        CALL overlandinput(i%, newtype%)

        IF newtype% <> 1 THEN
           CALL overlndzero(p1%)
           p1% = p1% - 1
           typechange% = 1
           modu(i%).typ = newtype%
           newtype% = 1
        END IF

    END IF

CASE 2
    p2% = p2% + 1
    modu(i%).p1 = p2%
    newtype% = 2
    CALL pipeinput(i%, newtype%)

    IF newtype% <> 2 THEN
       CALL pipezero(p2%)
       p2% = p2% - 1
       modu(i%).typ = newtype%
       newtype% = 2
       typechange% = 1
    END IF

CASE 3
     p3% = p3% + 1
     modu(i%).p1 = p3%
     newtype% = 3
     CALL trapinput(i%, newtype%)

     IF newtype% <> 3 THEN
        CALL trapchanzero(p3%)
        p3% = p3% - 1
        modu(i%).typ = newtype%
        newtype% = 3
        typechange% = 1
     END IF

CASE 4
     p4% = p4% + 1
     modu(i%).p1 = p4%
     storage(p4%).typ = 1
     newtype% = 4
     CALL storinput(i%, newtype%)

     IF newtype% <> 4 THEN
```

```
                CALL storagezero(p4%)
                p4% = p4% - 1
                modu(i%).typ = newtype%
                newtype% = 4
                typechange% = 1
              END IF

          CASE 5
              p5% = p5% + 1
              modu(i%).p1 = p5%
              newtype% = 5
              CALL compinput(i%, newtype%)

              IF newtype% <> 5 THEN
                CALL compzero(p5%)
                p5% = p5% - 1
                modu(i%).typ = newtype%
                newtype% = 5
                typechange% = 1
              END IF


          CASE 6
              p6% = p6% + 1
              modu(i%).p1 = p6%
              newtype% = 6
              CALL aquifinput(i%, newtype%)

              IF newtype% <> 6 THEN
                CALL aquizero(p6%)
                p6% = p6% - 1
                modu(i%).typ = newtype%
                newtype% = 6
                typechange% = 1
              END IF

        END SELECT

      LOOP WHILE typechange% = 1

        IF flag.esc% = 1 THEN
             flag.esc% = 0
             nomod% = i% - 1
             EXIT SUB
        END IF

        IF flag.insert% = 1 THEN
           nomod% = i%
           flag.out% = 1
        ELSE
           i% = i% + 1
        END IF

     LOOP UNTIL flag.out% = 1

END SUB

SUB leftarrow (M%, row(), col(), counter%)

     M% = M% - 1
     IF M% < 1 THEN M% = 1
     LOCATE row(M%) + counter% - 1, col(M%): COLOR 7, 0

END SUB

SUB modulechange (ans$, i%)
```

```
    COLOR 3, 0

    FOR kk% = 6 TO 20
       LOCATE kk%, 1
       PRINT SPACE$(79);
    NEXT kk%

    IF modu(i%).typ = 4 THEN
        pnt% = modu(i%).p1
        IF ans$ = "Yes" OR ans$ = "yes" THEN
             storage(pnt%).typ = 1
        ELSE
             storage(pnt%).typ = 0
        END IF
    END IF

END SUB

SUB modulenumberinfo (i%)

newnum: CALL clearscreen

     IF i% > 1 THEN
       IF modu(i% - 1).modno >= 900 THEN
            COLOR 3, 4
            LOCATE 2, 1
            PRINT USING "&###"; "Previous module = "; modu(i% - 2).modno
       ELSE
            COLOR 3, 4
            LOCATE 2, 1
            PRINT USING "&###"; "Previous module = "; modu(i% - 1).modno
       END IF
     END IF

     COLOR 3, 0: LOCATE 4, 1: PRINT "Module number ..........................
:"
     LOCATE 4, 45: PRINT SPACE$(10)
     COLOR 7, 0: LOCATE 4, 45
     INPUT "", temp

     IF temp = 0 THEN
       EXIT SUB
     ELSE
       FOR zv% = 1 TO i%
           IF modu(zv%).modno = temp THEN
              BEEP: BEEP: PRINT : PRINT : PRINT "Module  number "; temp; "
has already been used ... press any key"
              WHILE INKEY$ = "": WEND
              GOTO newnum
           END IF
       NEXT zv%

       modu(i%).modno = temp

     END IF

    LOCATE 2, 1: PRINT SPACE$(30)

    COLOR 3, 0
    LOCATE 5, 1: PRINT "Downst eam module ..................... :"
    COLOR 7, 0
    LOCATE 5, 45: INPUT "", modu(i%).dsmod

    COLOR 3, 0: LOCATE 7, 1: PRINT "Type of module .........................
:"
          LOCATE 8, 3: PRINT "1 = overland flow plane"
          LOCATE 9, 3: PRINT "2 = circular pipe"
```

```
          LOCATE 10, 3: PRINT "3 = trapezoidal channel"
          LOCATE 11, 3: PRINT "4 = storage basin"
          LOCATE 12, 3: PRINT "5 = compound channel"
          LOCATE 13, 3: PRINT "6 = aquifer"
      COLOR 7, 0
      LOCATE 7, 45
      INPUT "", modu(i%).typ

      SELECT CASE modu(i%).typ

          CASE 1
            COLOR 3, 0
            LOCATE 14, 1: PRINT "Parallel module for infiltration ........ :"
            COLOR 7, 0
            LOCATE 14, 45
            INPUT "", modu(i%).infmod

          CASE 2, 3, 5
            COLOR 3, 0
            LOCATE 14, 1
            PRINT "Parallel module for overflows ........... :"
            COLOR 7, 0
            LOCATE 14, 45
            INPUT "", modu(i%).ofl

      END SELECT

END SUB

SUB moduzero (p%)

      modu(p%).modno = 0
      modu(p%).ofl = 0
      modu(p%).dsmod = 0
      modu(p%).infmod = 0
      modu(p%).pl = 0

END SUB

SUB overlandinput (i%, newtype%)


      REDIM row(11), col(11)

      CALL clearscreen

      COLOR 0, 2
      LOCATE 23, 13
      PRINT prompt1$

      COLOR 3, 0
      LOCATE 3, 1: PRINT "Module number ........................... :"
      COLOR 3, 4
      LOCATE 3, 45: PRINT USING "######"; modu(i%).modno
      COLOR 3, 0
      LOCATE 4, 1: PRINT "Downstream module ....................... :"
      COLOR 3, 4
      LOCATE 4, 45: PRINT USING "######"; modu(i%).dsmod
      COLOR 3, 0
      LOCATE 5, 1: PRINT "Module type ............................. :"
      COLOR 3, 4
      LOCATE 5, 45: PRINT USING "######"; modu(i%).typ

      pnt% = modu(i%).pl
      COLOR 3, 0: LOCATE 6, 1
      PRINT "Width of catchment (m) ................. :"
      COLOR 3, 4: LOCATE 6, 45
```

```
    PRINT USING "######"; overlnd(pnt%).wid
    COLOR 3, 0: LOCATE 7, 1
    PRINT "Length of catchment(m) ..................... :"
    COLOR 3, 4: LOCATE 7, 45
    PRINT USING "######"; overlnd(pnt%).lng
    COLOR 3, 0: LOCATE 8, 1
    PRINT "Manning n of catchment ..................... :"
    COLOR 3, 4: LOCATE 8, 45
    PRINT USING "#.####"; overlnd(pnt%).man
    COLOR 3, 0: LOCATE 9, 1
    PRINT "Slope of catchment (m/m) ................... :"
    COLOR 3, 4: LOCATE 9, 45
    PRINT USING "#.####"; overlnd(pnt%).slo

    IF modu(i%).infmod >= 900 THEN

        pnt1% = modu(i% + 1).p1
        COLOR 3, 0: LOCATE 10, 1
        PRINT "Permeability (mm/h) ....................... :"
        COLOR 3, 4: LOCATE 10, 45
        PRINT USING "####.#"; aqui(pnt1%).perm
        COLOR 3, 0: LOCATE 11, 1
        PRINT "Suction head (m) ........................... :"
        COLOR 3, 4: LOCATE 11, 45
        PRINT USING "##.###"; aqui(pnt1%).sorp
        COLOR 3, 0: LOCATE 12, 1
        PRINT "Moisture content (Fraction by volume) ... :"
        COLOR 3, 4: LOCATE 12, 45
        PRINT USING "##.###"; aqui(pnt1%).imo
        COLOR 3, 0: LOCATE 13, 1
        PRINT "Porosity ................................... :"
        COLOR 3, 4: LOCATE 13, 45
        PRINT USING "##.###"; aqui(pnt1%).por
        nvalues% = 11

    ELSE

        COLOR 3, 0: LOCATE 10, 1
        PRINT "Parallel module for infiltration ........ :"
        COLOR 3, 4: LOCATE 10, 45
        PRINT USING "######"; modu(i%).infmod
        nvalues% = 8

    END IF

    NBLOCK% = 1
    NCOLS% = 1
    nfield% = 6
    counter% = 1
    FOR M% = 1 TO nvalues%
       row(M%) = 2 + M%
       col(M%) = 45
    NEXT M%
    M% = 1

    flag.out% = 0

    DO

        CALL screeneditor(NBLOCK%, NCOLS%, nfield%, nvalues%, counter%, row(),
col(), a$, CHANGE%, M%, mm%, FLAGNEXT%, FLAGCURSOR%, FLAGCR%, flag.string%)

        IF flag.esc% = 1 THEN
            EXIT SUB
        END IF

        IF CHANGE% = 1 THEN
```

```
            IF mm% = 1 THEN modu(i%).modno = VAL(a$)
            IF mm% = 2 THEN modu(i%).dsmod = VAL(a$)
            IF mm% = 3 THEN
               newtype% = VAL(a$)
               EXIT SUB
            END IF
          IF mm% = 4 THEN overlnd(pnt%).wid = VAL(a$)
          IF mm% = 5 THEN overlnd(pnt%).lng = VAL(a$)
          IF mm% = 6 THEN overlnd(pnt%).man = VAL(a$)
          IF mm% = 7 THEN overlnd(pnt%).slo = VAL(a$)

          IF modu(i%).infmod >= 900 THEN

            IF mm% = 8 THEN aqui(pnt1%).perm = VAL(a$)
            IF mm% = 9 THEN aqui(pnt1%).sorp = VAL(a$)
            IF mm% = 10 THEN aqui(pnt1%).imc = VAL(a$)
            IF mm% = 11 THEN aqui(pnt1%).por = VAL(a$)

          ELSE

            IF mm% = 8 THEN modu(i%).infmod = VAL(a$)

          END IF


          COLOR 7, 1: LOCATE row(M%) + counter% - 1, col(M%)

          IF mm% = 6 OR mm% = 7 OR mm% >= 9 THEN
             PRINT USING "#.####"; VAL(a$)
          ELSEIF mm% = 8 THEN
             PRINT USING "###.##"; VAL(a$)
          ELSE
             PRINT USING "######"; VAL(a$)
          END IF
        END IF

     IF FLAGNEXT% = 1 THEN
         FLAGNEXT% = 0' reset
         COLOR 3, 0
         EXIT SUB
         ELSEIF FLAGCURSOR% = 1 THEN
            CALL uparrow(counter%, M%, row(), col(), NBLOCK%, NCOLS%)
            FLAGCURSOR% = 0
            ELSEIF FLAGCURSOR% = 4 THEN
               CALL downarrow(counter%, NBLOCK%, M%, row(), col(), nvalues%,
NCOLS%)
               FLAGCURSOR% = 0
               ELSEIF FLAGCR% = 0 THEN
                  CALL  downarrow(counter%,  NBLOCK%,  M%,  row(),  col(),
nvalues%, NCOLS%)
               END IF

 LOOP UNTIL flag.out% = 1

END SUB

SUB overlndzero (p%)

    overlnd(p%).man = 0
    overlnd(p%).slo = 0
    overlnd(p%).lng = 0
    overlnd(p%).wid = 0

END SUB

SUB pipeinput (i%, newtype%)
```

```
    REDIM row(11), col(11)

    CALL clearscreen

    COLOR 0, 2
    LOCATE 23, 13
    PRINT prompt1$

    COLOR 3, 0
    LOCATE 3, 1: PRINT "Module number ..... :"
    COLOR 3, 4
    LOCATE 3, 45: PRINT USING "######"; modu(i%).modno
    COLOR 3, 0
    LOCATE 4, 1: PRINT "Downstream module ..... :"
    COLOR 3, 4
    LOCATE 4, 45: PRINT USING "######"; modu(i%).dsmod
    COLOR 3, 0
    LOCATE 5, 1: PRINT "Module type ..... :"
    COLOR 3, 4
    LOCATE 5, 45: PRINT USING "######"; modu(i%).typ

    pnt% = modu(i%).p1

    COLOR 3, 0: LOCATE 6, 1: PRINT "Pipe length (m) .....
:"
    COLOR 3, 4: LOCATE 6, 45: PRINT USING "######"; pipe(pnt%).lng
    COLOR 3, 0: LOCATE 7, 1: PRINT "Slope (m/m) .....
:"
    COLOR 3, 4: LOCATE 7, 45: PRINT USING "##.###"; pipe(pnt%).slo
    COLOR 3, 0: LOCATE 8, 1: PRINT "Roughness n .....
:"
    COLOR 3, 4: LOCATE 8, 45: PRINT USING "##.###"; pipe(pnt%).man
    COLOR 3, 0: LOCATE 9, 1: PRINT "Pipe diameter (m) .....
:"
    COLOR 3, 4: LOCATE 9, 45: PRINT USING "###.##"; pipe(pnt%).diam
    COLOR 3, 0: LOCATE 10, 1: PRINT "Parallel module for overflows .....
:"
    COLOR 3, 4: LOCATE 10, 45: PRINT USING "######"; modu(i%).ofl

     nvalues% = 8
     NBLOCK% = 1
     NCOLS% = 1
     nfield% = 6
     counter% = 1
     FOR M% = 1 TO nvalues%
          row(M%) = 2 + M%
          col(M%) = 45
     NEXT M%
     M% = 1

     flag.end% = 0

  DO
      CALL screeneditor(NBLOCK%, NCOLS%, nfield%, nvalues%, counter%, row(),
col(), a$, CHANGE%, M%, mm%, FLAGNEXT%, FLAGCURSOR%, FLAGCR%, flag.string%)

       IF flag.esc% = 1 THEN
          EXIT SUB
       END IF

      IF CHANGE% = 1 THEN
          IF mm% = 1 THEN modu(i%).modno = VAL(a$)
          IF mm% = 2 THEN modu(i%).dsmod = VAL(a$)
          IF mm% = 3 THEN
             newtype% = VAL(a$)
             EXIT SUB
          END IF
```

```
            IF mm% = 4 THEN pipe(pnt%).lng = VAL(a$)
            IF mm% = 5 THEN pipe(pnt%).slo = VAL(a$)
            IF mm% = 6 THEN pipe(pnt%).man = VAL(a$)
            IF mm% = 7 THEN pipe(pnt%).diam = VAL(a$)
            IF mm% = 8 THEN modu(i%).ofl = VAL(a$)

            COLOR 7, 1
            LOCATE row(M%) + counter% - 1, col(M%)

            IF mm% = 5 OR mm% = 6 THEN
                PRINT USING "##.###"; VAL(a$)
            ELSEIF mm% = 7 THEN
              PRINT USING "###.##"; VAL(a$)
            ELSE
              PRINT USING "######"; VAL(a$)
            END IF
        END IF

        IF FLAGNEXT% = 1 THEN
             FLAGNEXT% = 0' reset
             EXIT SUB
        ELSEIF FLAGCURSOR% = 1 THEN
             CALL uparrow(counter%, M%, row(), col(), NBLOCK%, NCOLS%)
             FLAGCURSOR% = 0
        ELSEIF FLAGCURSOR% = 4 THEN
             CALL  downarrow(counter%,  NBLOCK%,  M%,  row(),  col(),  nvalues%,
NCOLS%)
             FLAGCURSOR% = 0
        ELSEIF FLAGCR% = 0 THEN
             CALL  downarrow(counter%,  NBLOCK%,  M%,  row(),  col(),  nvalues%,
NCOLS%)
        END IF

    LOOP UNTIL flag.out% = 1


END SUB

SUB pipezero (p%)

        pipe(p%).slo = 0
        pipe(p%).diam = 0
        pipe(p%).lng = 0
        pipe(p%).man = 0
        pipe(p%).cap = 0
        pipe(p%).min = 0

END SUB

SUB raininput

        REM $DYNAMIC

        REDIM rain.temp(100), row(100), col(100)

        clearscreen

        COLOR 3, 0
        CLS
        COLOR 0, 2
        LOCATE 1, 28: PRINT " WITSKM - Rain entry mode ": COLOR 3, 0

        COLOR 0, 2
        LOCATE 23, 13
        PRINT prompt1$

        COLOR 3, 0
```

```
    LOCATE 6, 1
    PRINT " Rainfall Intensities in mm/hr :"

    hyeto.number.prev% = hyeto.number%
    hyeto.number% = CINT(rain.time / tint)
    ratio = (hyeto.number% / hyeto.number.prev%)

    FOR i% = 1 TO hyeto.number.prev%
       rain.temp(i%) = rain(i%)
       rain(i%) = 0!
    NEXT i%

    IF ratio >= 1 THEN
      k = 1
      FOR i% = 1 TO hyeto.number.prev%
         l = i% * ratio
         FOR j% = k TO l
            rain(j%) = rain.temp(i%)
         NEXT j%
         k = l + 1
      NEXT i%

   ELSE

      j% = 0

      FOR i% = (1 / ratio) TO hyeto.number.prev% STEP (1 / ratio)
        j% = j% + 1
        rain(j%) = rain.temp(i%)
      NEXT i%

  END IF

  COLOR 3, 4

  FOR i% = 1 TO hyeto.number%

     IF i% > 10 THEN
         col(i%) = ((i% * 8) - 7) - 80 * INT((i% - 1) / 10)
     ELSE
         col(i%) = (i% * 8) - 7
     END IF

     row(i%) = 10 + INT((i% - 1) / 10) - 1

     LOCATE row(i%), col(i%)
     PRINT USING "####.#"; rain(i%)

  NEXT i%

  ERASE rain.temp

  COLOR 3, 0

  nvalues% = hyeto.number%
  NBLOCK% = 1
  NCOLS% = 10
  nfield% = 6
  counter% = 1
  M% = 1

  flag.end% = 0

 DO
     CALL screeneditor(NBLOCK%, NCOLS%, nfield%, nvalues%, counter%, row(),
col(), a$, CHANGE%, M%, mm%, FLAGNEXT%, FLAGCURSOR%, FLAGCR%, flag.string%)
```

```
        IF flag.esc% = 1 THEN
             ERASE row, col
             EXIT SUB
        END IF

        IF CHANGE% = 1 THEN
           rain(mm%) = VAL(a$)
           LOCATE row(M%) + counter% - 1, col(M%)
           COLOR 7, 1
           PRINT USING "#####.#"; VAL(a$)
        END IF

        IF FLAGNEXT% = 1 THEN
           FLAGNEXT% = 0' reset
           ERASE row, col
           EXIT SUB
        END IF
        IF FLAGCURSOR% = 1 THEN
           uparrow counter%, M%, row(), col(), NBLOCK%, NCOLS%
           FLAGCURSOR% = 0
        END IF
        IF FLAGCURSOR% = 2 THEN
           leftarrow M%, row(), col(), counter%
           FLAGCURSOR% = 0
        END IF
        IF FLAGCURSOR% = 3 THEN
           rightarrow M%, nvalues%, counter%, row
           FLAGCURSOR% = 0
        END IF
        IF FLAGCURSOR% = 4 THEN
           downarrow counter%, NBLOCK%, M%, row(), col(), nva
           FLAGCURSOR% = 0
        END IF

  LOOP UNTIL flag.end% = 1


END SUB

REM $STATIC
SUB rightarrow (M%, nvalues%, counter%, row(), col())

     M% = M% + 1
     IF M% > nvalues% THEN M% = nvalues%
     LOCATE row(M%) + counter% - 1, col(M%): COLOR 7, 0

END SUB

SUB screeneditor (NBLOCK%, NCOLS%, nfield%, nvalues%, counter%, row(), col(),
a$, CHANGE%, M%, mm%, FLAGNEXT%, FLAGCURSOR%, FLAGCR%, flag.string%)

     FLAGCR% = 1'set flag
     a$ = ""
     b$ = ""
     LOCATE , , 1, 6, 7
     LOCATE row(M%) + counter% - 1, col(M%)

     WHILE FLAGCR%
        mm% = M%
        CHANGE% = 0
7120    a$ = INKEY$: IF a$ = "" THEN 7120

        SELECT CASE a$

           CASE CHR$(27)
              flag.esc% = 1
              FLAGCR% = 0
```

```
            GOTO 7410' escape

         CASE CHR$(32)
            FLAGNEXT% = 1
            FLAGCR% = 0
            GOTO 7410

         CASE CHR$(13)
            FLAGCR% = 0
            GOTO 7410

      END SELECT

      IF LEN(a$) > 1 THEN

          arrowkey = ASC(MID$(a$, 2, 1))

          SELECT CASE arrowkey

             CASE 72, 91
                CALL uparrow(counter%, M%, row(), col(), NBLOCK%, NCOLS%)
                GOTO 7410

             CASE 75, 88
                CALL leftarrow(M%, row(), col(), counter%)
                GOTO 7410

             CASE 90, 77
                CALL rightarrow(M%, nvalues%, counter%, row(), col())
                GOTO 7410

             CASE 80, 89
                CALL  downarrow(counter%,  NBLOCK%,  M%,  row(),  col(),
nvalues%, NCOLS%)
                GOTO 7410

          END SELECT

      ELSE

7210      COLOR 7, 0: LOCATE row(M%) + counter% - 1, col(M%)
          PRINT SPACE$(nfield%)' else clear rest of field
          LOCATE row(M%) + counter% - 1, col(M%)
          PRINT a$;
          ichar% = 1
          CHANGE% = 1

          WHILE FLAGCR%

7260         b$ = INKEY$: IF b$ = "" THEN 7260

             IF b$ = CHR$(27) THEN
                   flag.esc% = 1
                   FLAGCR% = 0
                   GOTO 7410
             END IF

             IF flag.string% = 0 THEN
                  IF b$ = CHR$(32) THEN
                      FLAGNEXT% = 1
                      FLAGCR% = 0
                      GOTO 7410
                  END IF
             END IF

             IF b$ = CHR$(13) THEN
                FLAGCR% = 0
```

```
                GOTO 7400
            END IF

            IF LEN(b$) > 1 THEN

                arrowkey = ASC(MID$(b$, 2, 1))

                SELECT CASE arrowkey

                    CASE 72, 91
                        FLAGCURSOR% = 1
                        FLAGCR% = 0
                        GOTO 7400

                    CASE 75, 88
                        FLAGCURSOR% = 2
                        FLAGCR% = 0
                        GOTO 7400

                    CASE 90, 77
                        FLAGCURSOR% = 3
                        FLAGCR% = 0
                        GOTO 7400

                    CASE 80, 89
                        FLAGCURSOR% = 4
                        FLAGCR% = 0
                        GOTO 7400

                END SELECT

            ELSE

                IF b$ = CHR$(8) THEN
                    e$ = a$: a$ = ""
                    ichar% = ichar% - 1
                    a$ = MID$(e$, 1, LEN(e$) - 1)
                    LOCATE row(M%) + counter% - 1, col(M%)
                    PRINT SPACE$(nfield%)
                    LOCATE row(M%) + counter% - 1, col(M%)
                    PRINT a$;
                ELSE
                    a$ = a$ + b$
                    ichar% = ichar% + 1
                    IF ichar% > nfield% THEN
                        a$ = MID$(a$, 1, nfield%)
                        BEEP: GOTO 7260
                    END IF
                    LOCATE row(M%) + counter% - 1, col(M%) + ichar% -
1
                    COLOR 7, 0
                    PRINT b$
                    LOCATE row(M%) + counter% - 1  col(M%) + ichar%
                    b$ = ""
                END IF
            END IF
7400        WEND
        END IF
7410 WEND

    LOCATE , , 0

END SUB

SUB storagezero (p%)

    storage(p%).cl = 0
```

```
        storage(p%).sl = 0
        storage(p%).a = 0
        storage(p%).b = 0
        storage(p%).stlev = 0
        storage(p%).typ = 1
        storage(p%).ccu = 0
        storage(p%).cu = 0
        storage(p%).ccs = 0
        storage(p%).cs = 0
        storage(p%).csp = 0
        storage(p%).sp = 0
        storage(p%).depth = 0
        storage(p%).crit1 = 0
        storage(p%).crit2 = 0
        storage(p%).crit3 = 0
        storage(p%).prevstor = 0

END SUB

SUB storinput (i%, newtype%)

    REDIM row(17), col(17)

start: CALL clearscreen

    COLOR 0, 2
    LOCATE 23, 13
    PRINT prompt1$

    COLOR 3, 0
    LOCATE 3, 1: PRINT "Module number ..... :"
    COLOR 3, 4
    LOCATE 3, 45: PRINT USING "######"; modu(i%).modno
    COLOR 3, 0
    LOCATE 4, 1: PRINT "Downstream module ..... :"
    COLOR 3, 4
    LOCATE 4, 45: PRINT USING "######"; modu(i%).dsmod
    COLOR 3, 0
    LOCATE 5, 1: PRINT "Module type ..... :"
    COLOR 3, 4
    LOCATE 5, 45: PRINT USING "######"; modu(i%).typ

    pnt% = modu(i%).p1

    COLOR 3, 0: LOCATE 6, 1: PRINT "Coeff a in stor(m^3)=a*(depth(m))^b .....
:"
    COLOR 3, 4: LOCATE 6, 45: PRINT USING "####.#"; storage(pnt%).a
    COLOR 3, 0: LOCATE 7, 1
    PRINT "Coeff b in stor(m^3)=a*(depth(m))^b..... :"
    COLOR 3, 4: LOCATE 7, 45: PRINT USING "##.###"; storage(pnt%).b
    COLOR 3, 0: LOCATE 8, 1
    PRINT "Spillway level (m) ..... :"
    COLOR 3, 4: LOCATE 8, 45: PRINT USING "###.##"; storage(pnt%).sl
    COLOR 3, 0: LOCATE 9, 1
    PRINT "Outlet ..... :"

   IF storage(pnt%).typ = 1 THEN
       COLOR 3, 4: LOCATE 9, 45: PRINT "   Yes"
   ELSE
       COLOR 3, 4: LOCATE 9, 45: PRINT "    No"
   END IF

       IF storage(pnt%).typ = 1 THEN
           COLOR 3, 0
           LOCATE 10, 1: PRINT "Outlet invert level (m) ..... :"
           COLOR 3, 4
           LOCATE 10, 45: PRINT USING "###.##"; storage(pnt%).cl
```

```
            COLOR 3, 0
            LOCATE 11, 1: PRINT "Coeff c for unsubmerged outlet c*dep^d .. :"
            COLOR 3, 4
            LOCATE 11, 45: PRINT USING "###.##"; storage(pnt%).ccu
            COLOR 3, 0
            LOCATE 12, 1: PRINT "Coeff d for unsubmerged outlet c*dep^d .. :"
            COLOR 3, 4
            LOCATE 12, 45: PRINT USING "###.##"; storage(pnt%).cu
            COLOR 3, 0
            LOCATE 13, 1: PRINT "Coeff e for submerged outlet e*dep^f .... :"
            COLOR 3, 4
            LOCATE 13, 45: PRINT USING "###.##"; storage(pnt%).ccs
            COLOR 3, 0
            LOCATE 14, 1: PRINT "Coeff f for submerged outlet e*dep^f .... :"
            COLOR 3, 4
            LOCATE 14, 45: PRINT USING "###.##"; storage(pnt%).cs
            COLOR 3, 0
            LOCATE 15, 1: PRINT "Coeff g for spillway  g*dep^h ........... :"
            COLOR 3, 4
            LOCATE 15, 45: PRINT USING "####.#"; storage(pnt%).csp
            COLOR 3, 0
            LOCATE 16, 1: PRINT "Coeff h for spillway  g*dep^h ........... :"
            COLOR 3, 4
            LOCATE 16, 45: PRINT USING "##.###"; storage(pnt%).sp
            COLOR 3, 0
            LOCATE 17, 1: PRINT "Depth or diam (m) of outlet ............. :"
            COLOR 3, 4
            LOCATE 17, 45: PRINT USING "##.###"; storage(pnt%).depth
            COLOR 3, 0
            LOCATE 18, 1: PRINT "Initial water level in dam (m) .......... :"
            COLOR 3, 4
            LOCATE 18, 45: PRINT USING "###.##"; storage(pnt%).stlev
            nvalues% = 16
        ELSE
            COLOR 3, 0
            LOCATE 10, 1: PRINT "Coeff g for spillway  g*dep^h ........... :"
            COLOR 3, 4
            LOCATE 10, 45: PRINT USING "####.#"; storage(pnt%).csp
            COLOR 3, 0
            LOCATE 11, 1: PRINT "Coeff h for spillway  g*dep^h ........... :"
            COLOR 3, 4
            LOCATE 11, 45: PRINT USING "##.###"; storage(pnt%).sp
            COLOR 3, 0
            LOCATE 12, 1: PRINT "Initial water level in dam (m) .......... :"
            COLOR 3, 4
            LOCATE 12, 45: PRINT USING "###.##"; storage(pnt%).stlev
            nvalues% = 10
        END IF

        NBLOCK% = 1
        NCOLS% = 1
        nfield% = 6
        counter% = 1

        FOR M% = 1 TO nvalues%
          row(M%) = 2 + M%
          col(M%) = 45
        NEXT M%
        M% = 1

      flag.end% = 0

   DO
       CALL screeneditor(NBLOCK%, NCOLS%, nfield%, nvalues%, counter%, row(),
col(), a$, CHANGE%, M%, mm%, FLAGNEXT%, FLAGCURSOR%, FLAGCR%, flag.string%)

        IF flag.esc% = 1 THEN
```

```
        EXIT SUB
    END IF

IF CHANGE% = 1 THEN
    IF mm% = 1 THEN modu(i%).modno = VAL(a$)
    IF mm% = 2 THEN modu(i%).dsmod = VAL(a$)
    IF mm% = 3 THEN
        newtype% = VAL(a$)
        EXIT SUB
    END IF
    IF mm% = 4 THEN storage(pnt%).a = VAL(a$)
    IF mm% = 5 THEN storage(pnt%).b = VAL(a$)
    IF mm% = 6 THEN storage(pnt%).sl = VAL(a$)
    IF mm% = 7 THEN
       ans$ = a$
       CALL modulechange(ans$, i%)
       GOTO start
    END IF
    IF storage(pnt%).typ = 1 THEN
       IF mm% = 8 THEN storage(pnt%).cl = VAL(a$)
       IF mm% = 9 THEN storage(pnt%).csu = VAL(a$)
       IF mm% = 10 THEN storage(pnt%).cu = VAL(a$)
       IF mm% = 11 THEN storage(pnt%).ccs = VAL(a$)
       IF mm% = 12 THEN storage(pnt%).cs = VAL(a$)
       IF mm% = 13 THEN storage(pnt%).csp = VAL(a$)
       IF mm% = 14 THEN storage(pnt%).sp = VAL(a$)
       IF mm% = 15 THEN storage(pnt%).depth = VAL(a$)
       IF mm% = 16 THEN storage(pnt%).stlev = VAL(a$)

       COLOR 7, 1: LOCATE row(M%) + counter% - 1, col(M%)
       IF mm% <= 3 THEN
          PRINT USING "######"; VAL(a$)
       ELSEIF mm% = 4 OR mm% = 13 THEN
          PRINT USING "####.#"; VAL(a$)
       ELSEIF mm% = 5 OR mm% = 15 OR mm% = 14 THEN
          PRINT USING "##.###"; VAL(a$)
       ELSE
          PRINT USING "###.##"; VAL(a$)
       END IF

    ELSE
       IF mm% = 8 THEN storage(pnt%).csp = VAL(a$)
       IF mm% = 9 THEN storage(pnt%).sp = VAL(a$)
       IF mm% = 10 THEN storage(pnt%).stlev = VAL(a$)

       COLOR 7, 1: LOCATE row(M%) + counter% - 1, col(M%)

       IF mm% <= 3 THEN
           PRINT USING "######"; VAL(a$)
       ELSEIF mm% = 4 OR mm% = 8 THEN
           PRINT USING "####.#"; VAL(a$)
       ELSEIF mm% = 6 OR mm% = 10 THEN
           PRINT USING "###.##"; VAL(a$)
       ELSE
           PRINT USING "##.###"; VAL(a$)
       END IF

    END IF

END IF

IF FLAGNEXT% = 1 THEN
    FLAGNEXT% = 0' reset
    EXIT SUB
ELSEIF FLAGCURSOR% = 1 THEN
    CALL uparrow(counter%, M%, row(), col(), NBLOCK%, NCOLS%)
    FLAGCURSOR% = 0
```

```
      ELSEIF FLAGCURSOR% = 4 THEN
          CALL downarrow(counter%, NBLOCK%, M%, row(), col(), nvalues%, NCOLS%)
          FLAGCURSOR% = 0
      ELSEIF FLAGCR% = 0 THEN
          CALL downarrow(counter%, NBLOCK%, M%, row(), col(), nvalues%, NCOLS%)
      END IF

  LOOP UNTIL flag.end% = 1

END SUB

SUB TimestepInfo

      REDIM row(3), col(3)

      clearscreen

      COLOR 0, 2
      LOCATE 23, 13
      PRINT prompt1$


      COLOR 3, 0
      LOCATE 4, 1
      PRINT "Time interval (hours) ..................... :"
      COLOR 3, 4
      LOCATE 4, 45
      PRINT USING "#.###"; tint
      COLOR 3, 0
      LOCATE 6, 1
      PRINT "Simulation duration (hours) ............. :  "
      COLOR 3, 4
      LOCATE 6, 45
      PRINT USING "##.##"; sim.time
      COLOR 3, 0
      LOCATE 8, 1
      PRINT "Rainfall duration (hours) ............... :"
      COLOR 3, 4
      LOCATE 8, 45
      PRINT USING "##.##"; rain.time

      nvalues% = 3
      NBLOCK% = 1
      NCOLS% = 1
      nfield% = 5
      counter% = 1

      FOR M% = 1 TO nvalues%
         row(M%) = 2 + 2 * M%
         col(M%) = 45
      NEXT M%

      M% = 1

     flag.end% = 0

     DO

        CALL screeneditor(NBLOCK%, NCOLS%, nfield%, nvalues%, counter%, row(),
col(), a$, CHANGE%, M%, mm%, FLAGNEXT%, FLAGCURSOR%, FLAGCR%, flag.string%)

        IF flag.esc% = 1 THEN
            EXIT SUB
        END IF

      IF CHANGE% = 1 THEN
```

```
            IF mm% = 1 THEN tint = VAL(a$)
            IF mm% = 2 THEN sim.time = VAL(a$)
            IF mm% = 3 THEN rain.time = VAL(a$)
            noit% = CINT(sim.time / tint)


            COLOR 7, 1
            LOCATE row(M%) + counter% - 1, col(M%)

            IF mm% = 1 THEN
                PRINT USING "#.###"; VAL(a$)
            ELSE
                PRINT USING "##.##"; VAL(a$)
            END IF

        END IF

        IF FLAGNEXT% = 1 THEN
            FLAGNEXT% = 0' reset
            EXIT SUB
        ELSEIF FLAGCURSOR% = 1 THEN
            uparrow counter%, M%, row(), col(), NBLOCK%, NCOLS%
            FLAGCURSOR% = 0
        ELSEIF FLAGCURSOR% = 4 THEN
            downarrow counter%, NBLOCK%, M%, row(), col(), nvalues%, NCOLS%
            FLAGCURSOR% = 0
        ELSEIF FLAGCR% = 0 THEN
            downarrow counter%, NBLOCK%, M%, row(), col(), nvalues%, NCOLS%
        END IF

    LOOP UNTIL flag.end% = 1

END SUB

SUB TitleEnter

        clearscreen

        COLOR 3, 0
        LOCATE 3, 1
        PRINT "Enter a title describing the data set ...."
        LOCATE 5, 1
        PRINT "Title line"
        COLOR 7, 0
        LOCATE 5, 12
        LINE INPUT "", title$

END SUB

SUB trapchanzero (p%)

        trapchan(p%).slo = 0
        trapchan(p%).lng = 0
        trapchan(p%).man = 0
        trapchan(p%).wid = 0
        trapchan(p%).ss1 = 0
        trapchan(p%).ss2 = 0
        trapchan(p%).mdep = 0
        trapchan(p%).cap = 0

END SUB

SUB trapinput (i%, newtype%)

        REDIM row(11), col(11)

        CALL clearscreen
```

```
        COLOR 0, 2
        LOCATE 23, 13
        PRINT prompt1$

        COLOR 3, 0
        LOCATE 3, 1: PRINT "Module number ..... :"
        COLOR 3, 4
        LOCATE 3, 45: PRINT USING "######"; modu(i%).modno
        COLOR 3, 0
        LOCATE 4, 1: PRINT "Downstream module ..... :"
        COLOR 3, 4
        LOCATE 4, 45: PRINT USING "######"; modu(i%).dsmod
        COLOR 3, 0
        LOCATE 5, 1: PRINT "Module type ..... :"
        COLOR 3, 4
        LOCATE 5, 45: PRINT USING "######"; modu(i%).typ

        pnt% = modu(i%).p1

         COLOR 3, 0: LOCATE 6, 1: PRINT "Length of trapezoidal channel (m)
....... :"
        COLOR 3, 4: LOCATE 6, 45: PRINT USING "######"; trapchan(pnt%).lng
        COLOR 3, 0: LOCATE 7, 1: PRINT "Bed slope (m/m)
..... :"
        COLOR 3, 4: LOCATE 7, 45: PRINT USING "##.###"; trapchan(pnt%).slo
        COLOR 3, 0: LOCATE 8, 1: PRINT "Roughness n
..... :"
        COLOR 3, 4: LOCATE 8, 45: PRINT USING "##.###"; trapchan(pnt%).man
        COLOR 3, 0: LOCATE 9, 1: PRINT "Base width of trapezoidal channel
(m).... :"
        COLOR 3, 4: LOCATE 9, 45: PRINT USING "###.##"; trapchan(pnt%).wid
        COLOR 3, 0: LOCATE 10, 1: PRINT "LH side slope (horiz/vert)
..... :"
        COLOR 3, 4: LOCATE 10, 45: PRINT USING "###.##"; trapchan(pnt%).ss1
        COLOR 3, 0: LOCATE 11, 1: PRINT "RH side slope (horiz/vert)
..... :"
        COLOR 3, 4: LOCATE 11, 45: PRINT USING "###.##"; trapchan(pnt%).ss2
        COLOR 3, 0: LOCATE 12, 1: PRINT "Maximum flow depth (m)
..... :"
        COLOR 3, 4: LOCATE 12, 45: PRINT USING "###.##"; trapchan(pnt%).mdep
        COLOR 3, 0: LOCATE 13, 1: PRINT "Parallel module for overflows
..... :"
        COLOR 3, 4: LOCATE 13, 45: PRINT USING "######"; modu(i%).ofl

        nvalues% = 11
        NBLOCK% = 1
        NCOLS% = 1
        nfield% = 6
        counter% = 1
        FOR M% = 1 TO nvalues%
             row(M%) = 2 + M%
             col(M%) = 45
        NEXT M%
        M% = 1

        flag.end% = 0

  DO

      CALL screeneditor(NBLOCK%, NCOLS%, nfield%, nvalues%, counter%, row(),
col(), a$; CHANGE%, M%, mm%, FLAGNEXT%, FLAGCURSOR%, FLAGCR%, flag.string%)

      IF flag.esc% = 1 THEN
           EXIT SUB
      END IF

      IF CHANGE% = 1 THEN
```

```
            IF mm% = 1 THEN modu(i%).modno = VAL(a$)
            IF mm% = 2 THEN modu(i%).dsmod = VAL(a$)
            IF mm% = 3 THEN
               newtype% = VAL(a$)
               EXIT SUB
            END IF
            IF mm% = 4 THEN trapchan(pnt%).lng = VAL(a$)
            IF mm% = 5 THEN trapchan(pnt%).slo = VAL(a$)
            IF mm% = 6 THEN trapchan(pnt%).man = VAL(a$)
            IF mm% = 7 THEN trapchan(pnt%).wid = VAL(a$)
            IF mm% = 8 THEN trapchan(pnt%).ss1 = VAL(a$)
            IF mm% = 9 THEN trapchan(pnt%).ss2 = VAL(a$)
            IF mm% = 10 THEN trapchan(pnt%).mdep = VAL(a$)
            IF mm% = 11 THEN modu(i%).ofl = VAL(a$)

            COLOR 7, 1
            LOCATE row(M%) + counter% - 1, col(M%)
            IF mm% = 5 OR mm% = 6 OR mm% = 10 THEN
                PRINT USING "##.###"; VAL(a$)
            ELSEIF mm% = 7 OR mm% = 8 OR mm% = 9 THEN
                PRINT USING "###.##"; VAL(a$)
            ELSE
                PRINT USING "######"; VAL(a$)
            END IF
        END IF

        IF FLAGNEXT% = 1 THEN
            FLAGNEXT% = 0' reset
            EXIT SUB
        ELSEIF FLAGCURSOR% = 1 THEN
            CALL uparrow(counter%, M%, row(), col(), NBLOCK%, NCOLS%)
            FLAGCURSOR% = 0
        ELSEIF FLAGCURSOR% = 4 THEN
            CALL downarrow(counter%, NBLOCK%, M%, row(), col(), nvalues%,
NCOLS%)
            FLAGCURSOR% = 0
        ELSEIF FLAGCR% = 0 THEN
           CALL downarrow(counter%, NBLOCK%, M%, row(), col(), nvalues%, NCOLS%)
        END IF

    LOOP UNTIL flag.end% = 1

END SUB

SUB uparrow (counter%, M%, row(), col(), NBLOCK%, NCOLS%)

      counter% = counter% - 1
      IF counter% < 1 THEN
         M.TEMP% = M%: M% = M% - NCOLS%
         counter% = NBLOCK% 're-set counter
         IF M% < 1 THEN M% = M.TEMP%: counter% = 1' can't move higher than high
         LOCATE row(M%) + counter% - 1, col(M%): COLOR 7, 0
      ELSE
         LOCATE row(M%) + counter% - 1, col(M%): COLOR 7, 0
      END IF

END SUB

DECLARE SUB connecttable ()
DECLARE SUB setuppointermatrix (pointer(), index())
DECLARE SUB orderofcalculation (pointer!(), index!())
DECLARE SUB check (modnum!, order!(), count%, flag1!, flag2!)
DECLARE SUB overaquistack ()
DECLARE SUB upstreammods ()
DECLARE FUNCTION areapipe! (angle, pnt%)
DECLARE FUNCTION storvol! (pnt%, y!)
```

```
TYPE modconnectivity
      modno AS INTEGER
      ofl AS INTEGER
      typ AS INTEGER
      dsmod AS INTEGER
      infmod AS INTEGER
      pl AS INTEGER
END TYPE

TYPE overmod
     man AS SINGLE
     slo AS SINGLE
     lng AS SINGLE
     wid AS SINGLE
END TYPE

TYPE aquimod
    slo AS SINGLE
    wid AS SINGLE
    lng AS SINGLE
    depth AS SINGLE
    wtl AS SINGLE
    sorp AS SINGLE
    perm AS SINGLE
    por AS SINGLE
    imc AS SINGLE
    cap AS SINGLE
    volume AS SINGLE
    yprev AS SINGLE
END TYPE

TYPE pipemod
    slo AS SINGLE
     diam AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     cap AS SINGLE
     min AS SINGLE
END TYPE

TYPE trapmod
      slo AS SINGLE
      lng AS SINGLE
      man AS SINGLE
      wid AS SINGLE
      ss1 AS SINGLE
      ss2 AS SINGLE
      mdep AS SINGLE
      cap AS SINGLE
END TYPE

TYPE stormod
     cl AS SINGLE
     sl AS SINGLE
     a AS SINGLE
     b AS SINGLE
     stlev AS SINGLE
     typ AS INTEGER
     ccu AS SINGLE
     cu AS SINGLE
     ccs AS SINGLE
     cs AS SINGLE
     csp AS SINGLE
     sp AS SINGLE
     depth AS SINGLE
     crit1 AS SINGLE
     crit2 AS SINGLE
```

```
    crit3 AS SINGLE
    prevstor AS SINGLE
END TYPE

TYPE compmod
    slo AS SINGLE
    lng AS SINGLE
    nosegs AS INTEGER
    nopts AS INTEGER
END TYPE

COMMON SHARED modu() AS modconnectivity, pipe() AS pipemod, storage() AS
stormod
COMMON SHARED overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON SHARED qin(), qout(), unit(), con(), ovflo(), order(), nomod%, noit%,
tint
COMMON SHARED m, cver(), rain(), pi
COMMON SHARED prompt1$, flag.file%, tcode%
COMMON SHARED title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON SHARED sim.time, rain.time, hyeto.number%, newmod%, flag.insert%
COMMON SHARED compchan() AS compmod, x(), y(), n(), segno()

errorproc1:

      SELECT CASE ERR
            CASE 25, 24, 68

               BEEP: BEEP: PRINT : PRINT : PRINT "printer not connected or
switched on ... press any key"
               WHILE INKEY$ = "": WEND
               RESUME
            CASE ELSE
               ON ERROR GOTO 0
      END SELECT



END

SUB check (modnum, order(), count%, flag1, flag2)

    mark% = 0
    FOR j% = 1 TO count% - 1
       IF modnum = order(1, j%) THEN
          flag2 = flag2 + 1
          mark% = 1
          EXIT FOR
       END IF
    NEXT j%
    IF mark% = 0 THEN
       flag1 = 1
    END IF

END SUB

SUB connecttable

      ON ERROR GOTO errorproc1

      WIDTH LPRINT 80
      LPRINT CHR$(18);
                              l   i   n   e   1   $          =          "
-------------------------------------------------------------------------"
      title1$ = "   : no : mod no :     upstream module nos
                :"
      LPRINT line1$
      LPRINT title1$
```

```
      FOR k% = 1 TO nomod%
         LPRINT line1$
         LPRINT "   : ";
         LPRINT USING "###"; k%;
         LPRINT " : ";
         LPRINT USING "###"; modu(k%).modno;
         LPRINT "    : ";
         FOR j% = 1 TO 10
             IF j% <= con(k%, 0) THEN
                LPRINT USING "###"; modu(con(k%, j%)).modno;
                LPRINT " : ";
             ELSE
                LPRINT "--- : ";
             END IF
         NEXT j%
      NEXT k%
      LPRINT line1$
      LPRINT " modules 900 and up are dummy aquifers"
      LPRINT

      LPRINT
      line1$ = "  ------------------------------------------------"
      title1$ = "   : no : mod no :      overflow module nos      :"
      LPRINT line1$
      LPRINT title1$

      FOR k% = 1 TO nomod%

         LPRINT line1$
         LPRINT "   : ";
         LPRINT USING "##"; k%;
         LPRINT " :   ";
         LPRINT USING "###"; modu(k%).modno;
         LPRINT "    : ";

         FOR j% = 1 TO 5
            IF j% <= ovflo(k%, 0) THEN
               LPRINT USING "###"; modu(ovflo(k%, j%)).modno;
               LPRINT " : ";
            ELSE
               LPRINT "--- : ";
            END IF
         NEXT j%

      NEXT k%
      LPRINT line1$
      LPRINT " modules 900 and up are dummy aquifers"
      LPRINT

END SUB

SUB modconnect (print.connec%)

    CLS
    LOCATE 11, 28
    COLOR 0, 2
    PRINT "Calculating connectivity"
    COLOR 3, 0


    REM $DYNAMIC

    REDIM pointer(2, 141), index(165)


'   Call sub upstreammods to determine the upstream modules and the
'   modules which flow onto each module.
```

```
'    The number of upstream modules and their numbers are stored in array
'    CON and the overflow modules in array OVFLO

     CALL upstreammods

'    Call overaquistack to determine the overland and associated aquifers
'    The array UNIT is used to store the information.

     CALL overaquistack

'    Call subs setuppointermatrix and orderofcalculation to determine the
'    upstream - downstream order of calculation of the modules.

     CALL setuppointermatrix(pointer(), index())

     CALL orderofcalculation(pointer(), index())

  IF flag.esc% = 0 THEN
    IF print.connec% = 1 THEN
       CALL connecttable
       print.connec% = 0
    END IF
  END IF

END SUB

REM $STATIC
SUB orderofcalculation (pointer(), index())

REM $DYNAMIC

REDIM test(140)

     count% = 1

     FOR k% = 1 TO nomod%
        IF pointer(2, k%) = pointer(2, k% + 1) THEN
           order(1, count%) = modu(k%).modno
           order(2, count%) = k%
'          LPRINT , order(1, count%), order(2, count%)
           count% = count% + 1
        END IF
     NEXT k%

     adder% = 0

     DO UNTIL count% = nomod% + 1

        FOR i% = 1 TO nomod%
           flag1 = 0
           flag2 = 0
           IF test(i%) = 1 GOTO 10
               upsmod = pointer(2, i% + 1) - pointer(2, i%)
               start = pointer(2, i%)
               FOR k% = 0 TO upsmod - 1
                   modups = index(start + k%)
                   CALL check(modups, order(), count%, flag1, flag2)
                   IF flag1 = 1 THEN
                       EXIT FOR
                   ELSEIF flag2 = upsmod THEN
                       order(1, count%) = modu(i%).modno
                       order(2, count%) = i%
'                       LPRINT , order(1, count%), order(2, count%)
                       count% = count% + 1
                       test(i%) = 1
                   END IF
               NEXT k%
```

```
10 :    NEXT i%

        adder% = adder% + 1

        IF adder% > nomod% + 10 THEN
            BEEP
            BEEP
            PRINT
            PRINT "Connectivity incorrect. !!!  Please check "
            PRINT "Press any key"
            WHILE INKEY$ = "": WEND
            ERASE test, pointer, index
            flag.esc% = 1
            EXIT SUB
        END IF

        LOOP


 ERASE test, pointer, index

END SUB

REM $STATIC
SUB overaquistack

    FOR j% = 1 TO nomod%
        num% = 0
        IF modu(j%).typ = 1 THEN
            num% = num% + 1
            unit(j%, 0) = num%
            unit(j%, num%) = j%
            l% = j%

            DO
               num% = num% + 1
               FOR k% = 1 TO nomod%
                  IF modu(l%).infmod = modu(k%).modno THEN
                      unit(j%, 0) = num%
                      unit(j%, num%) = k%
                      l% = k%
                    '  LPRINT unit(j%, 0), unit(l%, num%)
                      EXIT FOR
                  END IF
               NEXT k%
            LOOP UNTIL modu(l%).infmod = 0

        END IF

    NEXT j%

END SUB

SUB setuppointermatrix (pointer(), index())

     position% = 1
     pointer(2, 1) = position%

     FOR j% = 1 TO nomod%
         pointer(1, j%) = modu(j%).modno
         FOR i% = 1 TO nomod%
             IF modu(i%).dsmod = modu(j%).modno THEN
                index(position%) = modu(i%).modno
                position% = position% + 1
             END IF
             IF modu(i%).ofl = modu(j%).modno THEN
```

```
                index(position%) = modu(i%).modno
                position% = position% + 1
            END IF
            IF modu(i%).modno = modu(j%).infmod THEN
                index(position%) = modu(i%).modno
                position% = position% + 1
            END IF
        NEXT i%
        pointer(2, j% + 1) = position%
    NEXT j%

'    FOR k% = 1 TO position%
'        LPRINT pointer(1, k%), pointer(2, k%), index(k%)
'    NEXT k%

END SUB

SUB upstreammods

    FOR j% = 1 TO nomod%
        count% = 0: add% = 0
        FOR k% = 1 TO nomod%
            IF modu(j%).modno = modu(k%).dsmod THEN
                count% = count% + 1
                con(j%, 0) = count%
                con(j%, count%) = k%
            END IF
            IF modu(j%).modno = modu(k%).ofl THEN
                add% = add% + 1
                ovflo(j%, 0) = add%
                ovflo(j%, add%) = k%
            END IF
        NEXT k%
    NEXT j%

END SUB

DECLARE SUB qcrit (h%, qc1!, qc2!)
DECLARE SUB outflow (h%, fla%)
DECLARE SUB newtrapdep (modnum%, i%, p%, ynew!)
DECLARE SUB volcalc (num%, volume!)
DECLARE FUNCTION getfilename$ (heading$)
DECLARE SUB clearscreen ()
DECLARE SUB fileoutput ()
DECLARE SUB modconnect (print.connec%)
TYPE modconnectivity
     modno AS INTEGER
     ofl AS INTEGER
     typ AS INTEGER
     dsmod AS INTEGER
     infmod AS INTEGER
     pl AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
    lng AS SINGLE
    wid AS SINGLE
END TYPE

TYPE aquimod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wtl AS SINGLE
```

```
    sorp AS SINGLE
    perm AS SINGLE
    por AS SINGLE
    imc AS SINGLE
    cap AS SINGLE
    volume AS SINGLE
    yprev AS SINGLE
END TYPE

TYPE pipemod
     slo AS SINGLE
     diam AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     cap AS SINGLE
     min AS SINGLE
END TYPE

TYPE trapmod
      slo AS SINGLE
      lng AS SINGLE
      man AS SINGLE
      wid AS SINGLE
      ss1 AS SINGLE
      ss2 AS SINGLE
      mdep AS SINGLE
      cap AS SINGLE
END TYPE

TYPE stormod
    cl AS SINGLE
    sl AS SINGLE
    a AS SINGLE
    b AS SINGLE
    stlev AS SINGLE
    typ AS INTEGER
    ccu AS SINGLE
    cu AS SINGLE
    ccs AS SINGLE
    cs AS SINGLE
    csp AS SINGLE
    sp AS SINGLE
    depth AS SINGLE
    crit1 AS SINGLE
    crit2 AS SINGLE
    crit3 AS SINGLE
    prevstor AS SINGLE
END TYPE

TYPE compmod
    slo AS SINGLE
    lng AS SINGLE
    nosegs AS INTEGER
    nopts AS INTEGER
END TYPE

COMMON SHARED modu() AS modconnectivity, pipe() AS pipemod, storage() AS
stormod
COMMON SHARED overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON SHARED qin(), qout(), unit(), con(), ovflo(), order(), nomod%, noit%,
tint
COMMON SHARED m, over(), rain(), pi
COMMON SHARED prompt1$, flag.file%, tcode%
COMMON SHARED title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON SHARED sim.time, rain.time, hyeto.number%, newmod%, flag.insert%
COMMON SHARED compchan() AS compmod, x(), y(), n(), segno()
```

```
errorproc:

      SELECT CASE ERR
            CASE 25, 24, 68

               BEEP: BEEP: PRINT : PRINT : PRINT "printer not connected or
switched on ... press any key"
               WHILE INKEY$ = "": WEND
               RESUME

           CASE 64, 52

               BEEP
               BEEP
               PRINT
               PRINT
               PRINT "Bad filename (limited to 8 characters) ... press any
key"
               WHILE INKEY$ = "": WEND
               RESUME begin

           CASE ELSE
               ON ERROR GOTO 0
      END SELECT

begin: CALL fileoutput


END

SUB data.echo

      ON ERROR GOTO errorproc

'     Subroutine to print a data file
      COLOR 3, 0: CLS : COLOR 0, 2
      LOCATE 10, 22: PRINT " Printing data file "
      LOCATE 10, 42: PRINT file$ + " ": COLOR 3, 0
      LPRINT CHR$(27); CHR$(69)
      LPRINT " WITSKM data file "; : LPRINT file$
      LPRINT CHR$(27); CHR$(70);
      LPRINT : LPRINT title$
      LPRINT : LPRINT "Time interval (h) ..............  :"; tint
      LPRINT "Simulation duration (h) .....  :"; sim.time
      LPRINT "Rainfall duration (h) .....  :"; rain.time
      LPRINT "Hyetograph ordinates (mm/h) :-"
      FOR k% = 1 TO hyeto.number%
         LPRINT " ";
         LPRINT USING "####.##"; rain(k%);
      NEXT k%
      LPRINT ""
      LPRINT
      LPRINT : LPRINT "Number of modules : "; : LPRINT USING "###"; nomod%
      LPRINT
      WIDTH LPRINT 255
      LPRINT CHR$(15);

      FOR j% = 1 TO 6

     count% = 0

     FOR k% = 1 TO nomod%

         IF modu(k%).typ = 1 AND j% = 1 THEN

        count% = count% + 1
```

```
        IF count% = 1 THEN
         LPRINT " Overland Flow and Aquifer Modules"
         LPRINT "--------------------------------------"
         LPRINT
                                                L   P   R   I   N   T          "
-------------------------------------------------------------------------------
-------------------------------------------------------------"
         LPRINT ": No : D/S Mod : Inf Mod : Lgt(m) : Wid(m) : Slope : Man n
: Alpha : Depth(m): Perm(mm/hr): Suc Hd(m): Por : Mois Con : W.T. Level(m):"
                                                L   P   R   I   N   T          "
-------------------------------------------------------------------------------
-------------------------------------------------------------"
        END IF

        FOR z% = 1 TO unit(k%, 0)
            p% = unit(k%, z%)
            pnt% = modu(p%).pl

            LPRINT " :";
            LPRINT USING "###"; modu(p%).modno;
            LPRINT " :    ";
            LPRINT USING "###"; modu(p%).dsmod;
            LPRINT "    :    ";
            LPRINT USING "###"; modu(p%).infmod;
            LPRINT "    : ";
            IF z% = 1 THEN
            alp = SQR(overlnd(pnt%).slo) / overlnd(pnt%).man
           LPRINT USING "####.#"; overlnd(pnt%).lng;
           LPRINT " : ";
           LPRINT USING "####.#"; overlnd(pnt%).wid;
           LPRINT " : ";
           LPRINT USING "#.###"; overlnd(pnt%).slo;
           LPRINT " : ";
           LPRINT USING "#.###"; overlnd(pnt%).man;
           LPRINT " : ";
           LPRINT USING "##.###"; alp;
           LPRINT ":  -----  :    ----    :  -----  : --- :";
           LPRINT "   -----   :      ----    :"
            ELSE
           LPRINT USING "####.#"; aqui(pnt%).lng;
           LPRINT " : ";
           LPRINT USING "####.#"; aqui(pnt%).wid;
           LPRINT " : ";
           LPRINT USING "#.###"; aqui(pnt%).slo;
           LPRINT " : -----  :";
           LPRINT " -----  : ";
           LPRINT USING "###.##"; aqui(pnt%).depth;
           LPRINT "   :    ";
           LPRINT USING "###.##"; aqui(pnt%).perm;
           LPRINT "    :  ";
           LPRINT USING "##.###"; aqui(pnt%).sorp;
           LPRINT "   : ";
           LPRINT USING "#.##"; aqui(pnt%).por;
           LPRINT ":    ";
           LPRINT USING "#.###"; aqui(pnt%).imc;
           LPRINT "   :    ";
           LPRINT USING "###.##"; aqui(pnt%).wtl;
           LPRINT "    :"
            END IF

        NEXT z%
                                                L   P   R   I   N   T
"-------------------------------------------------------------------------------
-------------------------------------------------------------"

        END IF
```

```
         IF modu(k%).typ = 2 AND j% = 2 THEN

        count% = count% + 1

        IF count% = 1 THEN
           LPRINT
           LPRINT " Pipe Modules "
           LPRINT "--------------"
           LPRINT
                                              L   P   R   I   N   T          "
-------------------------------------------------------------------------"
           LPRINT " : No : D/S Mod : Ovf Mod : Lgt(m) : Diam(m) : Slope : Man
n : Alpha :"
                                              L   P   R   I   N   T          "
-------------------------------------------------------------------------"
        END IF

        pnt% = modu(k%).pl
        alp = SQR(pipe(pnt%).slo) / pipe(pnt%).man
        LPRINT " :";
        LPRINT USING "###"; modu(k%).modno;
        LPRINT " :    ";
        LPRINT USING "###"; modu(k%).dsmod;
        LPRINT "    :    ";
        LPRINT USING "###"; modu(k%).ofl;
        LPRINT "    : ";
        LPRINT USING "####.#"; pipe(pnt%).lng;
        LPRINT " : ";
        LPRINT USING "##.###"; pipe(pnt%).diam;
        LPRINT "   : ";
        LPRINT USING "#.###"; pipe(pnt%).slo;
        LPRINT " : ";
        LPRINT USING "#.###"; pipe(pnt%).man;
        LPRINT " : ";
        LPRINT USING "##.###"; alp;
        LPRINT ":"
         END IF

         IF modu(k%).typ = 3 AND j% = 3 THEN

        count% = count% + 1

        IF count% = 1 THEN
           LPRINT " Trapezoidal Channel Modules"
           LPRINT "----------------------------"
           LPRINT
                                              L   P   R   I   N   T          "
-------------------------------------------------------------------------
-----------------------------------------"
           LPRINT " : No : D/S Mod : Ovf Mod : Lgt(m) : Base wid(m) : Slope
: LHS side slope : RHS side slope : Man n : Alpha : Depth(m) :"
                                              L   P   R   I   N   T          "
-------------------------------------------------------------------------
-----------------------------------------"
        END IF

        pnt% = modu(k%).pl
        alp = SQR(trapchan(pnt%).slo) / trapchan(pnt%).man
        LPRINT " :";
        LPRINT USING "###"; modu(k%).modno;
        LPRINT " :    ";
        LPRINT USING "###"; modu(k%).dsmod;
        LPRINT "    :    ";
        LPRINT USING "###"; modu(k%).ofl;
        LPRINT "    : ";
        LPRINT USING "####.#"; trapchan(pnt%).lng;
        LPRINT " :  ";
```

```
        LPRINT USING "#####.##"; trapchan(pnt%).wid;
        LPRINT "   : ";
        LPRINT USING "#.###"; trapchan(pnt%).slo;
        LPRINT " :    ";
        LPRINT USING "####.##"; trapchan(pnt%).ss1;
        LPRINT "     :    ";
        LPRINT USING "####.##"; trapchan(pnt%).ss2;
        LPRINT "     : ";
        LPRINT USING "#.###"; trapchan(pnt%).man;
        LPRINT " : ";
        LPRINT USING "##.###"; alp;
        LPRINT ":  ";
        LPRINT USING "###.###"; trapchan(pnt%).mdep;
        LPRINT " :"

         END IF

         IF modu(k%).typ = 4 AND j% = 4 THEN

        count% = count% + 1

        IF count% = 1 THEN
           LPRINT " Storage Modules"
           LPRINT "-----------------"
           LPRINT
                                                L   P   R   I   N   T          "
-------------------------------------------------------------------------------
 ------------------------------------------------------------"
           LPRINT " : No : DS Mod : FSL(m) : St cof : St ex : Srt Lev(m) : cof
usub : ex usub : cof sub : ex sub : cof sp : ex sp : out d(m) : lev out(m):"
                                               L   P   R   I   N   T          "
-------------------------------------------------------------------------------
-------------------------------------------------------------"
        END IF
        pnt% = modu(k%).pl
        LPRINT " : ";
        LPRINT USING "###"; modu(k%).modno;
        LPRINT ":  ";
        LPRINT USING "###"; modu(k%).dsmod;
        LPRINT "   : ";
        LPRINT USING "###.##"; storage(pnt%).sl;
        LPRINT " : ";
        LPRINT USING "####.#"; storage(pnt%).a;
        LPRINT " : ";
        LPRINT USING "#.###"; storage(pnt%).b;
        LPRINT " :  ";
        LPRINT USING "#####.##"; storage(pnt%).stlev;
        LPRINT "  : ";
        LPRINT USING "####.###"; storage(pnt%).ccu;
        LPRINT " : ";
        LPRINT USING "##.####"; storage(pnt%).cu;
        LPRINT " : ";
        LPRINT USING "####.##"; storage(pnt%).ccs;
        LPRINT " : ";
        LPRINT USING "#.####"; storage(pnt%).cs;
        LPRINT " : ";
        LPRINT USING "####.#"; storage(pnt%).csp;
        LPRINT " : ";
        LPRINT USING "#.###"; storage(pnt%).sp;
        LPRINT " : ";
        LPRINT USING "#,###.##"; storage(pnt%).depth;
        LPRINT "  :  ";
        LPRINT USING "#####.##"; storage(pnt%).cl;
        LPRINT " :"

         END IF
```

```
        IF modu(k%).typ = 5 AND j% = 5 THEN

        count% = count% + 1

        IF count% = 1 THEN
           LPRINT " Compound Channel Modules"
           LPRINT "-------------------------"
           LPRINT
                                                L   P   R   I   N   T          "
-------------------------------------------------------------------------------
-----"
           LPRINT " : No : D/S Mod : Ovf Mod : Lgt(m) : Number of Segs : Slope
: Number of Points :"
                                                L   P   R   I   N   T          "
-------------------------------------------------------------------------------
-----"
        END IF

        pnt% = modu(k%).pl
        LPRINT " :";
        LPRINT USING "###"; modu(k%).modno;
        LPRINT " :    ";
        LPRINT USING "###"; modu(k%).dsmod;
        LPRINT "    :    ";
        LPRINT USING "###"; modu(k%).ofl;
        LPRINT      : ";
        LPRINT USING "####.#"; compchan(pnt%).lng;
        LPRINT " :        ";
        LPRINT USING "###"; compchan(pnt%).nosegs;
        LPRINT "        : ";
        LPRINT USING "#.###"; compchan(pnt%).slo;
        LPRINT " :        ";
        LPRINT USING "###"; compchan(pnt%).nopts;
        LPRINT "        :"
                                                  L     P     R     I     N     T
"-------------------------------------------------------------------------------
-----"
        LPRINT
        LPRINT "       X coordinate (m)       Y coordinate (m)"
        LPRINT "       ----------------       ----------------"
        LPRINT

        FOR jj% = 1 TO compchan(pnt%).nopts
            LPRINT "           ";
            LPRINT USING "###.##"; x(pnt%, jj%);
            LPRINT "                    ";
            LPRINT USING "###.##"; y(pnt%, jj%)
        NEXT jj%

        LPRINT
        LPRINT " Segment number      left point      right point      roughness
(n) "
        LPRINT  "----------------        ----------        -----------
--------------"
        LPRINT

        FOR jj% = 1 TO compchan(pnt%).nosegs
            LPRINT "        ";
            LPRINT USING "##"; jj%;
            LPRINT "                ";
            LPRIN. USING "###"; segno(pnt%, jj%, 1);
            LPRINT "              ";
            LPRINT USING "###"; segno(pnt%, jj%, 2);
            LPRINT "                ";
            LPRINT USING "##.###"; n(pnt%, jj%)
        NEXT jj%
```

```
        END IF

    NEXT k%
    IF j% = 2 AND count% > 0 THEN
                                  L    P    R    I    N    T          "
-------------------------------------------------------------------------"
      LPRINT
    END IF

    IF j% = 3 AND count% > 0 THEN
                                  L    P    R    I    N    T          "
-------------------------------------------------------------------------
-----------------------------------------"
      LPRINT
    END IF

    IF j% = 4 AND count% > 0 THEN
                                  L    P    R    I    N    T          "
-------------------------------------------------------------------------
-------------------------------------------------------------"
      LPRINT
    END IF

     NEXT j%

END SUB

SUB fileoutput

    ON ERROR GOTO errorproc


    flag% = 0

    DO
       clearscreen

       heading$ = "Enter filename (0 to end)"

       file1$ = getfilename$(heading$)

       IF file1$ = "c:0.DAT" THEN
          EXIT SUB
       ELSE
          OPEN file1$ FOR OUTPUT AS #1
       END IF

getnum: clearscreen

       LOCATE 4, 1
       PRINT "Enter module number for data output"
       COLOR 7, 0
       LOCATE 4, 37
       INPUT "", num%
       COLOR 3, 0

       modnum% = 0

       FOR i% = 1 TO nomod%
          IF modu(i%).modno = num% THEN
               modnum% = i%
               EXIT FOR
          END IF
       NEXT i%
```

```
        IF modnum% <> 0 THEN
           FOR j% = 0 TO noit%
                PRINT #1, USING "####.##"; j% * tint * 60;
                PRINT #1, " ";
                PRINT #1, USING "####.###", qout(i%, j%)
           NEXT j%

           CLOSE #1

        ELSE

           BEEP
           BEEP
           PRINT
           PRINT
           PRINT "Module "; num%; " does not exist ... press any key"
           WHILE INKEY$ = "": WEND
           GOTO getnum

        END IF

  LOOP UNTIL flag% = 1

END SUB

SUB graphpl


getno:   SCREEN 0: WIDTH 80: CLS
         LOCATE 4, 1: COLOR 3, 0: PRINT "Which module's output hydrograph do
you want plotted (0 to return)"
         COLOR 7, 0: LOCATE 4, 68: INPUT "", num%
         FOR i% = 1 TO ncmod%
            IF modu(i%).modno = num% THEN
               modnum% = i%
               GOTO grapplo
            END IF
            IF num% = 0 THEN GOTO last
         NEXT i%
         BEEP: BEEP: PRINT : PRINT : PRINT "Module "; num%; " does not exist
... press any key"
         WHILE INKEY$ = "": WEND
         CLS 0
         GOTO getno

grapplo: CALL volcalc(modnum%, vol)


         SCREEN 2

         peak1 = 0
         peak2 = 0
         peak3 = 0
         FOR k% = 1 TO noit%
          IF qout(modnum%, k%) > peak1 THEN
              peak1 = qout(modnum%, k%)
          END IF
          IF qin(modnum%, k%) > peak2 THEN
              peak2 = qin(modnum%, k%)
          END IF
          IF rain(k%) > peak3 THEN
              peak3 = rain(k%)
          END IF
         NEXT k%

          rmax = peak3 + 5
         xmax = noit% * tint
```

```
'       VIEW (15, 15)-(600, 42), , 1
'       WINDOW (0, 0)-(xmax, rmax)
'       FOR i% = 0 TO noit% - 1
'         x1 = i% * tint
'         y1 = rain(i%)
'         x2 = (i% + 1) * tint
'         y2 = rain(i% + 1)
'         LINE (x1, y1)-(x2, y2)
'       NEXT i%

      IF modu(modnum%).typ <> 6 THEN
        LOCATE 5, 55: PRINT " peak inflow= ";
        PRINT USING "####.###"; peak2
        LOCATE 6, 55: PRINT "peak outflow= ";
        PRINT USING "####.###"; peak1
        LOCATE 7, 55: PRINT "Volume";
        PRINT USING "######.##"; vol
      ELSE
        LOCATE 5, 55: PRINT " peak inflow= ";
        PRINT USING "####.###"; peak2 * 1000! * 3600!
        LOCATE 6, 55: PRINT "peak outflow= ";
        PRINT USING "####.###"; peak1 * 1000! * 3600!
        LOCATE 7, 55: PRINT "Volume";
        PRINT USING "######.##"; vol
      END IF

        VIEW (32, 3)-(620, 165), , 1

        IF peak1 > peak2 THEN
         ymax = peak1 + peak1 / 10
        ELSE
         ymax = peak2 + peak2 / 10
        END IF

        IF ymax = 0 THEN
           ymax = .001
        END IF
        IF modu(modnum%).typ <> 6 THEN
           LOCATE 12, 2: PRINT "cumec"
           LOCATE 1, 1: PRINT USING "####.###"; ymax
        ELSE
           ymax = ymax * 1000 * 3600!
           LOCATE 12, 2: PRINT "l/hr"
           LOCATE 1, 1: PRINT USING "####.###"; ymax
        END IF
        LOCATE 22, 2: PRINT "0"
        LOCATE 23, 4: PRINT "0"
        LOCATE 23, 39: PRINT "Time mins"
        LOCATE 23, 73: PRINT USING "###.##"; sim.time * 60



        WINDOW (0, 0)-(xmax, ymax)
        FOR i% = 0 TO noit% - 1
          x1 = i% * tint
          x2 = (i% + 1) * tint
          y1 = qout(modnum%, i%)
          y3 = qin(modnum%, i%)
          y2 = qout(modnum%, i% + 1)
          y4 = qin(modnum%, i% + 1)

          IF modu(modnum%).typ <> 6 THEN
              LINE (x1, y1)-(x2, y2)
              LINE (x1, y3)-(x2, y4)
          ELSE
              LINE (x1, y1 * 1000 * 3600!)-(x2, y2 * 1000! * 3600!)
              LINE (x1, y3 * 1000 * 3600!)-(x2, y4 * 1000 * 3600!)
```

```
          END IF

        NEXT i%
waiting:        WHILE INKEY$ = "": WEND
                GOTO getno
last:

END SUB

SUB newtrapdep (modnum%, i%, p%, ynew)

     cl = storage(p%).cl: sl = storage(p%).sl
     d = storage(p%).depth
     csp = storage(p%).csp: sp = storage(p%).sp
     ccs = storage(p%).ccs: cs = storage(p%).cs
     yold = sl + .1

     DO
       abit1 = ccs * cs * (yold - cl - .5 * storage(pnt%).depth) ^ (cs - 1)
       abit2 = csp * sp * (yold - sl) ^ (sp - 1)
       dfdy = -abit1 - abit2
       f = ccs * (yold - cl - .5 * d) ^ cs
       f = f + csp * (yold - sl) ^ sp
       f = qout(modnum%, i%) - f
       ynew = yold - f / dfdy
       diff = ABS(1 - ynew / yold)
       yold = ynew
     LOOP UNTIL diff <= .001

END SUB

SUB outflow (h%, fla%)

     FOR k% = 1 TO noit%
         IF qout(h%, k%) <> 0 THEN
             fla% = 1
             EXIT SUB
         END IF
    NEXT k%

END SUB

SUB printout

         ON ERROR GOTO errorproc

         WIDTH LPRINT 80
         LPRINT CHR$(18);
         SCREEN 0: WIDTH 80: CLS

         LPRINT CHR$(27); CHR$(69)
         LPRINT " WITSKM data file "; : LPRINT file$
         LPRINT CHR$(27); CHR$(70);
         LPRINT : LPRINT title$
         LPRINT : LPRINT "Time interval (h) ..... :"; tint
         LPRINT "Simulation duration (h) ..... :"; sim.time
         LPRINT "Rainfall duration (h) ..... :"; rain.time
         LPRINT "Hyetograph ordinates (mm/h) :-"
         FOR k% = 1 TO hyeto.number%
           LPRINT " ";
           LPRINT USING "####.##"; rain(k%);
         NEXT k%
         LPRINT ""
         LPRINT


getno1:  CLS
```

```
        LOCATE 4, 1
        COLOR 3, 0
        PRINT "Enter module number for output (0 to return)"
        COLOR 7, 0: LOCATE 4, 50: INPUT "", num%
        FOR i% = 1 TO nomod%
           IF modu(i%).modno = num% THEN
              modnum% = i%
              GOTO skip1
           END IF
           IF num% = 0 THEN GOTO fini
        NEXT i%
        BEEP: BEEP: PRINT : PRINT : PRINT "Module "; num%; " does not exist
... press any key"
        WHILE INKEY$ = "": WEND
        GOTO getno1



skip1: CALL volcalc(modnum%, vol)


      IF modu(modnum%).typ = 2 OR modu(modnum%).typ = 3 THEN
        LPRINT " module number = "; modu(modnum%).modno;
        IF modu(modnum%).typ = 2 THEN
           LPRINT "capacity (m^3/s):";
           LPRINT USING "####.###"; pipe(modu(modnum%).p1).cap
           LPRINT "Volume (m^3):";
           LPRINT USING "######.##"; vol
        ELSE
           LPRINT "capacity (m^3/s):";
           LPRINT USING "####.##"; trapchan(modu(modnum%).p1).cap
           LPRINT "Volume (m^3):";
           LPRINT USING "######.##"; vol
        END IF
    ELSEIF modu(modnum%).typ = 1 THEN
        LPRINT " module number = "; modu(modnum%).modno
        LPRINT "Volume Runoff (m^3):";
        LPRINT USING "######.##"; vol
        po% = modu(modnum%).p1
        po1% = unit(modnum%, 2)
        LPRINT "Rainfall Volume=";

        rainvol = 0
        FOR h% = 1 TO noit%
           rainvol = rainvol + rain(h%) * tint * overlnd(po%).lng *
overlnd(po%).wid / 1000!
        NEXT h%
        LPRINT rainvol
    ELSE
        LPRINT " module number = "; modu(modnum%).modno
        LPRINT "Volume (m^3):";
        LPRINT USING "######.##"; vol
    END IF

    LPRINT

    LPRINT "   ";
    LPRINT "  Time";
    LPRINT "      Inflow ";
    LPRINT "     utflow";
    IF modu(modnum%).typ <> 4 THEN
       LPRINT "     overflow "
    ELSE
       LPRINT "    depth "
    END IF
```

```
      LPRINT "    ";
      LPRINT "   hr";
      IF modu(modnum%).typ <> 6 THEN
         LPRINT "         cumec";
         LPRINT "       cumec ";
         IF modu(modnum%).typ <> 4 THEN
            LPRINT "       cumec"
         ELSE
            LPRINT "       m "
         END IF
      ELSE
         LPRINT "          l/hr";
         LPRINT "        l/hr ";
         LPRINT "        l/hr"
      END IF

      LPRINT

      FOR i% = 0 TO noit%
       IF modu(modnum%).typ < 4 OR modu(modnum%).typ = 5 THEN
         LPRINT "    ";
         LPRINT USING "####.##"; i% * tint;
         LPRINT "    ";
         LPRINT USING "###.###"; qin(modnum%, i%);
         LPRINT "    ";
         LPRINT USING "###.###"; qout(modnum%, i%);
         LPRINT "    ";
         LPRINT USING "###.###"; over(modnum%, i%)

       ELSEIF modu(modnum%).typ = 6 THEN
         LPRINT "    ";
         LPRINT USING "####.##"; i% * tint;
         LPRINT "    ";
         LPRINT USING "####.####"; qin(modnum%, i%) * 3600000!;
         LPRINT "    ";
         LPRINT USING "####.####"; qout(modnum%, i%) * 3600000!;
         LPRINT "    ";
         LPRINT USING "####.####"; over(modnum%, i%) * 3600000!

       ELSE

         CALL outflow(modnum%, fla%)

         p% = modu(modnum%).pl
         IF fla% = 0 THEN
             LPRINT "no outflow from dam"
             LPRINT
             LPRINT "final level in dam="; (storage(p%).prevstor /
storage(p%).a) ^ (1 / storage(p%).h);
             LPRINT " m"
             GOTO getno1
         ELSE
            IF storage(p%).typ = 1 THEN
               CALL qcrit(modnum%, qc1, qc2)
               IF qout(modnum%, i%) = 0 THEN
                   dep = storage(p%).cl
               ELSEIF qout(modnum%, i%) > 0 AND qout(modnum%, i%) <= qc1 THEN
                   dep = (qout(modnum%, i%) / storage(p%).ccu) ^ (1 /
storage(p%).cu) + storage(p%).cl
               ELSEIF qout(modnum%, i%) > qc1 AND qout(modnum%, i%) <= qc2 THEN
                   dep = (qout(modnum%, i%) / storage(p%).ccs) ^ (1 /
storage(p%).cs)
                   dep = dep + storage(p%).cl + .5 * storage(p%).depth
               ELSE
                   CALL newtrapdep(modnum%, i%, p%, dep)
               END IF
            ELSE
```

```
            dep = (qout(modnum%, i%) / storage(p%).csp) ^ (1 /
storage(p%).sp)
            dep = dep + storage(p%).sl
         END IF
      END IF
      IF fla% <> 0 THEN
         LPRINT "  ";
         LPRINT USING "####.##"; i% * tint;
         LPRINT "  ";
         LPRINT USING "#####.###"; qin(modnum%, i%);
         LPRINT "   ";
         LPRINT USING "#####.###"; qout(modnum%, i%);
         LPRINT "  ";
         LPRINT USING "####.###"; dep
      END IF
    END IF
   NEXT i%
   LPRINT
   GOTO getno1
fini:
END SUB

SUB qcrit (h%, qc1, qc2)
   p% = modu(h%).p1
   qc1 = storage(p%).ccu * (1.5 * storage(p%).depth) ^ storage(p%).cu
   qc2 = storage(p%).ccs * (storage(p%).sl - storage(p%).cl - .5 *
storage(p%).depth) ^ storage(p%).cs
END SUB

SUB volcalc (num%, volume)

   volume = 0

   FOR j% = 1 TO noit%
      element = (qout(num%, j% - 1) + qout(num%, j%)) * tint * 3600 / 2
      volume = volume + element
   NEXT j%

END SUB
```

COMPUTATION SUBPROGRAM

```
'Control program for the computation subprogram

DECLARE SUB QvsA (l%, q!(), a!(), b!(), nohts!)
DECLARE SUB compoundcha (l%, i%, q(), a(), b(), nohts)
DECLARE SUB storagemod (l%, i%)
DECLARE SUB trapmodule (l%, i%)
DECLARE SUB pipemodule (l%, i%)
DECLARE SUB graphp1 ()
DECLARE SUB printout ()
DECLARE SUB checkaquisat (pnt2%, pnt3%, i%, infilout!)
DECLARE SUB aquiroute1 (infilin, infilout, pnt%, pnt1%, i%)
DECLARE SUB overroute1 (infilout, pnt%, pnt1%, i%)
DECLARE SUB potinfiltration (pnt2%, pnt3%, i%, infilt)
DECLARE SUB overland1 (l%, i%)
DECLARE SUB initialize ()
DECLARE SUB modconnect (print.connec%)
DECLARE SUB datainput ()

TYPE modconnectivity
     modno AS INTEGER
     ofl AS INTEGER
     typ AS INTEGER
     dsmod AS INTEGER
     infmod AS INTEGER
     p1 AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
    lng AS SINGLE
    wid AS SINGLE
END TYPE

TYPE aquimod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wtl AS SINGLE
   sorp AS SINGLE
   perm AS SINGLE
   por AS SINGLE
   imc AS SINGLE
   cap AS SINGLE
   volume AS SINGLE
   yprev AS SINGLE
END TYPE

TYPE pipemod
    slo AS SINGLE
    diam AS SINGLE
    lng AS SINGLE
    man AS SINGLE
    cap AS SINGLE
    min AS SINGLE
END TYPE

TYPE trapmod
     slo AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     wid AS SINGLE
     ss1 AS SINGLE
     ss2 AS SINGLE
     mdep AS SINGLE
```

```
     cap AS SINGLE
END TYPE

TYPE stormod
    cl AS SINGLE
    sl AS SINGLE
    a AS SINGLE
    b AS SINGLE
    stlev AS SINGLE
    typ AS INTEGER
    ccu AS SINGLE
    cu AS SINGLE
    ccs AS SINGLE
    cs AS SINGLE
    csp AS SINGLE
    sp AS SINGLE
    depth AS SINGLE
    crit1 AS SINGLE
    crit2 AS SINGLE
    crit3 AS SINGLE
    prevstor AS SINGLE
END TYPE

TYPE compmod
    slo AS SINGLE
    ing AS SINGLE
    nosegs AS INTEGER
    nopts AS INTEGER
END TYPE

COMMON SHARED modu() AS modconnectivity, pipe() AS pipemod, storage() AS
stormod
COMMON SHARED overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON SHARED qin(), qout(), unit(), con(), ovflo(), order(), nomod%, noit%,
tint
COMMON SHARED m, over(), rain(), pi
COMMON SHARED prompt1$, flag.file%, tcode%
COMMON SHARED title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON SHARED sim.time, rain.time, hyeto.number%, newmod%, flag.insert%
COMMON SHARED compchan() AS compmod, x(), y(), n(), segno()

  REDIM b(20), q(20), a(20)

  CALL modconnect(0)

  CALL initialize

  CLS

  FOR j% = 1 TO nomod%

    l% = order(2, j%)

    IF modu(l%).typ <> 6 THEN

      IF modu(l%).typ = 5 THEN
           CALL QvsA(l%, q(), a(), b(), nohts)
      END IF

      FOR i% = 1 TO noit%
         COLOR 7, 0
         CLS
         COLOR 0, 2
         LOCATE 12, 20: PRINT SPACE$(29)
         LOCATE 12, 20
         nommer% = modu(l%).modno
         PRINT "time step no"; i%; " Module no "; nommer%
```

```
        COLOR 3, 0

                SELECT CASE modu(l%).typ
                    CASE 1
                        CALL overland1(l%, i%)
                    CASE 2
                        CALL pipemodule(l%, i%)
                    CASE 3
                        CALL trapmodule(l%, i%)
                    CASE 4
                        CALL storagemod(l%, i%)
                    CASE 5

                        CALL compoundcha(l%, i%, q(), a(), b(), nohts)
                END SELECT

      NEXT i%

    END IF

NEXT j%

tcode% = 2

CLS

COLOR 0, 2
LOCATE 11, 28
PRINT "Chaining editor subprogram"
COLOR 3, 0
CHAIN "editor"


END

'Connectivity calculation module
DECLARE SUB connecttable ()
DECLARE SUB setuppointermatrix (pointer(), index())
DECLARE SUB orderofcalculation (pointer!(), index!())
DECLARE SUB check (modnum!, order!(), count%, flag1!, flag2!)
DECLARE SUB overaquistack ()
DECLARE SUB upstreammods ()
DECLARE FUNCTION areapipe! (angle, pnt%)
DECLARE FUNCTION storvol! (pnt%, y!)

TYPE modconnectivity
     modno AS INTEGER
     of1 AS INTEGER
     typ AS INTEGER
     dsmod AS INTEGER
     infmod AS INTEGER
     p1 AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
    lng AS SINGLE
    wid AS SINGLE
END TYPE

TYPE aquimod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wt1 AS SINGLE
```

```
    sorp AS SINGLE
    perm AS SINGLE
    por AS SINGLE
    imc AS SINGLE
    cap AS SINGLE
    volume AS SINGLE
    yprev AS SINGLE
END TYPE

TYPE pipemod
     slo AS SINGLE
     diam AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     cap AS SINGLE
     min AS SINGLE
END TYPE

TYPE trapmod
      slo AS SINGLE
      lng AS SINGLE
      man AS SINGLE
      wid AS SINGLE
      ss1 AS SINGLE
      ss2 AS SINGLE
      mdep AS SINGLE
      cap AS SINGLE
END TYPE

TYPE stormod
     cl AS SINGLE
     sl AS SINGLE
     a AS SINGLE
     b AS SINGLE
     stlev AS SINGLE
     typ AS INTEGER
     ccu AS SINGLE
     cu AS SINGLE
     ccs AS SINGLE
     cs AS SINGLE
     csp AS SINGLE
     sp AS SINGLE
     depth AS SINGLE
     crit1 AS SINGLE
     crit2 AS SINGLE
     crit3 AS SINGLE
     prevstor AS SINGLE
END TYPE

TYPE compmod
     slo AS SINGLE
     lng AS SINGLE
     nosegs AS INTEGER
     nopts AS INTEGER
END TYPE

COMMON SHARED modu() AS modconnectivity, pipe() AS pipemod, storage() AS
stormod
COMMON SHARED overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON SHARED qin(), qout(), unit(), con(), ovflo(), order(), nomod%, noit%,
tint
COMMON SHARED m, over(), rain(), pi
COMMON SHARED prompt1$, flag.file%, tcode%
COMMON SHARED title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON SHARED sim.time, rain.time, hyeto.number%, newmod%, flag.insert%
COMMON SHARED compchan() AS compmod, x(), y(), n(), segno()
```

```
errorproc1:

     SELECT CASE ERR
            CASE 25, 24, 68

               BEEP: BEEP: PRINT : PRINT : PRINT "printer not connected or
switched on ... press any key"
               WHILE INKEY$ = "": WEND
               RESUME
           CASE ELSE
               ON ERROR GOTO 0
      END SELECT


END

SUB check (modnum, order(), count%, flag1, flag2)

    mark% = 0
    FOR j% = 1 TO count% - 1
       IF modnum = order(1, j%) THEN
          flag2 = flag2 + 1
          mark% = 1
          EXIT FOR
       END IF
    NEXT j%
    IF mark% = 0 THEN
       flag1 = 1
    END IF

END SUB

SUB connecttable

      ON ERROR GOTO errorproc1

      WIDTH LPRINT 80
      LPRINT CHR$(18);
                              l   i   n   e   1   $          =          "
-------------------------------------------------------------------------"
      title1$ = "   : no : mod no :     upstream module nos
                :"
      LPRINT line1$
      LPRINT title1$

      FOR k% = 1 TO nomod%
         LPRINT line1$
         LPRINT "   : ";
         LPRINT USING "##"; k%;
         LPRINT " :   ";
         LPRINT USING "###"; modu(k%).modno;
         LPRINT "    : ";
         FOR j% = 1 TO 10
             IF j% <= con(k%, 0) THEN
                LPRINT USING "###"; modu(con(k%, j%)).modno;
                LPRINT " : ";
             ELSE
                LPRINT "---- : ";
             END IF
         NEXT j%
      NEXT k%
      LPRINT line1$
      LPRINT " modules 900 and up are dummy aquifers"
      LPRINT

      LPRINT
```

```
        line1$ = "  ----------------------------------------------"
        title1$ = "   : no : mod no :      overflow module nos      :"
        LPRINT line1$
        LPRINT title1$

        FOR k% = 1 TO nomod%

           LPRINT line1$
           LPRINT "   : ";
           LPRINT USING "##"; k%;
           LPRINT " :   ";
           LPRINT USING "###"; modu(k%).modno;
           LPRINT "    : ";

           FOR j% = 1 TO 5
              IF j% <= ovflo(k%, 0) THEN
                 LPRINT USING "###"; modu(ovflo(k%, j%)).modno;
                 LPRINT " : ";
              ELSE
                 LPRINT "--- : ";
              END IF
           NEXT j%

        NEXT k%
        LPRINT line1$
        LPRINT " modules 900 and up are dummy aquifers"
        LPRINT

END SUB

SUB initialize

'     INPUT "do you want to enter hydrograph for a module(y/n)"; ans$
     ans$ = "n"
     IF ans$ = "y" THEN
       INPUT "module no"; modnum%
       FOR j% = 1 TO nomod%
              IF modu(j%).modno = modnum% THEN
                   num% = j%
                   EXIT FOR
              END IF
       NEXT j%
       FOR j% = 0 TO noit%
            INPUT "flow"; qin(num%, j%)
       NEXT j%
     END IF

     FOR i% = 1 TO nomod%

         SELECT CASE modu(i%).typ

              CASE 1
                 IF i% <> num% THEN
                    FOR j% = 0 TO noit%
                      qin(i%, j%) = .0001
                      qout(i%, j%) = .0001
                    NEXT j%
                 END IF

              CASE 2

                  pnt% = modu(i%).p1
                  alp = SQR(pipe(pnt%).slo) / pipe(pnt%).man
                  theta = .827 * 2 * pi
                  area = areapipe(theta, pnt%)
                  perim = pipe(pnt%).diam * theta / 2
                  r = area / perim
```

```
    pipe(pnt%).cap = alp * area * r ^ (m - 1)
    theta = .02 * 2 * pi
    area = ar apipe(theta, pnt%)
    perim = pipe(pnt%).diam * theta / 2
    pipe(pnt%).min = alp * area * r ^ (m - 1)

    IF i% <> num% THEN
      FOR z% = 0 TO noit%
       qin(i%, z%) = pipe(pnt%).min
      NEXT z%
    END IF

    FOR z% = 0 TO noit%
      qout(i%, z%) = pipe(pnt%).min
      over(i%, z%) = 0!
    NEXT z%

CASE 3
    IF i% <> num% THEN
      FOR z% = 0 TO noit%
       qin(i%, z%) = .00001
      NEXT z%
    END IF

    FOR z% = 0 TO noit%
      qout(i%, z%) = .00001
      over(i%, z%) = 0!
    NEXT z%
    pnt% = modu(i%).p1
    ss = trapchan(pnt%).ss1 + trapchan(pnt%).ss2
    fact = SQR(1 + trapchan(pnt%).ss1 ^ 2)
    fact = fact + SQR(1 + trapchan(pnt%).ss2 ^ 2)
    area = trapchan(pnt%).mdep * trapchan(pnt%).wid
    area = area + ss * trapchan(pnt%).mdep ^ 2 / 2
    perim = trapchan(pnt%).wid + trapchan(pnt%).mdep * fact
    r = area / perim
    trapchan(pnt%).cap = SQR(trapchan(pnt%).slo) * area
    trapchan(pnt%).cap = cap * r ^ (m - 1) / trapchan(pnt%).man

CASE 4
    pnt% = modu(i%).p1

    FOR z% = 0 TO noit%
       qin(i%, z%) = 0!
    NEXT z%

    FOR z% = 0 TO noit%
       qout(i%, z%) = 0!
    NEXT z%

    IF storage(pnt%).typ = 0 THEN
         storage(pnt%).crit1 = storvol(pnt%, storage(pnt%).sl)
    ELSE
         storage(pnt%).crit1 = storvol(pnt%, storage(pnt%).cl)
         ht = 1.5 * storage(pnt%).depth
         qdt = storage(pnt%).ccu * ((ht) ^ storage(pnt%).cu)
         qdt = qdt * tint * 3600
         storage(pnt%).crit2 = qdt + storvol(pnt%, ht)
                               + storage(pnt%).cl

         ht = storage(pnt%).sl - storage(pnt%).cl - .5 *
              storage(pnt%).depth
         qdt = storage(pnt%).ccs * ((ht) ^ storage(pnt%).cs) *
               tint * 3600
         storage(pnt%).crit3 = qdt + storvol(pnt%,
                               storage(pnt%).sl)
    END IF
```

```
                storage(pnt%).prevstor = storvol(pnt%,
                                         storage(pnt%).stlev)

            CASE 5

                FOR z% = 0 TO noit%
                  qin(i%, z%) = 0!
                  qout(i%, z%) = 0!
                  over(i%, z%) = 0!
                NEXT z%


            CASE 6

                pnt% = modu(i%).pl
                aqui(pnt%).yprev = aqui(pnt%).wtl
                qout(i%, 0) = aqui(pnt%).wtl * aqui(pnt%).perm *
                              aqui(pnt%).wid * aqui(pnt%).slo / (1000! *
                              3600!)
                aqui(pnt%).cap = aqui(pnt%).slo * aqui(pnt%).perm *
                                 aqui(pnt%).depth * aqui(pnt%).wid / (1000!
                                 * 3600!)
                IF i% <> num% THEN
                  FOR j% = 0 TO noit%
                     qin(i%, j%) = 0!
                     aqui(pnt%).volume = 0
                  NEXT j%
                  FOR j% = 1 TO noit%
                     qout(i%, j%) = 0
                     over(i%, j%) = 0!
                  NEXT j%
                END IF

        END SELECT

    NEXT i%

END SUB

SUB modconnect (print.connec%)

    REM $DYNAMIC

    CLS

    COLOR 0, 2
    LOCATE 11, 28
    PRINT "Calculating connectivity"
    COLOR 3, 0

    REDIM pointer(2, 141), index(170)

'   INPUT "do you want to output the calculation order ans(y\n)"; ans$

'   Call sub upstreammods to determine the upstream modules and the
'   modules which flow onto each module.
'   The number of upstream modules and their numbers are stored in array
'   CON and the overflow modules in array OVFLO

    CALL upstreammods

'   Call overaquistack to determine the overland and associated aquifers
'   The array UNIT is used to store the information.

    CALL overaquistack

'   Call subs setuppointermatrix and orderofcalculation to determine the
```

```
'   upstream - downstream order of calculation of the modules.

    CALL setuppointermatrix(pointer(), index())

    CALL orderofcalculation(pointer(), index())

  IF flag.esc% = 0 THEN
    IF print.connec% = 1 THEN
       CALL connecttable
       print.connec% = 0
    END IF
  END IF

END SUB

REM $STATIC
SUB orderofcalculation (pointer(), index())

REM $DYNAMIC

DIM test(140)

    count% = 1

    FOR k% = 1 TO nomod%
       IF pointer(2, k%) = pointer(2, k% + 1) THEN
          order(1, count%) = modu(k%).modno
          order(2, count%) = k%
'         LPRINT , order(1, count%), order(2, count%)
          count% = count% + 1
       END IF
    NEXT k%

    adder% = 0

    DO UNTIL count% = nomod% + 1

       FOR i% = 1 TO nomod%
          flag1 = 0
          flag2 = 0
          IF test(i%) = 1 GOTO 10
              upsmod = pointer(2, i% + 1) - pointer(2, i%)
              start = pointer(2, i%)
              FOR k% = 0 TO upsmod - 1
                  modups = index(start + k%)
                  CALL check(modups, order(), count%, flag1, flag2)
                  IF flag1 = 1 THEN
                     EXIT FOR
                  ELSEIF flag2 = upsmod THEN
                      order(1, count%) = modu(i%).modno
                      order(2, count%) = i%
'                      LPRINT , order(1, count%), order(2, count%)
                      count% = count% + 1
                      test(i%) = 1
                  END IF
              NEXT k%
10 :    NEXT i%

       adder% = adder% + 1

       IF adder% > nomod% + 10 THEN
           BEEP
           BEEP
           PRINT
           PRINT "Connectivity incorrect. !!!  Please check "
           PRINT "Press any key"
           WHILE INKEY$ = "": WEND
```

```
            ERASE test, pointer, index
            flag.esc% = 1
            EXIT SUB
         END IF

         LOOP


 ERASE test, pointer, index

END SUB

REM $STATIC
SUB overaquistack

     FOR j% = 1 TO nomod%
          num% = 0
          IF modu(j%).typ = 1 THEN
              num% = num% + 1
              unit(j%, 0) = num%
              unit(j%, num%) = j%
              l% = j%

              DO
                 num% = num% + 1
                 FOR k% = 1 TO nomod%
                    IF modu(l%).infmod = modu(k%).modno THEN
                        unit(j%, 0) = num%
                        unit(j%, num%) = k%
                        l% = k%
                      ' LPRINT unit(j%, 0), unit(l%, num%)
                        EXIT FOR
                    END IF
                 NEXT k%
              LOOP UNTIL modu(l%).infmod = 0

          END IF

     NEXT j%

END SUB

SUB setuppointermatrix (pointer(), index())

      position% = 1
      pointer(2, 1) = position%

      FOR j% = 1 TO nomod%
          pointer(1, j%) = modu(j%).modno
          FOR i% = 1 TO nomod%
              IF modu(i%).dsmod = modu(j%).modno THEN
                 index(position%) = modu(i%).modno
                 position% = position% + 1
              END IF
              IF modu(i%).ofl = modu(j%).modno THEN
                 index(position%) = modu(i%).modno
                 position% = position% + 1
              END IF
              IF modu(i%).modno = modu(j%).infmod THEN
                 index(position%) = modu(i%).modno
                 position% = position% + 1
              END IF
          NEXT i%
          pointer(2, j% + 1) = position%
      NEXT j%
```

```
'    FOR k% = 1 TO position%
'        LPRINT pointer(1, k%), pointer(2, k%), index(k%)
'    NEXT k%

END SUB

SUB upstreammods

    FOR j% = 1 TO nomod%
        count% = 0: add% = 0
        FOR k% = 1 TO nomod%
            IF modu(j%).modno = modu(k%).dsmod THEN
                count% = count% + 1
                con(j%, 0) = count%
                con(j%, count%) = k%
            END IF
            IF modu(j%).modno = modu(k%).ofl THEN
                add% = add% + 1
                ovflo(j%, 0) = add%
                ovflo(j%, add%) = k%
            END IF
        NEXT k%
    NEXT j%

END SUB

'Subroutine for routing flow overland and through aquifers
'The infiltration is calculated using the subroutine potinfiltration
'

DECLARE SUB muscungcoeff (length, c!, k!, theta!, cof1!, cof2!, cof3!, cof4!)
DECLARE SUB aquiroute1 (infilin!, infout!, pnt%, pnt1%, i%)
DECLARE SUB overroute1 (infout!, pnt%, pnt1%, i%)
DECLARE SUB depthforinfil.aqui (infilin!, pnt%, pnt1%, i%, availdep!)
DECLARE SUB checkaquisat (pnt2%, pnt3%, i%, infout!)
DECLARE SUB potinfiltration (pnt2%, pnt3%, i%, infilt!)
DECLARE SUB depthforinfil.plane (pnt%, pnt1%, i%, availdep!)

TYPE modconnectivity
     modno AS INTEGER
     ofl AS INTEGER
     typ AS INTEGER
     dsmod AS INTEGER
     infmod AS INTEGER
     p1 AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
    lng AS SINGLE
    wid AS SINGLE
END TYPE

TYPE aquimod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wtl AS SINGLE
   sorp AS SINGLE
   perm AS SINGLE
   por AS SINGLE
   lmc AS SINGLE
   cap AS SINGLE
   volume AS SINGLE
   yprev AS SINGLE
```

```
END TYPE

TYPE pipemod
    slo AS SINGLE
    diam AS SINGLE
    lng AS SINGLE
    man AS SINGLE
    cap AS SINGLE
    min AS SINGLE
END TYPE

TYPE trapmod
     slo AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     wid AS SINGLE
     ss1 AS SINGLE
     ss2 AS SINGLE
     mdep AS SINGLE
     cap AS SINGLE
END TYPE

TYPE stormod
    cl AS SINGLE
    sl AS SINGLE
    a AS SINGLE
    b AS SINGLE
    stlev AS SINGLE
    typ AS INTEGER
    ccu AS SINGLE
    cu AS SINGLE
    ccs AS SINGLE
    cs AS SINGLE
    csp AS SINGLE
    sp AS SINGLE
    depth AS SINGLE
    crit1 AS SINGLE
    crit2 AS SINGLE
    crit3 AS SINGLE
    prevstor AS SINGLE
END TYPE

TYPE compmod
    slo AS SINGLE
    lng AS SINGLE
    nosegs AS INTEGER
    nopts AS INTEGER
END TYPE

COMMON SHARED modu() AS modconnectivity, pipe() AS pipemod, storage() AS
stormod
COMMON SHARED overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON SHARED qin(), qout(), unit(), con(), ovflo(), order(), nomod%, noit%,
tint
COMMON SHARED m, over(), rain(), pi
COMMON SHARED prompt1$, flag.file%, tcode%
COMMON SHARED title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON SHARED sim.time, rain.time, hyeto.number%, newmod%, flag.insert%
COMMON SHARED compchan() AS compmod, x(), y(), n(), segno()



END

  B aquiroute1 (infilin, infout, pnt%, pnt1%, i%) -
      LPRINT "aquifer"; modu(pnt%).modno
      LPRINT "infilin"; infilin; "infilout"; infilout
```

```
      effecstor = aqui(pnt1%).por - aqui(pnt1%).imc
      denom = aqui(pnt1%).lng * effecstor * aqui(pnt1%).wid
      abit = (infilin - infout) * tint / (effecstor * 1000!)
      abita = (qout(pnt%, i% - 1) - qin(pnt%, i% - 1)) * tint * 3600 / denom
      y = aqui(pnt1%).yprev + abit - abita
      qout(pnt%, i%) = y * aqui(pnt1%).slo * aqui(pnt1%).wid * aqui(pnt1%).perm
/ (1000! * 3600!)
      aqui(pnt1%).yprev = y
'     LPRINT "y="; y(pnt%, i%), "q="; qout(pnt%, i%)
'     LPRINT

END SUB

SUB checkaquisat (pnt2%, pnt3%, i%, infout)

      effecstor = aqui(pnt3%).por - aqui(pnt3%).imc

      aqui(pnt3%).volume = aqui(pnt3%).volume + infout * tint / 1000!

'     LPRINT " volume="; volume(pnt2%, i%)

      IF   aqui(pnt3%).volume   /   effecstor   >=   aqui(pnt3%).depth   -
aqui(pnt3%).yprev THEN
            aqui(pnt3%).volume = aqui(pnt3%).depth * effecstor
'           qout(pnt2%, i% - 1) = aqui(pnt3%).cap
            aqui(pnt3%).yprev = aqui(pnt3%).depth
      END IF


END SUB

SUB depthforinfil.aqui (infilin, pnt%, pnt1%, i%, availdep)

      availdep = 0

      IF qout(pnt%, i% - 1) > 0 THEN
         den = aqui(pnt1%).wid * aqui(pnt1%).perm * aqui(pnt1%).slo / (1000!
* 3600!)
         ybeg = qin(pnt%, i% - 1) / den
         yend = qout(pnt%, i% - 1) / den
      '   PRINT "ybeg="; ybeg; "yend="; yend
       '  WHILE INKEY$ = "": WEND
         availdep = (yend + ybeg) / 2
      END IF

      area = aqui(pnt1%).wid * aqui(pnt1%).lng
      availdep = availdep + infilin * tint / 1000
      availdep = availdep + (qin(pnt%, i% - 1) + qin(pnt%, i%)) * tint * 3600
/ (2 * area)
      availdep = availdep * aqui(pnt1%).por

END SUB

SUB depthforinfil.plane (pnt%, pnt1%, i%, availdep)

      availdep = 0
      alp = SQR(overlnd(pnt1%).slo) / overlnd(pnt1%).man
      area = overlnd(pnt1%).wid * overlnd(pnt1%).lng

      IF qout(pnt%, i% - 1) > .0001 THEN
         den = overlnd(pnt1%).wid * alp
         ybeg = (qin(pnt%, i% - 1) / den) ^ (1 / m)
         yend = (qout(pnt%, i% - 1) / den) ^ (1 / m)
         availdep = (ybeg + yend) / 2
      END IF

      availdep = availdep + rain(i%) * tint / 1000!
```

```
    IF (qin(pnt%, i%) + qin(pnt%, i% - 1)) / 2 > .0001 THEN
        availdep = availdep + (qin(pnt%, i% - 1) + qin(pnt%, i%)) * tint *
3600 / (2 * area)
    END IF

    ' LPRINT "availdep"; availdep

END SUB

SUB overland1 (l%, i%)

DIM availdepth(unit(l%, 0)), infilout(unit(l%, 0))

    FOR k% = unit(l%, 0) TO 1 STEP -1
        pnt% = unit(l%, k%)

        FOR f% = 1 TO con(l%, 0)
            qin(pnt%, i%) = qin(pnt%, i%) + qout(con(pnt%, f%), i%)
        NEXT f%

        FOR n% = 1 TO ovflo(l%, 0)
            qin(pnt%, i%) = qin(pnt%, i%) + over(ovflo(    , n%), i%)
        NEXT n%

        IF k% <> 1 THEN
           p% = modu(pnt%).p1
           IF qin(pnt%, i%) > aqui(p%).cap THEN
             pnt1% = unit(l%, k% - 1)
             qin(pnt1%, i%) = qin(pnt1%, i%) + qin(pnt%, i%) - aqui(p%).cap
             qin(pnt%, i%) = aqui(p%).cap
           END IF

        END IF

        IF qin(pnt%, i%) < 0! THEN
           qin(pnt%, i%) = 0!
        END IF

    NEXT k%

    FOR f% = 1 TO unit(l%, 0) - 1

        pnt% = unit(l%, f%)
        pnt1% = modu(pnt%).p1

        IF f% <> unit(l%, 0) THEN
           pnt2% = unit(l%, f% + 1): pnt3% = modu(pnt2%).p1
           IF aqui(pnt3%).perm <> 0 THEN
             IF aqui(pnt3%).volume / (aqui(pnt3%).por - aqui(pnt3%).imc) <
aqui(pnt3%).depth THEN
                CALL potinfiltration(pnt2%, pnt3%, i%, potinfil)
             ELSE
                potinfil = aqui(pnt3%).perm
             END IF
           END IF

           IF modu(pnt%).typ = 1 THEN

              CALL depthforinfil.plane(pnt%, pnt1%, i%, availdepth(f%))

              IF availdepth(f%) * 1000 < potinfil * tint THEN
                  infilout(f%) = availdepth(f%) * 1000 / tint

              ELSE
                  infilout(f%) = potinfil
              END IF
            ' LPRINT " infiltration"; infilout
```

```
            ELSE
                effecstor = aqui(pnt1%).por - aqui(pnt1%).imc

               IF aqui(pnt1%).volume / effecstor < aqui(pnt1%).depth THEN
                    infilin = 0
               ELSE
                    infilin = infilout(f% - 1)
               END IF

               CALL    depthforinfil.aqui(infilin,    pnt%,    pnt1%,    i%,
availdepth(f%))

               IF availdepth(f%) * 1000 >= aqui(pnt1%).perm * tint THEN
                  infilout(f%) = aqui(pnt1%).perm

                  IF infilout(f%) > potinfil THEN
                       infilout(f%) = potinfil
                  END IF

               ELSEIF availdepth(f%) * 1000 < potinfil * tint THEN

                       infilout(f%) = availdepth(f%) * 1000 / tint

               ELSEIF availdepth(f%) * 1000 > potinfil * tint THEN

                       infilout(f%) = potinfil

               END IF
            END IF
         END IF
      NEXT f%

      FOR f% = unit(1%, 0) TO 2 STEP -1
        pnt% = unit(1%, f%)
        pnt1% = modu(pnt%).p1
        effecstor = aqui(pnt1%).por - aqui(pnt1%).imc
        IF aqui(pnt1%).perm <> 0 THEN
         IF ABS(aqui(pnt1%).volume / effecstor - aqui(pnt1%).depth) <= .001
THEN
            denom = aqui(pnt1%).lng * aqui(pnt1%).wid * aqui(pnt1%).por
            infilt = aqui(pnt1%).depth - aqui(pnt1%).yprev
            infilt = infilt + (aqui(pnt1%).cap - qin(pnt%, i% - 1)) * tint *
3600 / denom
            infilt = infilt * aqui(pnt1%).por * 1000 / tint
            infilt = infilt + infilout(f%)
            IF infilt <= infilout(f% - 1) THEN
                infilout(f% - 1) = infilt
            END IF
         END IF
        END IF

      NEXT f%

      FOR f% = 1 TO unit(1%, 0)

        pnt% = unit(1%, f%)
        pnt1% = modu(pnt%).p1
              LPRINT "potinfil", potinfil
        IF f% <> unit(1%, 0) THEN
          pnt2% = unit(1%, f% + 1): pnt3% = modu(pnt2%).p1

          IF modu(pnt%).typ = 1 THEN

             IF ABS(availdepth(f%) * 1000! - infilout(f%) * tint) < .0001 THEN
                   qout(pnt%, i%) = .0001
             ELSE
```

```
                CALL overroute1(infilout(f%), pnt%, pnt1%, i%)

            END IF

'            LPRINT , i%, modu(pnt%).modno
'               LPRINT , "availdepth="; availdepth; "infilout="; infilout;
"rain="; rain(i%)
'            LPRINT "q1="; qin(pnt%, i% - 1), "q3=", qout(pnt%, i% - 1)
'            LPRINT "q2=", qin(pnt%, i%), "qout=", qout(pnt%, i%)
'            LPRINT

         ELSE

            IF availdepth(f%) * 1000 = infilout/f%) * tint THEN
                qout(pnt%, i%) = 0
            ELSE
                IF aqui(pnt1%).volume / effecstor < aqui(pnt1%).depth THEN
                   infilin = 0
                ELSE
                    infilin = infilout(f% - 1)
                END IF

                CALL aquiroute1(infilin, infilout(f%), pnt%, pnt1%, i%)

            END IF

'            LPRINT , i%, modu(pnt%).modno
'               LPRINT , "availdepth="; availdepth; "infilout="; infilout;
"rain="; rain(i%)
'            LPRINT "q1="; qin(pnt%, i% - 1), "q3=", qout(pnt%, i% - 1)
'            LPRINT "q2=", qin(pnt%, i%), "qout=", qout(pnt%, i%)
'            LPRINT

         END IF

         IF aqui(pnt3%).perm <> 0 THEN

            IF aqui(pnt3%).volume / (aqui(pnt3%).por - aqui(pnt3%).imc) <
aqui(pnt3%).depth THEN
                  CALL checkaquisat(pnt2%, pnt3%, i%, infilout(f%))
            END IF
              '  LPRINT "volume under aqui"; volume(pnt2%, i%)
         END IF

       ELSE

         IF aqui(pnt1%).perm <> 0 THEN

             effecstor = aqui(pnt1%).por - aqui(pnt1%).imc

             IF aqui(pnt1%).volume / effecstor < aqui(pnt1%).depth THEN
                   infilin = 0
             ELSE
                   infilin = infilout(f% - 1)
             END IF

             IF aqui(pnt1%).lng <> 0 THEN
                  CALL aquiroute1(infilin, infilout(f%), pnt%, pnt1%, i%)
             END IF

         END IF

       END IF

       IF modu(pnt%).typ = 6 THEN
             IF qout(pnt%, i%) > aqui(pnt1%).cap THEN
                  pnt4% = unit(1%, f% - 1)
```

```
                over(pnt%, i%) = qout(pnt%, i%) - aqui(pntl%).cap
                qout(pnt4%, i%) = qout(pnt4%, i%) + qout(pnt%, i%) -
aqui(pntl%).cap
                qout(pnt%, i%) = aqui(pntl%).cap
                aqui(pntl%).yprev = aqui(pntl%).depth
            END IF
        END IF


    NEXT f%

END SUB

SUB overroute1 (infout, pnt%, pntl%, i%)

    alp = SQR(overlnd(pntl%).slo) / overlnd(pntl%).man
    q1 = qin(pnt%, i% - 1) / overlnd(pntl%).wid
    q2 = qin(pnt%, i%) / overlnd(pntl%).wid
    q3 = qout(pnt%, i% - 1) / overlnd(pntl%).wid
    q = (q1 + q2 + 2 * q3) / 4

    c1 = m * alp ^ (1 / m) * q1 ^ (1 - 1 / m)
    c2 = m * alp ^ (1 / m) * q2 ^ (1 - 1 / m)
    c3 = m * alp ^ (1 / m) * q3 ^ (1 - 1 / m)
    c = (c1 + c2 + 2 * c3) / 4

    theta = (1 + c * tint * 3600 / overlnd(pntl%).lng - q /
(overlnd(pntl%).lng * c * overlnd(pntl%).slo)) / 2
    k = overlnd(pntl%).lng / c

    IF theta < 0 THEN
        theta = 0
    ELSEIF theta > 1! THEN
        theta = 1!
    END IF

    CALL muscungcoeff(length, c, k, theta, cof1, cof2, cof3, cof4)

    qran = (rain(i%) - infout) * overlnd(pntl%).lng / 1000 *
        overlnd(pntl%).wid / 3600

    IF cof1 * qin(pnt%, i% - 1) + cof2 * qin(pnt%, i%) + qout(pnt%, i% - 1)
       * cof3 < 0 THEN
        qout(pnt%, i%) = cof4 * qran
        IF qout(pnt%, i%) <= 0 THEN
           qout(pnt%, i%) = .0001
        END IF
        EXIT SUB
    ELSE
        qout(pnt%, i%) = cof1 * qin(pnt%, i% - 1) + cof2 * qin(pnt%, i%)
+ qout(pnt%, i% - 1) * cof3 + cof4 * qran
    END IF

    IF qout(pnt%, i%) < 0 THEN
        qout(pnt%, i%) = .0001
        EXIT SUB
    END IF

    DO
      q4 = qout(pnt%, i%) / overlnd(pntl%).wid
      c4 = m * alp ^ (1 / m) * q4 ^ (1 - 1 / m)
      q = (q1 + q2 + q3 + q4) / 4
      c = (c1 + c2 + c3 + c4) / 4
      theta = (1 + c * tint * 3600 / overlnd(pntl%).lng - q /
(overlnd(pntl%).lng * c * overlnd(pntl%).slo)) / 2
      k = overlnd(pntl%).lng / c
      IF theta < 0 THEN
```

```
          theta = 0
        ELSEIF theta > 1 THEN
          theta = 1
        END IF
        CALL muscungcoeff(length, c, k, theta, cof1, cof2, cof3, cof4)

        qran = (rain(i%) - infout) * overlnd(pnt1%).lng / 1000 *
overlnd(pnt1%).wid / 3600
        qtemp = cof1 * qin(pnt%, i% - 1) + cof2 * qin(pnt%, i%) + qout(pnt%,
i% - 1) * cof3 + cof4 * qran

        diff = ABS(qtemp - qout(pnt%, i%))
        qout(pnt%, i%) = (qtemp + qout(pnt%, i%)) / 2

        IF qout(pnt%, i%) <= 0 THEN
           qout(pnt%, i%) = .0001
           EXIT SUB
        END IF

    LOOP UNTIL diff < .001

END SUB

SUB potinfiltration (pnt2%, pnt3%, i%, infilt)

      effecstorbot = aqui(pnt3%).por - aqui(pnt3%).imc
      gam = aqui(pnt3%).sorp * effecstorbot
      f = aqui(pnt3%).volume

     IF effecstorbot <> 0 THEN
          b = 2 * f - aqui(pnt3%).perm * tint / 1000!
          c = -2 * aqui(pnt3%).perm * tint * (gam + f) / 1000!
          df = (-b + SQR(b ^ 2 - 4 * c)) / 2
          infilt = df * 1000! / tint

          IF (f + df) / effecstorbot > aqui(pnt3%).depth - aqui(pnt3%).yprev
THEN
              infilt = aqui(pnt3%).perm * (1 + aqui(pnt3%).sorp /
                       aqui(pnt3%).depth)
          END IF

   END IF

END SUB

'Section of program to route flows through trapezoidal channels
'

DECLARE SUB muscungcoeff (length, c!, k!, theta!, cof1!, cof2!, cof3!, cof4!)
DECLARE SUB flowintomodule (ind1%, ind2%)
DECLARE SUB trapchanroute (l%, i%)
DECLARE SUB newtraptrap (flow!, yfinal!, l%, pnt%)
DECLARE SUB trapchar (pnt%, y!, b!, c!)

TYPE modconnectivity
     modno AS INTEGER
     ofl AS INTEGER
     typ AS INTEGER
     dsmod AS INTEGER
     infmod AS INTEGER
     pl AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
    lng AS SINGLE
```

```
    wid AS SINGLE
END TYPE

TYPE aquimod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wtl AS SINGLE
   sorp AS SINGLE
   perm AS SINGLE
   por AS SINGLE
   imc AS SINGLE
   cap AS SINGLE
   volume AS SINGLE
   yprev AS SINGLE
END TYPE

TYPE pipemod
    slo AS SINGLE
    diam AS SINGLE
    lng AS SINGLE
    man AS SINGLE
    cap AS SINGLE
    min AS SINGLE
END TYPE

TYPE trapmod
     slo AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     wid AS SINGLE
     ss1 AS SINGLE
     ss2 AS SINGLE
     mdep AS SINGLE
     cap AS SINGLE
END TYPE

TYPE stormod
    cl AS SINGLE
    sl AS SINGLE
    a AS SINGLE
    b AS SINGLE
    stlev AS SINGLE
    typ AS INTEGER
    ccu AS SINGLE
    cu AS SINGLE
    ccs AS SINGLE
    cs AS SINGLE
    csp AS SINGLE
    sp AS SINGLE
    depth AS SINGLE
    crit1 AS SINGLE
    crit2 AS SINGLE
    crit3 AS SINGLE
    prevstor AS SINGLE
END TYPE

TYPE compmod
    slo AS SINGLE
    lng AS SINGLE
    nosegs AS INTEGER
    nopts AS INTEGER
END TYPE

COMMON SHARED modu() AS modconnectivity, pipe() AS pipemod, storage() AS
stormod
```

```
COMMON SHARED overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON SHARED qin()  qout(), unit(), con(), ovflo(), order(), nomod%, noit%,
tint
COMMON SHARED m, over(), rain(), pi
COMMON SHARED prompt1$, flag.file%, tcode%
COMMON SHARED title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON SHARED sim.time, rain.time, hyeto.number%, newmod%, flag.insert%
COMMON SHARED compchan() AS compmod, x(), y(), n(), segno()

END

SUB newtraptrap (flow, yfinal, l%, pnt%)

  alp = SQR(trapchan(pnt%).slo) / trapchan(pnt%).man
  ybegin = (flow / (alp * trapchan(pnt%).wid)) ^ (3 / 5)
  ss = trapchan(pnt%).ss1 + trapchan(pnt%).ss2
  b = trapchan(pnt%).wid
  fact = SQR(1 + trapchan(pnt%).ss1 ^ 2) + SQR(1 + trapchan(pnt%).ss2 ^ 2)

  DO
       area = ybegin * b + ss * ybegin ^ 2 / 2
       perim = b + ybegin * fact
       r = area / perim
       dady = b + ybegin * ss
       drdy = dady / perim - area * fact / perim ^ 2
       dfdy = dady * r ^ (m - 1) + (m - 1) * area * r ^ (m - 2) * drdy
       yfinal = ybegin - (area * r ^ (m - 1) - flow / alp) / dfdy
       diff = ABS(yfinal - ybegin)
       ybegin = yfinal
  LOOP UNTIL diff < .001

END SUB

SUB trapchanroute (l%, i%)

          pnt% = modu(l%).p1
          FOR f% = 1 TO 3
             SELECT CASE f%
                CASE 1
                   flow = qin(l%, i% - 1)
                   CALL newtraptrap(flow, yin, l%, pnt%)
                   CALL trapchar(pnt%, yin, b1, c1)
                CASE 2
                   flow = qin(l%, i%)
                   CALL newtraptrap(flow, yint, l%, pnt%)
                   CALL trapchar(pnt%, yint, b2, c2)
                CASE 3
                   flow = qout(l%, i% - 1)
                   CALL newtraptrap(flow, yout, l%, pnt%)
                   CALL trapchar(pnt%, yout, b3, c3)
             END SELECT
          NEXT f%

          q = (qin(l%, i% - 1) + qin(l%, i%) + 2 * qout(l%, i% - 1)) / 4
          b = (b1 + b2 + 2 * b3) / 4
          c = (c1 + c2 + 2 * c3) / 4

'          LPRINT "i%="; i%
'          LPRINT "b,q,c="; b; q; c
'                theta = (1 - q / (b * c * trapchan(pnt%).lng *
trapchan(pnt%).slo)) / 2
          theta = (1 + c * tint * 3600 / trapchan(pnt%).lng - q / (b * c *
trapchan(pnt%).lng * trapchan(pnt%).slo)) / 2
'           LPRINT "thetatc", theta

          IF theta < 0 THEN
              theta = 0
```

```
          ELSEIF theta > 1 THEN
              theta = 1
          END IF

'          LPRINT "theta="; theta

          k = trapchan(pnt%).lng / c
          length = trapchan(pnt%).lng
          CALL muscungcoeff(length, c, k, theta, cof1, cof2, cof3, cof4)
'          LPRINT "cof1="; cof1; "cof2="; cof2; "cof3="; cof3; "cof4="; cof4

          qout(l%, i%) = cof1 * qin(l%, i% - 1) + cof2 * qin(l%, i%) +
qout(l%, i% - 1) * cof3

'          LPRINT "qout="; qout(l%, i%)
'          LPRINT

END SUB

SUB trapchar (pnt%, y, b, c)

     alp = SQR(trapchan(pnt%).slo) / trapchan(pnt%).man
     ss = trapchan(pnt%).ss1 + trapchan(pnt%).ss2
     fact = SQR(1 + trapchan(pnt%).ss1 ^ 2) + SQR(1 + trapchan(pnt%).ss2 ^ 2)
     area = trapchan(pnt%).wid * y + ss * y ^ 2 / 2
     wetperim = trapchan(pnt%).wid + y * fact
     b = trapchan(pnt%).wid + y * ss
     r = area / wetperim
     c = alp * r ^ (m - 1) * (m - (m - 1) * r * fact / b)


END SUB

SUB trapmodule (l%, i%)

     CALL flowintomodule(l%, i%)

     IF qin(l%, i%) <= 0 THEN
          qin(l%, i%) = .0001
     END IF
     IF qin(l%, i%) > trapchan(modu(l%).p1).cap THEN
          over(l%, i%) = qin(l%, i%) - trapchan(modu(l%).p1).cap
          qin(l%, i%) = trapchan(modu(l%).p1).cap
     END IF

     CALL trapchanroute(l%, i%)

     IF qout(l%, i%) < 0 THEN
        qout(l%, i%) = .0001
     END IF

     IF qout(l%, i%) > trapchan(modu(l%).p1).cap THEN
        qout(l%, i%) = trapchan(modu(l%).p1).cap
     END IF

END SUB

'Section of program to route flows down compound channels

DECLARE SUB flowintomodule (ind1%, ind2%)
DECLARE SUB interp (ht!, x1!, x2!, y1!, y2!, xint!, yint!)
DECLARE SUB searchforsegment (p%, ht!, y!(), lpt!, rpt!, flag!)
DECLARE FUNCTION area! (npts!, x1!(), y1!())
DECLARE FUNCTION perim! (npts!, x2!(), y2!())
DECLARE SUB celcalc (q!, q!(), a!(), cel!, nohts!)
DECLARE SUB topwidth (flow!, q!(), b!(), wid!, nohts!)
DECLARE SUB coeffs (k!, theta!, cof1!, cof2!, cof3!, cof4)
```

```
TYPE modconnectivity
     modno AS INTEGER
     ofl AS INTEGER
     typ AS INTEGER
     dsmod AS INTEGER
     infmod AS INTEGER
     pl AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
    lng AS SINGLE
    wid AS SINGLE
END TYPE

TYPE aquimod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wtl AS SINGLE
   sorp AS SINGLE
   perm AS SINGLE
   por AS SINGLE
   imc AS SINGLE
   cap AS SINGLE
   volume AS SINGLE
   yprev AS SINGLE
END TYPE

TYPE pipemod
    slo AS SINGLE
    diam AS SINGLE
    lng AS SINGLE
    man AS SINGLE
    cap AS SINGLE
    min AS SINGLE
END TYPE

TYPE trapmod
     slo AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     wid AS SINGLE
     ss1 AS SINGLE
     ss2 AS SINGLE
     mdep AS SINGLE
     cap AS SINGLE
END TYPE

TYPE stormod
    cl AS SINGLE
    sl AS SINGLE
    a AS SINGLE
    b AS SINGLE
    stlev AS SINGLE
    typ AS INTEGER
    ccu AS SINGLE
    cu AS SINGLE
    ccs AS SINGLE
    cs AS SINGLE
    csp AS SINGLE
    sp AS SINGLE
    depth AS SINGLE
    crit1 AS SINGLE
    crit2 AS SINGLE
```

```
    crit3 AS SINGLE
    prevstor AS SINGLE
END TYPE

TYPE compmod
    slo AS SINGLE
    lng AS SINGLE
    nosegs AS INTEGER
    nopts AS INTEGER
END TYPE


COMMON SHARED modu() AS modconnectivity, pipe() AS pipemod, storage() AS
stormod
COMMON SHARED overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON SHARED qin(), qout(), unit(), con(), ovflo(), order(), nomod%, noit%,
tint
COMMON SHARED expo, over(), rain(), pi
COMMON SHARED prompt1$, flag.file%, tcode%
COMMON SHARED title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON SHARED sim.time, rain.time, hyeto.number%, nawmod%, flag.insert%
COMMON SHARED compchan() AS compmod, x(), y(), n(), segno()

END

FUNCTION area (npts, x1(), y1())

         a = 0

         FOR k = 1 TO npts
           IF k = npts THEN
             a = a + (y1(1) + y1(npts)) * (x1(1) - x1(npts)) / 2
           ELSE
            a = a + (y1(k + 1) + y1(k)) * (x1(k + 1) - x1(k)) / 2
           END IF
         NEXT k

         area = ABS(a)

END FUNCTION

SUB celcalc (q, q(), a(), cel, nohts)

      i = 0
      flag = 0


      DO
         IF q = q(nohts) THEN
            cel = (q(nohts) - q(nohts - 1)) / (a(nohts) - a(nohts - 1))
            flag = 1
         ELSEIF q >= q(i) AND q < q(i + 1) THEN
            flag = 1
            cel = (q(i + 1) - q(i)) / (a(i + 1) - a(i))
         END IF
         i = i + 1
       LOOP UNTIL flag = 1


END SUB

SUB coeffs (k, theta, cof1, cof2, cof3, cof4)

      denom = tint * 3600 + k * (1 - theta)
      cof2 = (tint * 3600 - k * theta) / denom
      cof1 = k * theta / denom
      cof3 = (k * (1 - theta)) / denom
```

```
      cof4 = tint * 3600! / denom

END SUB

SUB compoundcha (l%, i%, q(), a(), b(), nohts)

    p% = modu(l%).p1

    CALL flowintomodule(l%, i%)

     IF qin(l%, i%) > q(nohts) THEN
          over(l%, i%) = qin(l%, i%) - q(nohts)
          qin(l%, i%) = q(nohts)
     END IF

        q1 = qin(l%, i% - 1)
        q2 = qin(l%, i%)
        q3 = qout(l%, i% - 1)
        FOR k% = 1 TO 3
           SELECT CASE k%
             CASE 1
                CALL celcalc(q1, q(), a(), c1, nohts)
                CALL topwidth(q1, q(), b(), b1, nohts)
             CASE 2
                CALL celcalc(q2, q(), a(), c2, nohts)
                CALL topwidth(q2, q(), b(), b2, nohts)
             CASE 3
                CALL celcalc(q3, q(), a(), c3, nohts)
                CALL topwidth(q3, q(), b(), b3, nohts)
           END SELECT
        NEXT k%
        c = (c1 + c2 + 2 * c3) / 4
        b = (b1 + b2 + 2 * b3) / 4
        qave = (q1 + q2 + 2 * q3) / 4

        theta = (1 + c * tint * 3600 / compchan(p%).lng - qave / (b * c *
compchan(p%).slo * compchan(p%).lng)) / 2

        IF theta < 0 THEN
           theta = 0
        ELSEIF theta > 1 THEN
           theta = 1
        END IF

        k = compchan(p%).lng / c

        CALL coeffs(k, theta, cof1, cof2, cof3, cof4)

        qran = rain(i%) * compchan(p%).lng / 1000! * b / 3600!
        qout(l%, i%) = q1 * cof1 + q2 * cof2 + q3 * cof3 + cof4 * qran
        IF qout(l%, i%) < 0 THEN
           qout(l%, i%) = .00001
        END IF

END SUB

SUB interp (ht, x1, x2, y1, y2, xint, yint)

     slope = (x2 - x1) / (y2 - y1)
     xint = slope * (ht - y1) + x1
     yint = ht

END SUB

FUNCTION perim (npts, x2(), y2())
```

```
        p = 0

        FOR k = 1 TO npts - 1
           xsqrd = (x2(k + 1) - x2(k)) ^ 2
           ysqrd = (y2(k + 1) - y2(k)) ^ 2
           p = p + (xsqrd + ysqrd) ^ .5
        NEXT k

        perim = p

END FUNCTION

SUB QvsA (l%, q(), a(), b(), nohts)

    REM $DYNAMIC

    REDIM height(20), x1(20), y1(20), x2(20), y2(20)

    p% = modu(l%).p1

    nohts = 10

    FOR k% = 2 TO compchan(p%).nopts
      IF y(p%, k%) > y(p%, k% - 1) THEN
        htlowpt = y(p%, k% - 1)
        EXIT FOR
      END IF
    NEXT k%

    depth = y(p%, 1) - htlowpt

    yinc = depth / nohts


    a(0) = 0
    q(0) = 0

    FOR k% = 1 TO nohts

       height(k%) = htlowpt + k% * yinc

       FOR kk% = 1 TO compchan(p%).nosegs

         lpt = segno(p%, kk%, 1)
         rpt = segno(p%, kk%, 2)

         CALL searchforsegment(p%, height(k%), y(), lpt, rpt, flag)

         IF flag = 1 THEN


            IF height(k%) > y(p%, lpt) AND height(k%) > y(p%, rpt) THEN
               flag1 = 1
            ELSEIF height(k%) <= y(p%, lpt) AND height(k%) <= y(p%, rpt) THEN
               flag1 = 2
            ELSEIF height(k%) <= y(p%, lpt) AND height(k%) > y(p%, rpt) THEN
               flag1 = 3
            ELSE
               flag1 = 4
            END IF

            SELECT CASE flag1

              CASE 1

                 npts = rpt - lpt + 3
                 x1(1) = x(p%, lpt)
```

```
                y1(1) = height(k%)
                x1(npts) = x(p%, rpt)
                y1(npts) = height(k%)

                FOR kkk% = 1 TO rpt - lpt + 1
                  x1(1 + kkk%) = x(p%, lpt + kkk% - 1)
                  x2(kkk%) = x(p%, lpt + kkk% - 1)
                  y1(kkk% + 1) = y(p%, lpt + kkk% - 1)
                  y2(kkk%) = y(p%, lpt + kkk% - 1)
                NEXT kkk%

                segarea = area(npts, x1(), y1())
                npts = npts - 2
                segperim = perim(npts, x2(), y2())

              CASE 2

                FOR kkk% = lpt + 1 TO rpt
                   IF height(k%) > y(p%, kkk%) THEN
                      CALL interp(height(k%), x(p%, kkk% - 1), x(p%, kkk%),
y(p%, kkk% - 1), y(p%, kkk%), x1(1), y1(1))
                      lpt = kkk%
                      EXIT FOR
                   END IF
                NEXT kkk%

                FOR kkk% = rpt - 1 TO lpt STEP -1
                   IF height(k%) > y(p%, kkk%) THEN
                      CALL interp(height(k%), x(p%, kkk% + 1), x(p%, kkk%),
y(p%, kkk% + 1), y(p%, kkk%), xval, yval)
                      rpt = kkk%
                      EXIT FOR
                   END IF
                NEXT kkk%

                npts = rpt - lpt + 3
                x1(npts) = xval
                y1(npts) = yval

                FOR kkk% = 1 TO rpt - lpt + 1
                   x1(kkk% + 1) = x(p%, lpt + kkk% - 1)
                   y1(kkk% + 1) = y(p%, lpt + kkk% - 1)
                NEXT kkk%

                segarea = area(npts, x1(), y1())
                segperim = perim(npts, x1(), y1())

              CASE 3

                FOR kkk% = lpt + 1 TO rpt
                   IF height(k%) > y(p%, kkk%) THEN
                      CALL interp(height(k%), x(p%, kkk% - 1), x(p%, kkk%),
y(p%, kkk% - 1), y(p%, kkk%), x1(1), y1(1))
                      lpt = kkk%
                      EXIT FOR
                   END IF
                NEXT kkk%

                npts = rpt - lpt + 3
                x1(npts) = x(p%, rpt)
                y1(npts) = height(k%)
                x2(1) = x1(1)
                y2(1) = y1(1)

                FOR kkk% = 1 TO rpt - lpt + 1
                   x1(kkk% + 1) = x(p%, lpt + kkk% - 1)
                   y1(kkk% + 1) = y(p%, lpt + kkk% - 1)
```

```
                x2(kkk% + 1) = x(p%, lpt + kkk% - 1)
                y2(kkk% + 1) = y(p%, lpt + kkk% - 1)
             NEXT kkk%

             segarea = area(npts, x1(), y1())
             npts = npts - 1
             segperim = perim(npts, x2(), y2())

          CASE 4

             FOR kkk% = rpt - 1 TO lpt STEP -1
                IF height(k%) > y(p%, kkk%) THEN
                   CALL interp(height(k%), x(p%, kkk% + 1), x(p%, kkk%),
y(p%, kkk% + 1), y(p%, kkk%), x1(1), y1(1))
                   rpt = kkk%
                   EXIT FOR
                END IF
             NEXT kkk%

             npts = rpt - lpt + 3
             x1(npts) = x(p%, lpt)
             y1(npts) = height(k%)
             x2(1) = x1(1)
             y2(1) = y1(1)

             FOR kkk% = 1 TO rpt - lpt + 1
                x1(kkk% + 1) = x(p%, rpt - kkk% + 1)
                y1(kkk% + 1) = y(p%, rpt - kkk% + 1)
                x2(kkk% + 1) = x(p%, rpt - kkk% + 1)
                y2(kkk% + 1) = y(p%, rpt - kkk% + 1)
             NEXT kkk%

             segarea = area(npts, x1(), y1())
             npts = npts - 1
             segperim = perim(npts, x2(), y2())

       END SELECT

    '  LPRINT "ht="; height(k%); "segarea="; segarea; "segperim="; segperim
          r = segarea / segperim
          qseg = SQR(compchan(p%).slo) * segarea * r ^ (2 / 3) / n(p%, kk%)
          q(k%) = q(k%) + qseg
          a(k%) = a(k%) + segarea
          b(k%) = x1(npts) - x1(1)
    '    LPRINT "a="; a(k%); "q="; q(k%)
    '    LPRINT "qseg="; qseg; "q(k%)="; q(k%); "a(k%)="; a(k%)

       END IF

     NEXT kk%
  ' LPRINT
    NEXT k%

    ERASE height, x1, x2, y1, y2

END SUB

REM $STATIC
SUB searchforsegment (p%, ht, y(), lpt, rpt, flag)


     flag = 0
     FOR k% = lpt TO rpt
        IF ht > y(p%, k) THEN
           flag = 1
           EXIT SUB
        END IF
```

```
      NEXT k%

END SUB

SUB topwidth (flow, q(), b(), wid, nohts)

     i = 0
     flag = 0

     DO
        IF flow >= q(i) AND flow <= q(i + 1) THEN
            slo = (b(i + 1) - b(i)) / (q(i + 1) - q(i))
            wid = slo * (flow - q(i)) + b(i)
            flag = 1
        END IF
        i = i + 1
     LOOP UNTIL flag = 1


END SUB

'Section of program to route flows down pipes

DECLARE FUNCTION areapipe! (angle!, pnt%)
DECLARE SUB muscungcoeff (length, c!, k!, theta!, cof1!, cof2!, cof3!, cof4!)
DECLARE SUB newtrappipes (flow!, thetfin!, l%, pnt%)
DECLARE SUB flowintomodule (ind1%, ind2%)
DECLARE SUB piperoute (l%, i%)
DECLARE SUB pipechar (indic%, angle!, b!, c!)
TYPE modconnectivity
     modno AS INTEGER
     of1 AS INTEGER
     typ AS INTEGER
     dsmod AS INTEGER
     infmod AS INTEGER
     p1 AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
    lng AS SINGLE
    wid AS SINGLE
END TYPE

TYPE aquimod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wtl AS SINGLE
   sorp AS SINGLE
   perm AS SINGLE
   por AS SINGLE
   imc AS SINGLE
   cap AS SINGLE
   volume AS SINGLE
   yprev AS SINGLE
END TYPE

TYPE pipemod
    slo AS SINGLE
    diam AS SINGLE
    lng AS SINGLE
    man AS SINGLE
    cap AS SINGLE
    min AS SINGLE
```

```
END TYPE

TYPE trapmod
     slo AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     wid AS SINGLE
     ss1 AS SINGLE
     ss2 AS SINGLE
     mdep AS SINGLE
     cap AS SINGLE
END TYPE

TYPE stormod
    cl AS SINGLE
    sl AS SINGLE
    a AS SINGLE
    b AS SINGLE
    stlev AS SINGLE
    typ AS INTEGER
    ccu AS SINGLE
    cu AS SINGLE
    ccs AS SINGLE
    cs AS SINGLE
    csp AS SINGLE
    sp AS SINGLE
    depth AS SINGLE
    crit1 AS SINGLE
    crit2 AS SINGLE
    crit3 AS SINGLE
    prevstor AS SINGLE
END TYPE

TYPE compmod
    slo AS SINGLE
    lng AS SINGLE
    nosegs AS INTEGER
    nopts AS INTEGER
END TYPE

COMMON SHARED modu() AS modconnectivity, pipe() AS pipemod, storage() AS
stormod
COMMON SHARED overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON SHARED qin(), qout(), unit(), con(), ovflo(), order(), nomod%, noit%,
tint
COMMON SHARED m, over(), rain(), pi
COMMON SHARED prompt1$, flag.file%, tcode%
COMMON SHARED title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON SHARED sim.time, rain.time, hyeto.number%, newmod%, flag.insert%
COMMON SHARED compchan() AS compmod, x(), y(), n(), segno()


END

FUNCTION areapipe (angle, pnt%)

    halfa = angle / 2
    areapipe = pipe(pnt%).diam ^ 2 * (halfa - SIN(halfa) * COS(halfa)) / 4

END FUNCTION

SUB flowintomodule (ind1%, ind2%)

     FOR f% = 1 TO con(ind1%, 0)
          qin(ind1%, ind2%) = qin(ind1%, ind2%) + qout(con(ind1%, f%), ind2%)
     NEXT f%
```

```
    FOR n% = 1 TO ovflo(ind1%, 0)
        qin(ind1%, ind2%) = qin(ind1%, ind2%) + over(ovflo(ind1%, n%),
ind2%)
    NEXT n%

END SUB

SUB muscungcoeff (length, c, k, theta, cof1, cof2, cof3, cof4)

    denom = tint * 3600 + k * (1 - theta)
    cof2 = (tint * 3600 - k * theta) / denom
'    IF cof2 < 0 THEN
'        theta = c * tint * 3600 / length
'        denom = tint * 3600 + k * (1 - theta)
'        cof2 = 0
'    END IF
    cof1 = k * theta / denom
    cof3 = (k * (1 - theta)) / denom
    cof4 = tint * 3600 / denom

END SUB

SUB newtrappipes (flow, thetfin, l%, pnt%)

    alp = SQR(pipe(pnt%).slo) / pipe(pnt%).man
    IF ABS(pipe(pnt%).cap - flow) < .0001 THEN
        thetfin = .827 * 2 * pi
        EXIT SUB
    END IF

    thetbeg = 5.15 - 2.2 * SQR(LOG(pipe(pnt%).cap) - LOG(flow))
    IF thetbeg < .1 * 2 * pi THEN
        thetbeg = .1 * 2 * pi
    END IF
    d = pipe(pnt%).diam

    DO
      halthet = thetbeg / 2
      area = areapipe(thetbeg, pnt%)
      perim = d * halthet
      r = area / perim
      dadth = d ^ 2 * SIN(halthet) ^ 2 / 4
      abit = d * COS(halthet) * SIN(halthet) / (2 * thetbeg ^ 2)
      abit1 = d * (1 - 2 * SIN(halthet) ^ 2) / (4 * thetbeg)
      drdth = abit - abit1
      dfdth = dadth * r ^ (m - 1) + area * (m - 1) * r ^ (m - 2) * drdth
      thetfin = thetbeg - (area * r ^ (m - 1) - flow / alp) / dfdth
      value = alp * area * r ^ (m - 1)
'      PRINT "dfdth"; dfdth; "thetfin"; thetfin; "flow calc"; value
      diff = ABS(1 - value / flow)
      thetbeg = thetfin
   LOOP UNTIL diff <= .01


END SUB

SUB pipechar (indic%, angle, b, c)

    d = pipe(indic%).diam
    alp = SQR(pipe(indic%).slo) / pipe(indic%).man
    hala = angle / 2
    coshala = COS(hala)
    sinhala = SIN(hala)
    y = d * (1 - coshala) / 2
    a = d ^ 2 * (hala - coshala * sinhala) / 4
    p = hala * d
    r = a / p
```

```
    b = 2 * (d * y - y ^ 2)
    abit = 2 / (d * angle ^ 2 * TAN(hala))
    abit2 = (2 - 1 / SIN(hala) ^ 2) / (d * angle)
    drda = abit + abit2
    c = alp * (r ^ (m - 1) + (m - 1) * a * r ^ (m - 2) * drda)

END SUB

SUB pipemodule (l%, i%)

     CALL flowintomodule(l%, i%)

     IF qin(l%, i%) < pipe(modu(l%).pl).min THEN
           qin(l%, i%) = pipe(modu(l%).pl).min
     END IF

     IF qin(l%, i%) > pipe(modu(l%).pl).cap THEN
          over(l%, i%) = qin(l%, i%) - pipe(modu(l%).pl).cap
          qin(l%, i%) = pipe(modu(l%).pl).cap
     END IF

     CALL piperoute(l%, i%)

     IF qout(l%, i%) < pipe(modu(l%).pl).min THEN
        qout(l%, i%) = pipe(modu(l%).pl).min
     END IF


END SUB

SUB piperoute (l%, i%)

     pnt% = modu(l%).pl

     FOR f% = 1 TO 3

          SELECT CASE f%
             CASE 1
                flow = qin(l%, i% - 1)
                CALL newtrappipes(flow, thetin, l%, pnt%)
                CALL pipechar(pnt%, thetin, b1, c1)
             CASE 2
                flow = qin(l%, i%)
                CALL newtrappipes(flow, thetint, l%, pnt%)
                CALL pipechar(pnt%, thetint, b2, c2)
             CASE 3
                flow = qout(l%, i% - 1)
                CALL newtrappipes(flow, thetout, l%, pnt%)
                CALL pipechar(pnt%, thetout, b3, c3)
           END SELECT

     NEXT f%
'     LPRINT "thetin="; thetin; "thetint="; thetint; "thetout="; thetout
'     LPRINT "c1="; c1; "c2="; c2; "c3="; c3
     q = (qin(l%, i% - 1) + qin(l%, i%) + 2 * qout(l%, i% - 1)) / 4
     b = (b1 + b2 + 2 * b3) / 4
     c = (c1 + c2 + 2 * c3) / 4

      theta = (1 + c * tint * 3600 / pipe(pnt%).lng - q / (b * c *
pipe(pnt%).lng * pipe(p    .slo)) / 2
'      theta = (1 - q / (b * c * pipe(pnt%).lng * pipe(pnt%).slo)) / 2
     IF theta < 0 THEN
         theta = 0
     ELSEIF theta > 1! THEN
         theta = 1!
     END IF
```

```
      k = pipe(pnt%).lng / c
      length = pipe(pnt%).lng
      CALL muscungcoeff(length, c, k, theta, cof1, cof2, cof3, cof4)
    '  LPRINT "theta="; theta; "b="; b; "c="; c
    '  LPRINT "cof1="; cof1; "cof2="; cof2; "cof3="; cof3; "cof4="; cof4
    '  LPRINT "q1="; qin(l%, i% - 1); "q2="; qin(l%, i%); "q3="; qout(l%, i%
- 1)
      qout(l%, i%) = cof1 * qin(l%, i% - 1) + cof2 * qin(l%, i%) + qout(l%, i%
- 1) * cof3
    '  LPRINT "q4="; qout(l%, i%)
    '  LPRINT "l%="; l%; "i%"; i%;
END SUB

'Program section to route flows through a storage module


DECLARE FUNCTION storvol! (pnt%, y!)
DECLARE SUB flowintomodule (ind1%, ind2%)
DECLARE SUB stornrap (pnt%, value!, ynew!, index!)
DECLARE SUB storageroute (l%, i%)
TYPE modconnectivity
     modno AS INTEGER
     of1 AS INTEGER
     typ AS INTEGER
     dsmo AS INTEGER
     infmod AS INTEGER
     p1 AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
    lng AS SINGLE
    wid AS SINGLE
END TYPE

TYPE aquamod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wtl AS SINGLE
   sorp AS SINGLE
   perm AS SINGLE
   por AS SINGLE
   imc AS SINGLE
   cap AS SINGLE
  volume AS SINGLE
   yprev AS SINGLE
END TYPE

TYPE pipemod
    slo AS SINGLE
    diam AS SINGLE
    lng AS SINGLE
    man AS SINGLE
    cap AS SINGLE
    min AS SINGLE
END TYPE

TYPE trapmod
     slo AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     wid AS SINGLE
     ss1 AS SINGLE
     ss2 AS SINGLE
```

```
     mdep AS SINGLE
     cap AS SINGLE
END TYPE

TYPE stormod
    cl AS SINGLE
    sl AS SINGLE
    a AS SINGLE
    b AS SINGLE
    stlev AS SINGLE
    typ AS INTEGER
    ccu AS SINGLE
    cu AS SINGLE
    ccs AS SINGLE
    cs AS SINGLE
    csp AS SINGLE
    sp AS SINGLE
    depth AS SINGLE
    crit1 AS SINGLE
    crit2 AS SINGLE
    crit3 AS SINGLE
    prevstor AS SINGLE
END TYPE

TYPE compmod
    slo AS SINGLE
    lng AS SINGLE
    nosegs AS INTEGER
    nopts AS INTEGER
END TYPE

COMMON SHARED modu() AS modconnectivity, pipe() AS pipemod, storage() AS
stormod
COMMON SHARED overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON SHARED qin(), qout(), unit(), con(), ovflo(), order(), nomod%, noit%,
tint
COMMON SHARED m, over(), rain(), pi
COMMON SHARED prompt1$, flag.file%, tcode%
COMMON SHARED title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON SHARED sim.time, rain.time, hyeto.number%, newmod%, flag.insert%
COMMON SHARED compchan() AS compmod, x(), y(), n(), segno()


END

SUB storagemod (l%, i%)

     CALL flowintomodule(l%, i%)

     CALL storageroute(l%, i%)


END SUB

SUB storageroute (l%, i%)

    pnt% = modu(l%).p1

    value = storage(pnt%).prevstor + qin(l%, i%) * tint * 3600

    IF value <= storage(pnt%).crit1 THEN
       qout(l%, i%) = 0!
    ELSEIF storage(pnt%).typ = 0 THEN
       CALL stornrap(pnt%, value, y, 4)
       qout(l%, i%) = storage(pnt%).csp * (y - storage(pnt%).sl) ^
storage(pnt%).sp
```

```
    ELSEIF storage(pnt%).typ = 1 THEN
       IF value > storage(pnt%).crit1 AND value <= storage(pnt%).crit2 THEN
           CALL stornrap(pnt%, value, y, 1)
           qout(l%, i%) = storage(pnt%).ccu * (y - storage(pnt%).cl) ^
storage(pnt%).cu
       ELSEIF value > storage(pnt%).crit2 AND value <= storage(pnt%).crit3
THEN
           CALL stornrap(pnt%, value, y, 2)
           qout(l%, i%) = storage(pnt%).ccs * (y - storage(pnt%).cl - .5 *
storage(pnt%).depth) ^ storage(pnt%).cs
       ELSEIF value > storage(pnt%).crit3 THEN
           CALL stornrap(pnt%, value, y, 3)
           qout(l%, i%) = storage(pnt%).ccs * (y - storage(pnt%).cl -
storage(pnt%).depth * .5) ^ storage(pnt%).cs
           qout(l%, i%) = qout(l%, i%) + storage(pnt%).csp * (y -
storage(pnt%).sl) ^ storage(pnt%).sp
       END IF
    END IF

    IF value <= storage(pnt%).crit1 THEN
        storage(pnt%).prevstor = value
    ELSE
        storage(pnt%).prevstor = storvol(pnt%, y)
    END IF


END SUB

SUB stornrap (pnt%, value, ynew, index)

     a = storage(pnt%).a: b = storage(pnt%).b
     cl = storage(pnt%).cl: sl = storage(pnt%).sl

     SELECT CASE index

          CASE 1
             ccu = storage(pnt%).ccu: cu = storage(pnt%).cu
             yold = .75 * storage(pnt%).depth + cl

             DO
               dfdy = a * b * yold ^ (b - 1) + ccu * cu * (yold - cl) ^ (cu
- 1) * tint * 3600
               numer = (a * yold ^ b + ccu * (yold - cl) ^ cu * 3600 * tint
- value) / dfdy
               ynew = yold - numer
               diff = ABS(1 - ynew / yold)
               yold = ynew
             LOOP UNTIL diff <= .01

          CASE 2
             ccs = storage(pnt%).ccs: cs = storage(pnt%).cs
             yold = (1.5 * storage(pnt%).depth + sl) / 2

             DO
               dfdy = a * b * yold ^ (b - 1) + ccs * cs * (yold - cl - .5 *
storage(pnt%).depth) ^ (cs - 1)
               numer = (yold - cl - .5 * storage(pnt%).depth) ^ cs
               ynew = yold - (a * yold ^ b + ccs * numer * 3600 * tint -
value) / dfdy
               diff = ABS(1 - ynew / yold)
               yold = ynew
             LOOP UNTIL diff <= .01

          CASE 3
             csp = storage(pnt%).csp: sp = storage(pnt%).sp
             ccs = storage(pnt%).ccs: cs = storage(pnt%).cs
             yold = sl + .1
```

```
            DO
               abit = a * b * yold ^ (b - 1)
               abit1 = ccs * cs * (yold - c1 - .5 * storage(pnt%).depth) ^
(cs - 1)
               abit2 = csp * sp * (yold - s1) ^ (sp - 1)
               dfdy = abit + (abit1 + abit2) * tint * 3600
               f = a * yold ^ b + (ccs * (yold - c1 - .5 *
storage(pnt%).depth) ^ cs) * 3600 * tint
               f = f + (csp * (yold - s1) ^ sp) * tint * 3600
               ynew = yold - (f - value) / dfdy
               diff = ABS(1 - ynew / yold)
               yold = ynew
            LOOP UNTIL diff <= .01

         CASE 4
            csp = storage(pnt%).csp: sp = storage(pnt%).sp
            yold = s1 + .1

            DO
               dfdy = (csp * sp * (yold - s1) ^ (sp - 1)) * tint * 3600 + a
* b * yold ^ (b - 1)
               f = a * yold ^ b + csp * (yold - s1) ^ sp * tint * 3600
               ynew = yold - (f - value) / dfdy
               IF ynew > 0 THEN
                  diff = ABS(1 - ynew / yold)
                  value1 = a * ynew ^ b + csp * (ynew - s1) ^ sp * tint *
3600
                  yold = ynew
               ELSE
                  diff = 0!
               END IF
            LOOP UNTIL diff <= .01
     END SELECT

END SUB

FUNCTION storvol (pnt%, y)

        storvol = storage(pnt%).a * y ^ storage(pnt%).b

END FUNCTION
```

CALIBRATION SUBPROGRAM

```
'Program for comparison of observed to simulated results

DECLARE SUB goodoffit ()
DECLARE SUB interp (tim!(), time!(), flow!(), i%, j%, flw!)

TYPE modconnectivity
     modno AS INTEGER
     ofl AS INTEGER
     typ AS INTEGER
     dsmod AS INTEGER
     infmod AS INTEGER
     pl AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
    lng AS SINGLE
    wid AS SINGLE
END TYPE

TYPE aquimod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wtl AS SINGLE
   sorp AS SINGLE
   perm AS SINGLE
   por AS SINGLE
   imc AS SINGLE
   cap AS SINGLE
   volume AS SINGLE
   yprev AS SINGLE
END TYPE

TYPE pipemod
    slo AS SINGLE
    diam AS SINGLE
    lng AS SINGLE
    man AS SINGLE
    cap AS SINGLE
    min AS SINGLE
END TYPE

TYPE trapmod
     slo AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     wid AS SINGLE
     ss1 AS SINGLE
     ss2 AS SINGLE
     mdep AS SINGLE
     cap AS SINGLE
END TYPE

TYPE stormod
    ol AS SINGLE
    sl AS SINGLE
    a AS SINGLE
    b AS SINGLE
    stlev AS SINGLE
    typ AS INTEGER
    ccu AS SINGLE
    cu AS SINGLE
```

```
    ccs AS SINGLE
    cs AS SINGLE
    csp AS SINGLE
    sp AS SINGLE
    depth AS SINGLE
    crit1 AS SINGLE
    crit2 AS SINGLE
    crit3 AS SINGLE
    prevstor AS SINGLE
END TYPE

TYPE compmod
    slo AS SINGLE
    lng AS SINGLE
    nosegs AS INTEGER
    nopts AS INTEGER
END TYPE


COMMON SHARED modu() AS modconnectivity, pipe() AS pipemod, storage() AS
stormod
COMMON SHARED overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON SHARED qin(), qout(), unit(), cor(), ovflo(), order(), nomod%, noit%,
tint
COMMON SHARED expo, over(), rain(), pi
COMMON SHARED prompt1$, flag.file%, tcode%
COMMON SHARED title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON SHARED sim.time, rain.time, hyeto.number%, newmod%, flag.insert%
COMMON SHARED compchan() AS compmod, x(), y(), n(), segno()

REM $DYNAMIC

CALL goodoffit

tcode% = 2

SCREEN 0

CLS

COLOR 0, 2
LOCATE 11, 28
PRINT "Chaining editor subprogram"
COLOR 3, 0

CHAIN "editor"

errorproc:

      SELECT CASE ERR
            CASE 25, 24, 68

               BEEP: BEEP: PRINT : PRINT : PRINT "printer not connected or
switched on ... press any key"
               WHILE INKEY$ = "": WEND
               RESUME begin

           CASE 53, 76
               BEEP: BEEP: PRINT : PRINT : PRINT "File does not exist ...
press any key"
               WHILE INKEY$ = "": WEND
               RESUME begin


           CASE 64, 52

               BEEP
```

```
                BEEP
                PRINT
                PRINT
                PRINT "Bad filename (limited to 8 characters) ... press any
key"
                WHILE INKEY$ = "": WEND
                RESUME begin

            CASE ELSE
                ON ERROR GOTO 0
        END SELECT

begin: CALL goodoffit

    tcode% = 2
    SCREEN 0

    CLS

    COLOR 0, 2
    LOCATE 11, 28
    PRINT "Chaining editor subprogram"
    COLOR 3, 0


    CHAIN "editor"

END

REM $STATIC

'Subroutine to calculate goodness of fit criteria

SUB goodoffit

REM $DYNAMIC

REDIM tim(160), flo(160), time(160), flow(160)

          ON ERROR GOTO errorproc

          SCREEN 0: WIDTH 80: CLS
          LOCATE 4, 1: COLOR 3, 0: PRINT "Enter observed data filename"
          COLOR 7, 0: LOCATE 4, 30: INPUT "", filename$

          CLS

getno:    LOCATE 4, 1: COLOR 3, 0: PRINT "Module number of simulated output"
          COLOR 7, 0: LOCATE 4, 35: INPUT "", num%

          FOR i% = 1 TO nomod%
             IF modu(i%).modno = num% THEN
                modnum% = i%
                GOTO grapplo
             END IF
    '         IF num% = 0 THEN GOTO last
          NEXT i%

          BEEP: BEEP: PRINT : PRINT : PRINT "Module "; num%; " does not exist
... press any key"
          WHILE INKEY$ = "": WEND
          CLS
          GOTO getno

grapplo: OPEN filename$ FOR INPUT AS #1

          INPUT #1, nopts%
```

```
peak.obs = 0
FOR j% = 0 TO nopts% - 1
   INPUT #1, tim(j%), flo(j%)
   IF flo(j%) > peak.obs THEN
       peak.obs = flo(j%)
   END IF
NEXT j%

peak.sim = 0
FOR j% = 0 TO noit%
    IF qout(modnum%, j%) > peak.sim THEN
        peak.sim = qout(modnum%, j%)
    END IF
NEXT j%

l% = 0

FOR i% = 0 TO noit%
   time(i%) = i% * tint * 60

   FOR j% = l% TO nopts% - 1
       IF time(i%) < tim(j%) THEN
        IF j% <> 0 THEN
         CALL interp(tim(), time(), flo(), i%, j%, y)
         flow(i%) = y
         l% = j%
         EXIT FOR
        ELSE
         flow(i%) = flo(j%)
        END IF
       ELSEIF time(i%) = tim(j%) THEN
         flow(i%) = flo(j%)
       END IF
   NEXT j%

NEXT i%


SCREEN 9
VIEW (10, 10)-(610, 320), , 1
xmax = tint * noit% * 60
IF peak.sim > peak.obs THEN
    ymax = peak.sim + .01
ELSE
    ymax = peak.obs + .01
END IF

WINDOW (0, 0)-(xmax, ymax)

FOR j% = 1 TO nopts% - 1
   x1 = tim(j% - 1)
   y1 = flo(j% - 1)
   x2 = tim(j%)
   y2 = flo(j%)
   LINE (x1, y1)-(x2, y2), , , &HFF00
NEXT j%


FOR j% = 1 TO noit%
   x1 = (j% - 1) * tint * 60
   y1 = qout(modnum%, j% - 1)
   x2 = j% * tint * 60
   y2 = qout(modnum%, j%)
   LINE (x1, y1)-(x2, y2)
NEXT j%
WHILE INKEY$ = "": WEND
```

```
        rat.peak = peak.sim / peak.obs

        CLS 0

'       LOCATE 4, 1: COLOR 3, 0: PRINT "Start and finish point and time step
for stats calcs"
'        COLOR 7, 0: LOCATE 4, 57: INPUT "", start, finish, timestep

        count = 0
        total = 0
        volsim = 0
        FOR j% = 1 TO noit%
          count = count + 1
          total = total + qout(modnum%, j%)
          ave = (qout(modnum%, j% - 1) + qout(modnum%, j%)) * tint * 3600 /
2
          volsim = volsim + ave
        NEXT j%
        ave.sim = total / count


        count = 0
        total = 0
        volobs = 0
        FOR j% = 1 TO noit%
           count = count + 1
           total = total + flow(j%)
           volobs = volobs + (flow(j% - 1) + flow(j%)) * tint * 3600 / 2
           diff1 = flow(j% - 1) - qout(modnum%, j% - 1)
           diff2 = flow(j%) - qout(modnum%, j%)
           a = a + ABS(((diff1 + diff2) * tint * 3600) / 2)
        NEXT j%
        ave.obs = total / count


        count = 0
        FOR j% = 1 TO noit%
           count = count + 1
           diff = flow(j%) - qout(modnum%, j%)
           diffe = flow(j%) - ave.obs
           diffsqr = diffsqr + diff ^ 2
           absdiff = absdiff + ABS(diff)
           diffesqr = diffesqr + diffe ^ 2
           IF flow(j%) > 0 THEN
              pee = pee + (diff / flow(j%)) ^ 2
           END IF
        NEXT j%
        eff = 1 - diffsqr / diffesqr
        pee = (pee / count) ^ .5


        PRINT "                    Observed    Simulated    Ratio (sim/obs) "
        PRINT "  ";
        PRINT "Peak(cub m/s)   ";
        PRINT USING "####.###"; peak.obs;
        PRINT "     ";
        PRINT USING "#####.###"; peak.sim;
        PRINT "      ";
        PRINT USING "##.####"; rat.peak
        PRINT "  ";
        PRINT "Volume (cub m)  ";
        PRINT USING "#####.##"; volobs;
        PRINT "     ";
        PRINT USING "######.##"; volsim;
        PRINT "      ";
        PRINT USING "##.####"; volsim / volobs
        PRINT "  ";
```

```
        PRINT "Ave flw(cub m/s)";
        PRINT USING "##.#####"; ave.obs;
        PRINT "    ";
        PRINT USING "###.#####"; ave.sim;
        PRINT "     ";
        PRINT USING "##.####"; ave.sim / ave.obs
        PRINT
        PRINT
        PRINT "  ";
        PRINT "SSR (cub m/s)^2 ";
        PRINT USING "###.####"; diffsqr
        PRINT "  ";
        PRINT "SAR (cub m/s)    ";
        PRINT USING "###.####"; absdiff
        PRINT "  ";
        PRINT "Coef Eff          ";
        PRINT USING "#.##"; eff
        PRINT "  ";
        PRINT "Prop err of est    ";
        PRINT USING "##.##"; pee
        PRINT "  ";
        PRINT "Abs Areas of div    ";
        PRINT USING "#####.#"; a

        WHILE INKEY$ = "": WEND

        CLOSE #1

        ERASE tim, flo, flow, time

END SUB

REM $STATIC

'Subroutine to interpolate observed hydrograph for change in time step

SUB interp (tim(), time(), flo(), i%, j%, flw)

     slo = (flo(j%) - flo(j% - 1)) / (tim(j%) - tim(j% - 1))
     flw = flo(j% - 1) + slo * (time(i%) - tim(j% - 1))

END SUB
```

GRAPHICS CONNECTIVITY SUBPROGRAM

```
'Program to output connectivity to screen

DECLARE SUB dataout1 (file1$)
DECLARE SUB datain1 (file1$)
TYPE modconnectivity
     modno AS INTEGER
     ofl AS INTEGER
     typ AS INTEGER
     dsmod AS INTEGER
     infmod AS INTEGER
     pl AS INTEGER
END TYPE

TYPE overmod
    man AS SINGLE
    slo AS SINGLE
    lng AS SINGLE
    wid AS SINGLE
END TYPE

TYPE aquimod
   slo AS SINGLE
   wid AS SINGLE
   lng AS SINGLE
   depth AS SINGLE
   wtl AS SINGLE
   sorp AS SINGLE
   perm AS SINGLE
   por AS SINGLE
   imc AS SINGLE
   cap AS SINGLE
   volume AS SINGLE
   yprev AS SINGLE
END TYPE

TYPE pipemod
    slo AS SINGLE
    diam AS SINGLE
    lng AS SINGLE
    man AS SINGLE
    cap AS SINGLE
    min AS SINGLE
END TYPE

TYPE trapmod
     slo AS SINGLE
     lng AS SINGLE
     man AS SINGLE
     wid AS SINGLE
     ss1 AS SINGLE
     ss2 AS SINGLE
     mdep AS SINGLE
     cap AS SINGLE
END TYPE

TYPE stormod
    cl AS SINGLE
    sl AS SINGLE
    a AS SINGLE
    b AS SINGLE
    stlev AS SINGLE
    typ AS INTEGER
    ccu AS SINGLE
    cu AS SINGLE
    ccs AS SINGLE
    cs AS SINGLE
```

```
    csp AS SINGLE
    sp AS SINGLE
    depth AS SINGLE
    crit1 AS SINGLE
    crit2 AS SINGLE
    crit3 AS SINGLE
    prevstor AS SINGLE
END TYPE

TYPE compmod
    slo AS SINGLE
    lng AS SINGLE
    nosegs AS INTEGER
    nopts AS INTEGER
END TYPE


COMMON SHARED modu() AS modconnectivity, pipe() AS pipemod, storage() AS
stormod
COMMON SHARED overlnd() AS overmod, aqui() AS aquimod, trapchan() AS trapmod
COMMON SHARED qin(), qout(), unit(), con(), ovflo(), order(), nomod%, noit%,
tint
COMMON SHARED expo, over(), rain(), pi
COMMON SHARED prompt1$, flag.file%, tcods%
COMMON SHARED title$, file$, flag.esc%, p1%, p2%, p3%, p4%, p5%, p6%
COMMON SHARED sim.time, rain.time, hyeto.number%, newmod%, flag.insert%
COMMON SHARED compchan(' AS compmod, x(), y(), n(), segno()

1     'File PLANLAST.BAS in QB
2     'This file forms a combination of PLAN.BAS (ex-PLAN6.BAS) and
3     'PLANTA.BAS (ex-PLAN8.BAS).
4     'Routine to draw n sequences with unlimited side branches up to the 2nd
5     'order, forming a modular representation of one or more catchments.
6     'Dr W.A.J. Paling, 19-7-1990

7     '

      file1$ = "b:temp.skm"

      CALL dataout1(file1$)


      ERASE con, over, order, ovflo, unit, rain, qin, qout, trapchan
      ERASE aqui, pipe, compchan, storage, overlnd

      REM $DYNAMIC

10    REDIM i1(140), i2(140), typ(140), k1(140), k2(140, 15), k3(400), k4(400,
15)
11    REDIM L2(400), TEMP(140), SEQ(50, 30), SUBR(5, 50, 20), TAL(10), NOL(30)
12    REDIM RECX(50), RECY(50), RAIN1(900), RAIN2(900), RAIN3(900), p(30)
13    'The dimension of K3, K4 and L2 should at least correspond with the
      CLS
      LOCATE 11, 28
      COLOR 0, 2: PRINT "screen connectivity routine"
      nall = 0
30    FOR i = 1 TO nomod%
50      IF modu(i).modno < 900 THEN
           nall = nall + 1
           i1(nall) = modu(i).modno
           i2(nall) = modu(i).dsmod
           typ(nall) = modu(i).typ
        END IF
80    NEXT i

100   '---------------- Backward Connectivity --------------------
110   FOR i = 1 TO nall          'K1() = quantity of incoming nodes
```

```
120    k1(i) = 0              'K2(,) = data file sequence nr of incoming node
130    FOR j = 1 TO nall
140      IF i2(j) <> i1(i) THEN 170
150      k1(i) = k1(i) + 1
160      k2(i, k1(i)) = j
         'PRINT "I="; I; "K1(I)="; K1(I); "K2(I,K1(I))="; K2(I, K1(I))
170    NEXT j
180  NEXT i
200  '----------------- Forward Connectivity ---------------------
210  FOR i = 1 TO nall       'L2() = data file sequence nr of subsequent node
220    FOR j = 1 TO nall
230      IF i2(i) = 0 GOTO 270
240      IF i1(j) <> i2(i) THEN 260
250      L2(i) = j: GOTO 270
260    NEXT j
270  NEXT i
300  '--------------- Establish the Individual Rows ---------------
310  FOR i = 1 TO nall                  'change from sequence nrs to code nrs
320    k3(i1(i)) = k1(i)
330    'PRINT "I="; I; "I1(I)="; I1(I); "K3(I1(I))="; K3(I1(I)),
340    FOR j = 1 TO k3(i1(i))
350      k4(i1(i), j) = i1(k2(i, j))
360      'PRINT "J="; J; "K4(I1(I),J)="; K4(I1(I), J),
370    NEXT j: 'PRINT
380  NEXT i: 'PRINT
     L = 0: LMAX = 0
     FOR i = 1 TO nall
       IF k1(i) <> 0 THEN 520            'find a starting point of a row
       K = i: L = L + 1: NL = 1: j = 1
       TEMP(j) = i1(i)
500    IF L2(K) = 0 THEN 510             'arrived at the end of a row
       j = j + 1: NL = NL + 1
       TEMP(j) = i1(L2(K))
       K = L2(K)
       GOTO 500
510    IF NL > LMAX THEN LMAX = NL       'NL = nr of blocks in a row
       FOR j = 1 TO NL
         SEQ(L, j) = TEMP(NL - j + 1)    'SEQ stores sequences of code nrs
         'PRINT "L="; L; "J="; J; "SEQ(L,J)="; SEQ(L, J)
       NEXT                              'LMAX = max nr of columns
520  NEXT                                'L = max nr of rows
     XL = 610 / LMAX: YL = 320 / L
     XL1 = .1 * XL: XL2 = .2 * XL: XL3 = .3 * XL: XL4 = .4 * XL
     XL5 = .5 * XL: XL6 =  6 * XL: XL7 = .7 * XL: XL8 = .8 * XL
     XL9 = .9 * XL
     XL05 = .05 * XL: XL25 = .25 * XL: XL65 = .65 * XL: XL85 = .85 * XL
     YL1 = .1 * YL: YL2 = .2 * YL: YL3 = .3 * YL: YL4 = .4 * YL
     YL5 = .5 * YL: YL6 = .6 * YL: YL7 = .7 * YL: YL8 = .8 * YL
     YL9 = .9 * YL: YL05 = .05 * YL
600  '----------------- Create Subsets -------------------------
     SALL = 1: PP = 1: p(PP) = SEQ(1, 1)
     FOR K = 1 TO L
       TAL(K) = 0                    'TAL() = nr of rows in a subset
       FOR i = 1 TO L
         IF SEQ(i, 1) <> p(PP) THEN AA = SEQ(i, 1): GOTO 610
         TAL(K) = TAL(K) + 1
         FOR j = 1 TO LMAX: SUBR(K, TAL(K), j) = SEQ(i, j): NEXT
610    NEXT
       FOR i = 1 TO K
'        PRINT AA, p(i)
         IF (AA = p(i)) OR (AA = 0) THEN 620
       NEXT
'      PRINT
       SALL = SALL + 1: PP = PP + 1: p(PP) = AA
     NEXT
620  FOR S = 1 TO SALL
       PRINT USING "Subset nr = ##"; S
```

```
      FOR i = 1 TO TAL(S)
'        PRINT USING "Row = ####   "; i;
        FOR j = 1 TO LMAX
'          PRINT USING "######"; SUBR(S, i, j);
        NEXT
'       PRINT
      NEXT
    NEXT
    FOR i = 1 TO 500: NEXT
1000 '---------------------- Start Drawing --------------------------
    CLS : KEY OFF: SCREEN 9: COLOR 1, 0
    XP = 610: YP = YL
    FOR SS = 1 TO SALL
      '----- establish longest row in subset -----
      NLON = 0                        'NLON = nr of nodes in longest row
      FOR i = 1 TO TAL(SS)           'RLON = subset sequence nr of longest row
        M = 0
        FOR j = 1 TO LMAX + 1
          IF SUBR(SS, i, j) <> 0 THEN M = M + 1: GOTO 1010
          IF M > NLON THEN NLON = M: RLON = i: GOTO 1020
1010      NEXT
1020    NEXT
        '----- draw longest row -----
        FOR i = 1 TO NLON
          NODE = SUBR(SS, RLON, i)
          XP = XP - XL: GOSUB 3000
        NEXT
        YP1 = YP - YL5
        '----- branches -----
        FOR n = NLON - 1 TO 1 STEP -1
          IF k3(SUBR(SS, RLON, n)) <= 1 THEN 1100
          '----- first order branches -----
          XP1 = XP
          NOL(1) = SUBR(SS, RLON, n + 1)
          FOR K = 2 TO k3(SUBR(SS, RLON, n))
            YP = YP + YL
            SNLON = 0
            FOR i = 1 TO TAL(SS)
              IF SUBR(SS, i, n) <> SUBR(SS, RLON, n) THEN 1040
              FOR j = 1 TO k3(SUBR(SS, RLON, n))
                IF SUBR(SS, i, n + 1) = NOL(j) THEN 1040
              NEXT
              M = 0
              FOR j = n + 1 TO NLON + 1
                IF SUBR(SS, i, j) <> 0 THEN M = M + 1: GOTO 1030
                IF M > SNLON THEN SNLON = M: SRLON = i: GOTO 1040
1030          NEXT j
1040        NEXT i
            FOR i = 1 TO SNLON  'draw 1st order branch
              NODE = SUBR(SS, SRLON, n + i)
              GOSUB 3000: XP = XP - XL
            NEXT
            IF (k3(L2(NODE)) = 1) AND (L2(NODE) <> n) THEN 1050
            GOSUB 1200 'second order branches
1050        NOL(K) = SUBR(SS, SRLON, n + 1)
            XP = XP1
          NEXT K
          LINE (XP + XL, YP - YL5)-(XP + XL, YP1)
          XP = XP1
1100      XP = XP + XL
          FOR i = 1 TO k3(SUBR(SS, RLON, n)): NOL(i) = 0: NEXT
        NEXT n
        XP = 320: YP = YP + YL
    NEXT SS
    GOTO 2000
1200 '----- second order branches -----
    XP = XP + XL: YP2 = YP - YL5
```

```
      FOR NN = SNLON - 1 TO 1 STEP -1
        XP2 = XP
        NOL1 = SUBR(SS, SRLON, n + NN + 1)
        IF k3(SUBR(SS, SRLON, n + NN)) = 1 THEN 1240
        FOR JJ = 1 TO k3(SUBR(SS, SRLON, n + NN))
          NODE = k4(SUBR(SS, SRLON, n + NN), JJ)
          IF NODE = NOL1 THEN 1230
          YP = YP + YL
1210      GOSUB 3000
          IF k3(NODE) = 0 THEN XP = XP2: GOTO 1230
          NODE = k4(NODE, 1): XP = XP - XL: GOTO 1210
          XP = XP2
1230    NEXT JJ
        LINE (XP2 + XL, YP - YL5)-(XP2 + XL, YP2)
1240    XP = XP + XL
      NEXT NN
      RETURN
2000 '-------------------- Rain ------------------------------
      PZ1 = RECX(0) + XL05 - 1: PZ2 = RECY(0)
      PZ3 = RECX(0) + XL65 + 1: PZ4 = RECY(0) + YL4 - 1
      FOR i = 0 TO 6
        PSET (RECX(0) + XL05 + i * XL1, RECY(0)), 3
        DRAW "D=" + VARPTR$(YL1) + "BD=" + VARPTR$(YL1) + "BD=" + VARPTR$(YL1)
      NEXT i
      'LINE (PZ1, PZ2)-(PZ3, PZ4), 2, B
      GET (PZ1, PZ2)-(PZ3, PZ4), RAIN1
      PUT (PZ1, PZ2), RAIN1
      FOR i = 0 TO 6
        PSET (RECX(0) + XL05 + i * XL1, RECY(0)), 3
        DRAW "BD=" + VARPTR$(YL1) + "D=" + VARPTR$(YL1) + "BD=" + VARPTR$(YL1)
      NEXT i
      'LINE (PZ1, PZ2)-(PZ3, PZ4), 2, B
      GET (PZ1, PZ2)-(PZ3, PZ4), RAIN2
      PUT (PZ1, PZ2), RAIN2
      FOR i = 0 TO 6
        PSET (RECX(0) + XL05 + i * XL1, RECY(0)), 3
        DRAW "BD=" + VARPTR$(YL1) + "BD=" + VARPTR$(YL1) + "D=" + VARPTR$(YL1)
      NEXT i
      'LINE (PZ1, PZ2)-(PZ3, PZ4), 2, B
      GET (PZ1, PZ2)-(PZ3, PZ4), RAIN3
      PUT (PZ1, PZ2), RAIN3
      FOR i = 0 TO KLM - 1: PUT (RECX(i), RECY(i)), RAIN1: NEXT i
2500  a$ = INKEY$
      IF a$ <> "" THEN 9999
      FOR i = 0 TO KLM - 1
        PUT (RECX(i), RECY(i)), RAIN1: PUT (RECX(i), RECY(i)), RAIN2
      NEXT i
      FOR i = 0 TO KLM - 1
        PUT (RECX(i), RECY(i)), RAIN2: PUT (RECX(i), RECY(i)), RAIN3
      NEXT i
      FOR i = 0 TO KLM - 1
        PUT (RECX(i), RECY(i)), RAIN3: PUT (RECX(i), RECY(i)), RAIN1
      NEXT i
      GOTO 2500
      FOR i = 0 TO KLM - 1
        PUT (RECX(i), RECY(i)), RAIN1: PUT (RECX(i), RECY(i)), RAIN2
      NEXT i


9999    ERASE L2, TEMP, SEQ, SUBR, TAL, NOL
        ERASE RECX, RECY, RAIN1, RAIN2, RAIN3, p
        ERASE i1, i2, k1, k2, k3, k4

        REDIM qin(140, 100), qout(140, 100)
        REDIM overlnd(70) AS overmod, aqui(70) AS aquimod, pipe(20) AS pipemod
        REDIM unit(140, 2), con(140, 10), ovflo(140, 8), order(2, 140)
        REDIM over(140, 100), rain(100)
```

```
      REDIM trapchan(20) AS trapmod, storage(3) AS stormod
      REDIM compchan(5) AS compmod, x(5, 10), y(5, 10), n(5, 5), segno(5, 5,
2)

      file1$ = "b:temp.skm"

      CALL datain1(file1$)

      KILL "b:temp.skm"

    tcode% = 3

    SCREEN 0
    WIDTH 80

    CLS
    COLOR 0, 2
    LOCATE 11, 28
    PRINT "Chaining editor subprogram"
    COLOR 3, 0

    CHAIN "editorf"

    END
3000 '------------ Subroutine for Drawing Modules in Series ---------------
     FOR II = 1 TO nall
       IF i1(II) = NODE THEN num1 = typ(II): GOTO 3010
     NEXT
3010 IF num1 = 1 THEN 3020
     '------ green background square ------
     LINE (XP + XL2, YP - YL5)-(XP + XL8, YP - YL2), 1, BF: GOTO 3030
3020 '------ green parallelepipidum ------
     PX1 = XP + XL05: PX2 = XP + XL65
     PX3 = XP + XL85: PX4 = XP + XL25
     PY1 = YP - YL2: PY3 = YP - YL6
     LINE (PX1, PY1)-(PX2, PY1), 1: LINE (PX2, PY1)-(PX3, PY3), 1
     LINE (PX3, PY3)-(PX4, PY3), 1: LINE (PX4, PY3)-(PX1, PY1), 1
     PAINT (XP + XL5, YP - YL5), 1, 1
3030 IF k3(NODE) = 0 THEN 3040
     '------ yellow line on the left ------
     LINE (XP, YP - YL5)-(XP + XL2 - 1, YP - YL5)
     '------ yellow line on the right ------
3040 LINE (XP + XL8 + 1, YP - YL5)-(XP + XL - 1, YP - YL5)
     '------ green arrows ------
     PSET (XP + XL - 1, YP - YL5), 1: DRAW "NH2;G2"
     IF num1 <> 1 THEN 3050
     '------ catchment ------
     RECX(KLM) = XP + XL2: RECY(KLM) = YP - YL: KLM = KLM + 1: GOTO 4040
3050 IF num1 <> 2 THEN 3060
     '------ pipeline ------
     IF XL < (5 / 8) * YL THEN R = .25 * XL ELSE R = .25 * YL
     CIRCLE (XP + XL5, YP - YL5), R, 2
     PSET (XP + XL5, YP - YL5), 2
     DRAW "NL=" + VARPTR$(R) + "NP=" + VARPTR$(R) + "BD=" + VARPTR$(YL1)
     DRAW "P 3,2;"
3060 IF num1 <> 4 THEN 3070
     '------ rectangular channel ------
     PSET (XP + XL3, YP - YL3), 2
     DRAW "NU=" + VARPTR$(YL4) + "R=" + VARPTR$(XL4) + "NU=" + VARPTR$(YL4)
     DRAW "U=" + VARPTR$(YL2) + "L=" + VARPTR$(XL4) + "R=" + VARPTR$(XL2)
     DRAW "BD=" + VARPTR$(YL1) + "P 3,2": GOTO 4040
3070 IF num1 <> 3 AND num1 <> 5 THEN 3080
     '------ channel with sloping sides ------
     LINE (XP + XL3, YP - YL7)-(XP + XL4, YP - YL3), 2
     LINE (XP + XL4, YP - YL3)-(XP + XL6, YP - YL3), 2
     LINE (XP + XL6, YP - YL3)-(XP + XL7, YP - YL7), 2
     LINE (XP + .35 * XL, YP - YL5)-(XP + .65 * XL, YP - YL5), 2
```

```
      PAINT (XP + XL5, YP - YL4), 3, 2
3080  IF num1 <> 6 THEN 4040
      '------ ?aquifer? ------
      LINE (XP + XL3, YP - YL3)-(XP + XL7, YP - YL6), 3, BF
      FOR k3 = 1 TO 3
        PSET (XP + (.2 + k3 * .15) * XL, YP - YL6), 2
        DRAW "NH=" + VARPTR$(YL1) + "NU=" + VARPTR$(YL1) + "NE=" +
VARPTR$(YL1)
      NEXT k3
4040  'continue
4045 '-------------------- Draw Numbers ---------------------------------
5000 NUM = NODE
     a = .1
     PSET (XP + a * XL, YP - .05 * YL), 2
     IF NUM < 1000 THEN 5100
     IF NUM < 9000 THEN 5010 ELSE GOSUB 5590: NUM = NUM - 9000: GOTO 5090
5010 IF NUM < 8000 THEN 5020 ELSE GOSUB 5580: NUM = NUM - 8000: GOTO 5090
5020 IF NUM < 7000 THEN 5030 ELSE GOSUB 5570: NUM = NUM - 7000: GOTO 5090
5030 IF NUM < 6000 THEN 5040 ELSE GOSUB 5560: NUM = NUM - 6000: GOTO 5090
5040 IF NUM < 5000 THEN 5050 ELSE GOSUB 5550: NUM = NUM - 5000: GOTO 5090
5050 IF NUM < 4000 THEN 5060 ELSE GOSUB 5540: NUM = NUM - 4000: GOTO 5090
5060 IF NUM < 3000 THEN 5070 ELSE GOSUB 5530: NUM = NUM - 3000: GOTO 5090
5070 IF NUM < 2000 THEN 5080 ELSE GOSUB 5520: NUM = NUM - 2000: GOTO 5090
5080 GOSUB 5510: NUM = NUM - 1000
5090 a = a + .125
     IF NUM < 100 THEN PSET (XP + a * XL, YP - .05 * YL), 2: GOSUB 5500: a =
a + .125
     IF NUM < 10 THEN PSET (XP + a * XL, YP - .05 * YL), 2: GOSUB 5500: a =
a + .125
5100 IF NUM < 100 THEN 5200
     PSET (XP + a * XL, YP - .05 * YL), 2
     IF NUM < 900 THEN 5110 ELSE GOSUB 5590: NUM = NUM - 900: GOTO 5190
5110 IF NUM < 800 THEN 5120 ELSE GOSUB 5580: NUM = NUM - 800: GOTO 5190
5120 IF NUM < 700 THEN 5130 ELSE GOSUB 5570: NUM = NUM - 700: GOTO 5190
5130 IF NUM < 600 THEN 5140 ELSE GOSUB 5560: NUM = NUM - 600: GOTO 5190
5140 IF NUM < 500 THEN 5150 ELSE GOSUB 5550: NUM = NUM - 500: GOTO 5190
5150 IF NUM < 400 THEN 5160 ELSE GOSUB 5540: NUM = NUM - 400: GOTO 5190
5160 IF NUM < 300 THEN 5170 ELSE GOSUB 5530: NUM = NUM - 300: GOTO 5190
5170 IF NUM < 200 THEN 5180 ELSE GOSUB 5520: NUM = NUM - 200: GOTO 5190
5180 GOSUB 5510: NUM = NUM - 100
5190 a = a + .125
     IF NUM < 10 THEN PSET (XP + a * XL, YP - .05 * YL), 2: GOSUB 5500: a =
a + .125
5200 IF NUM < 10 THEN 5300
     PSET (XP + a * XL, YP - .05 * YL), 2
     IF NUM < 90 THEN 5210 ELSE GOSUB 5590: NUM = NUM - 90: GOTO 5290
5210 IF NUM < 80 THEN 5220 ELSE GOSUB 5580: NUM = NUM - 80: GOTO 5290
5220 IF NUM < 70 THEN 5230 ELSE GOSUB 5570: NUM = NUM - 70: GOTO 5290
5230 IF NUM < 60 THEN 5240 ELSE GOSUB 5560: NUM = NUM - 60: GOTO 5290
5240 IF NUM < 50 THEN 5250 ELSE GOSUB 5550: NUM = NUM - 50: GOTO 5290
5250 IF NUM < 40 THEN 5260 ELSE GOSUB 5540: NUM = NUM - 40: GOTO 5290
5260 IF NUM < 30 THEN 5270 ELSE GOSUB 5530: NUM = NUM - 30: GOTO 5290
5270 IF NUM < 20 THEN 5280 ELSE GOSUB 5520: NUM = NUM - 20: GOTO 5290
5280 GOSUB 5510: NUM = NUM - 10
5290 a = a + .125
5300 PSET (XP + a * XL, YP - .05 * YL), 2
     IF NUM < 9 THEN 5310 ELSE GOSUB 5590: GOTO 5390
5310 IF NUM < 8 THEN 5320 ELSE GOSUB 5580: GOTO 5390
5320 IF NUM < 7 THEN 5330 ELSE GOSUB 5570: GOTO 5390
5330 IF NUM < 6 THEN 5340 ELSE GOSUB 5560: GOTO 5390
5340 IF NUM < 5 THEN 5350 ELSE GOSUB 5550: GOTO 5390
5350 IF NUM < 4 THEN 5360 ELSE GOSUB 5540: GOTO 5390
5360 IF NUM < 3 THEN 5370 ELSE GOSUB 5530: GOTO 5390
5370 IF NUM < 2 THEN 5380 ELSE GOSUB 5520: GOTO 5390
5380 IF NUM < 1 THEN 5385 ELSE GOSUB 5510: GOTO 5390
5385 GOSUB 5500
5390
```

```
5400 RETURN
5500 DRAW "L2;U4;R2;D4;": RETURN                    'draw  0
5510 DRAW "U4;": RETURN                             'draw  1
5520 DRAW "L2;U2;R2;U2;L2;": RETURN                 'draw  2
5530 DRAW "NL2;U2;NL1;U2;L2;": RETURN               'draw  3
5540 DRAW "U2;NU2;L2;U2;": RETURN                   'draw  4
5550 DRAW "NL2;U2;L2;U2;R2;": RETURN                'draw  5
5560 DRAW "NU2;L2;U2;NR2;U2;R2;": RETURN            'draw  6
5570 DRAW "U4;L2;": RETURN                          'draw  7
5580 DRAW "U2;NL2;U2;L2;D4;R2;": RETURN             'draw  8
5590 DRAW "NL2;U4;L2;D2;R2;": RETURN                'draw  9

REM $STATIC
SUB datain1 (file1$)


      OPEN file1$ FOR INPUT AS #1


      INPUT #1, title$
      INPUT #1, tint, sim.time, rain.time, hyeto.number%, noit%, nomod%
      INPUT #1, p1%, p2%, p3%, p4%, p5%, p6%, newmod%

      FOR i% = 1 TO hyeto.number%
         INPUT #1, rain(i%)
      NEXT i%

      FOR i% = 1 TO nomod%
         INPUT #1, modu(i%).modno
         INPUT    #1,    modu(i%).dsmod,    modu(i%).ofl,    modu(i%).typ,
modu(i%).infmod, modu(i%).p1

         SELECT CASE modu(i%).typ

           CASE 1
             p% = modu(i%).p1
             INPUT #1, overlnd(p%).man, overlnd(p%).slo, overlnd(p%).lng
             INPUT #1, overlnd(p%).wid

           CASE 2
             p% = modu(i%).p1
             INPUT #1, pipe(p%).lng, pipe(p%).diam, pipe(p%).slo, pipe(p%).man

           CASE 3
             p% = modu(i%).p1
             INPUT #1, trapchan(p%).slo, trapchan(p%).man, trapchan(p%).lng
             INPUT #1, trapchan(p%).wid, trapchan(p%).ss1, trapchan(p%).ss2,
trapchan(p%).mdep

           CASE 4
             p% = modu(i%).p1
             INPUT   #1,   storage(p%).typ,   storage(p%).a,   storage(p%).b,
storage(p%).c1
             INPUT  #1,  storage(p%).ccu,  storage(p%).cu,  storage(p%).ccs,
storage(p%).cs
             INPUT #1, storage(p%).depth
             INPUT  #1,  storage(p%).s1,  storage(p%).csp,  storage(p%).sp,
storage(p%).stlev

           CASE 5
             p% = modu(i%).p1
               INPUT    #1,    compchan(p%).slo,    compchan(p%).lng,
compchan(p%).nosegs, compchan(p%).nopts

             FOR j% = 1 TO compchan(p%).nopts
                  INPUT #1, x(p%, j%), y(p%, j%)
             NEXT j%
```

```
            FOR j% = 1 TO compchan(p%).nosegs
                INPUT #1, segno(p%, j%, 1), segno(p%, j%, 2), n(p%, j%)
            NEXT j%

          CASE 6
            p% = modu(i%).p1
            INPUT    #1,    aqui(p%).slo,    aqui(p%).wid,    aqui(p%).lng,
aqui(p%).depth
            INPUT #1, aqui(p%).wtl
            INPUT    #1,    aqui(p%).sorp,    aqui(p%).perm,    aqui(p%).imc,
aqui(p%).por

          END SELECT

      NEXT i%

  CLOSE #1

END SUB

SUB dataout1 (file1$)


      OPEN file1$ FOR OUTPUT AS #1

      WRITE #1, title$
      WRITE #1, tint, sim.time, rain.time, hyeto.number%, noit%, nomod%
      WRITE #1, p1%, p2%, p3%, p4%, p5%, p6%, newmod%

      FOR i% = 1 TO hyeto.number%
           WRITE #1, rain(i%)
      NEXT i%

      FOR i% = 1 TO nomod%
           WRITE #1, modu(i%).modno
           WRITE    #1,    modu(i%).dsmod,    modu(i%).ofl,    modu(i%).typ,
modu(i%).infmod, modu(i%).p1

          SELECT CASE modu(i%).typ

             CASE 1
               p% = modu(i%).p1
               WRITE #1, overlnd(p%).man, overlnd(p%).slo, overlnd(p%).lng
               WRITE #1, overlnd(p%).wid

             CASE 2
               p% = modu(i%).p1
               WRITE   #1,   pipe(p%).lng,   pipe(p%).diam,   pipe(p%).slo,
pipe(p%).man

             CASE 3
               p% = modu(i%).p1
               WRITE#1,trapchan(p%).slo,trapchan(p%).man,trapchan(p%).lng
                WRITE    #1,    trapchan(p%).wid,    trapchan(p%).ss1,
trapchan(p%).ss2, trapchan(p%).mdep

             CASE 4
               p% = modu(i%).p1
               WRITE  #1,  storage(p%).typ,  storage(p%).a,  storage(p%).b,
storage(p%).cl
               WRITE #1, storage(p%).ccu, storage(p%).cu, storage(p%).ccs,
storage(p%).cs
               WRITE #1, storage(p%).depth
               WRITE #1, storage(p%).sl, storage(p%).csp, storage(p%).sp,
storage(p%).stlev

             CASE 5
```

```
                p% = modu(i%).p1
                 WRITE    #1,    compchan(p%).slo,    compchan(p%).lng,
compchan(p%).nosegs, compchan(p%).nopts

                FOR j% = 1 TO compchan(p%).nopts
                    WRITE #1, x(p%, j%), y(p%, j%)
                NEXT j%

                FOR j% = 1 TO compchan(p%).nosegs
                    WRITE #1, segno(p%, j%, 1), segno(p%, j%, 2), n(p%, j%)
                NEXT j%

              CASE 6
                p% = modu(i%).p1
                WRITE   #1,   aqui(p%).slo,   aqui(p%).wid,   aqui(p%).lng,
aqui(p%).depth
                WRITE #1, aqui(p%).wtl
                WRITE   #1,   aqui(p%).sorp,   aqui(p%).perm,   aqui(p%).imc,
aqui(p%).por

          END SELECT

     NEXT i%

     CLOSE #1

END SUB
```