

CANADA, OCEAN and AQUATIC SCIENCES, Central Region



Environment
Canada

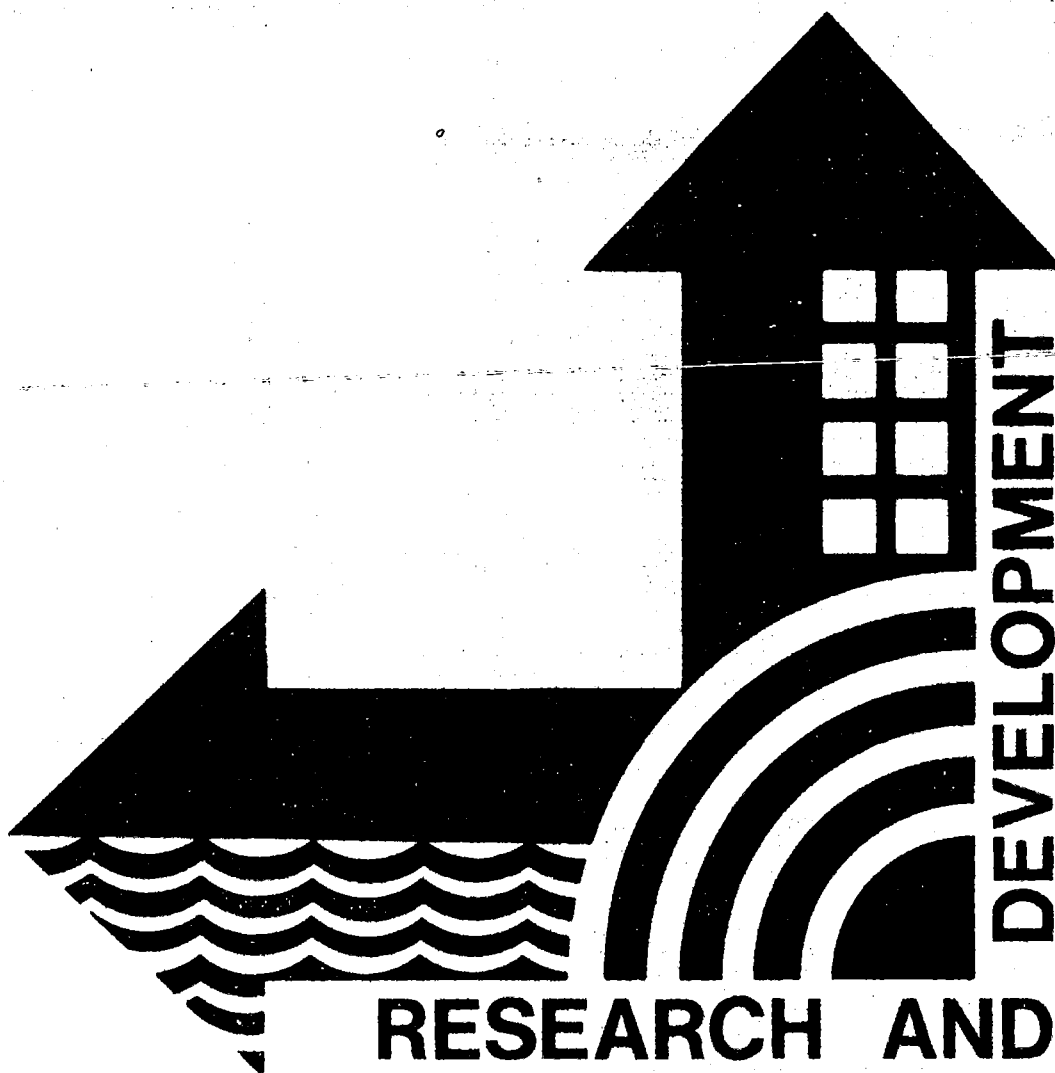
F + M.S.

O. + A.S.

MANUSCRIPT REPORT SERIES No. 1

UNSTEADY FLOW IN NETWORKS OF OPEN CHANNELS

L. R. MUIR



GB
651
M361
No. 1

OCEAN AND AQUATIC SCIENCES
CENTRAL REGION
CANADA CENTRE FOR INLAND WATERS
BURLINGTON, ONTARIO

UNSTEADY FLOW IN NETWORKS
OF
OPEN CHANNELS

by
L.R. Muir

MANUSCRIPT REPORT SERIES NO. 1

1975
OCEAN & AQUATIC SCIENCES
CENTRAL REGION

ABSTRACT

A method is presented for solving problems of one dimensional unsteady flow in arbitrary networks of open channels. The method uses a four point implicit weighted finite difference approximation to the shallow water, or St. Venant, equations. A stability analysis of the finite difference scheme for the linearized equations shows that the weighting factor is very important for stability and convergence. The system has second-order accuracy and conserves mass when the proper weighting factor is chosen. A complete description of an algorithm to solve any set of non-linear algebraic equations is presented which is useful for many other problems besides this particular application.

The computer model is used to solve three different types of problems. The computer model gives good results for unsteady and steady state flow problems. A complete users guide and a listing of the Fortran program is included.

ACKNOWLEDGMENTS

The author wishes to express his thanks to "Operation Preparedness" for instigating this project; to Dr. A.A. Smith of McMaster University, and Dr. S.J. Prinsenberg and Mr. R. Boulden of Ocean and Aquatic Sciences, Central Region, Research and Development Division, for critically reviewing the manuscript and providing many helpful comments; to Mrs. M. Kimmett, Mrs. V. Leroux and Mrs. J. Fiddes for typing the various drafts; and to Mr. R. Raz and Mr. R. Adamson for making many of the computer runs. Any errors or omissions that remain are solely the responsibility of the author.

TABLE OF CONTENTS

Abstract	iii
Acknowledgments	v
Table of Contents	vii
List of Figures	viii
List of Tables	ix
List of Symbols	x
 CHAPTER 1 INTRODUCTION	 1
1.1 Types of Problems	1
1.2 Outline of Research Program	2
1.3 Scope of This Report	5
 CHAPTER 2 THE NETWORK AS A GRAPH	 7
 CHAPTER 3 THE IMPLICIT METHOD	 11
 CHAPTER 4 EQUATIONS OF MOTION	 17
4.1 Continuity Equation	17
4.2 Momentum Equation	19
4.3 Junction Equations	21
 CHAPTER 5 FINITE-DIFFERENCE SCHEME	 25
5.1 Continuity Equation	26
5.2 Momentum Equation	27
 CHAPTER 6 NUMERICAL PROPERTIES OF THE FINITE-DIFFERENCE SCHEME	 31
6.1 Stability	31
6.2 Approximation	35
6.3 Conservation of Mass	38
6.4 Numerical Results	39
 CHAPTER 7 SOLUTION OF EQUATIONS	 41
7.1 Newton's Method	41
7.2 Brown's Algorithm	42
7.3 Matrix Formulation of Brown's Algorithm	46
 CHAPTER 8 APPLICATIONS	 49
8.1 Stoker's Problem	49
8.2 Network Problem	53
8.3 St. Clair River	57
 CHAPTER 9 SUMMARY AND CONCLUSIONS	 65
9.1 Summary	65
9.2 Advantages and Disadvantages	66
9.3 Conclusions	67
 References	 69
Appendix 1 User's Manual for MOD*	71
Appendix 2 Program Listing	93

LIST OF FIGURES

	<u>Page</u>
Figure 1 A River Network and its Graph Representation	8
Figure 2 Splitting of a Node	9
Figure 3 The X-T Solution Plane	12
Figure 4 An Interior Node	14
Figure 5 A Junction Node	14
Figure 6 Definition Sketch - An Irregular Channel	16
Figure 7 A Convergent Junction	22
Figure 8 The X-T Solution Plane Showing the Weighting Factor	24
Figure 9 Schematic Plan of the Ohio-Mississippi Junction for Stoker's Problem	50
Figure 10 Boundary Conditions on the Ohio River	51
Figure 11 Water Level Profiles for Stoker's Problem	52
Figure 12 Schematic Plan of Network Used for Example 2	54
Figure 13 Chart of St. Clair River System	58
Figure 14 Computed vs. Observed Water Levels	63

LIST OF TABLES

	<u>Page</u>
Table 1 Geometrical Data for Network Problem	55
Table 2 Results for Network Problem	56
Table 3 Geometrical Data for St. Clair Problem	60
Table 4 Observed and Computed Water Levels for the St. Clair River	62

LIST OF SYMBOLS

a	- Co-efficient defined by equation 6.12
b	- Topwidth of conveyance section of channel;
b	co-efficient defined by equation 6.12
b_s	- Topwidth of storage section of channel
b_N^i	- Function defined by equation 7.7
e	- Base of natural logarithms (2.718 . . .)
g	- Acceleration due to gravity
g_i	- Function defined by equation 7.7
h	- Depth of channel invert below arbitrary horizontal datum, or
h	- In Chapter 6, a small perturbation in depth
h^*	- Total depth of flow
	- In Chapter 6, an exact solution of the difference equations
h_{ij}	- Matrix defined by equation 7.11
i	- Index
i	- Imaginary unit, $\sqrt{-1}$, in Chapter 6
j	- Index
k	- Linearized friction parameter defined by equation 6.2
m	- Index in space dimension
n	- Index in time dimension
q	- Lateral inflow per unit length
t	- Time
u	- Velocity
u^*	- In Chapter 6, an exact solution of the difference equations
v	- Small perturbation in velocity
x	- Distance along the channel
z	- water level above datum
A	- Cross-sectional area
C	- Chezy friction factor
E, E_1, E_2	- Truncation error
F	- Friction force
F_x	- External force in X-direction
H	- Matrix defined by equation 7.11
H_o	- Mean depth of flow
$J(f)$	- Jacobian Matrix
K	- Arbitrary function of space and time

N	- Number of nodes in a network
O	- "Order of magnitude"
P	- Hydrostatic pressure
P_b	- Boundary pressure
P_r	- Wetted perimeter
Q	- Volume rate of flow
R	- Hydraulic radius
S_E	- Slope of energy gradient
V_o	- Mean velocity
DV	} - Co-efficients defined by equation 5.5
DZ	
HN	
HT	
HTS	
VP	
ZP	
ZO	
β	- Wave frequency
Δm	- Mass of a fluid element
Δx	- Computational distance step
Δt	- Computational time step
δh	- Perturbation error in depth
δu	- Perturbation error in velocity
ξ	- small positive number
η	- Correction factor for Bernoulli term
θ	- Finite-difference weighting factor
λ	- Stability co-efficient
π	- Constant (3.14159 . . .)
ρ	- Density
ϕ	- Arbitrary function
σ	- Wave number
ψ	- Average frictional shear stress

CHAPTER 1

1.0 INTRODUCTION

The prediction of stage, discharge and other flow characteristics due to unsteady flow in open channels has been of interest to engineers for many years. The traditional methods of flow profile analysis and backwater calculation have been dealt with by a variety of numerical and semi-graphical procedures [9, 17]. These methods are based on rather simple, but ingenious assumptions and a variety of empirically developed laws relating to fluid motion. These methods have given "reasonably" accurate answers to some of the practical problems encountered in engineering, but their neglect of acceleration and their general inability to deal with unsteady flow restrict these methods rather severely.

On the other hand, the subject of fluid mechanics has always been more or less mathematically rigorous, but the resulting equations of motion have not been amenable to solution in any practical manner. Since the 1930's, however, some of the methods of fluid mechanics have been applied to practical problems. This successful application has generally been at the cost of ignoring the inherent non-linearities in fluid flow problems and by making certain assumptions which were not always justified.

The development of the digital computer since the 1950's has produced a revolution in open channel hydraulics. The intractability of the equations, produced from the principles of fluid mechanics, to analytical solution is no longer a hindrance, since the digital computer is able to perform the numerical integration of these equations. There is a large body of literature dealing with the use of the digital computer in problems of open channel flow.

1.1 TYPES OF PROBLEMS

Problems of open channel flow may be broadly classed as one, two or three dimensional problems, which respectively increase the order of generality and complexity. Three dimensional considerations are usually necessary when dealing with long wide estuaries with a highly stratified density structure. Two dimensional considerations [18] are usually only necessary in dealing with broad estuaries or

rivers, wide straits or gulfs. In these problems the effect of the earth's rotation and the effect of wind stress on the circulation may be quite important.

Traditionally problems in open channel flows have been dealt with as one dimensional problems and it is for this class of problems that the computer model in this report is designed. This limitation to one dimension is not as confining as it may appear. Examples of one dimensional open channel flow problems which may be encountered are:

- Determination of mean velocities in an open channel when the water levels are known.
- Effects of storm surges in river systems.
- Prediction of tidal elevations in estuaries.
- Flood routing problems.
- Prediction of the effects on flow characteristics of changes in the geometry of waterways.
- Determination of the flow in various branches of a network.

The computer model developed in this report will deal with all of the above types of problems.

1.2 OUTLINE OF RESEARCH PROGRAM

This investigation is concerned with the one dimensional mathematical modeling of unsteady flow in networks of canals, rivers and estuaries. The analysis will allow the computation of water elevation and velocity in any network of open channels to which the following assumptions are applicable:

1. Flow is physically possible.
2. Flow is entirely subcritical (i.e. The Froude number is less than 1.0).
3. Flow is one dimensional.
4. Appropriate boundary conditions are available.
5. The section geometry of the channels is fixed (i.e. no deposition or scouring occurs).

Flow in open channels can be described by two equations, one expressing the conservation of mass (the continuity equation) and one expressing the conservation of momentum in the longitudinal direction (the momentum equation). In general terms these equations form a set

of non-linear partial differential equations of the hyperbolic type. Depending upon the assumptions made there are various methods which are available for the solution of these equations. These methods may be grouped as follows:

1.2.1. Analytical Methods

These methods are characterized by the extremely high level of mathematical ability required to solve extremely simplified problems in fluid dynamics. Fluid motion is usually represented by the linear superposition of several harmonic functions of time. Section geometry is usually specified by a simple mathematical function such as a circle or an ellipse and the change in section geometry in the longitudinal direction is usually specified by an exponential function. The ultimate practicality of this method is limited by the necessity to linearize the equations of motion and by the necessity to simplify the section geometry to a form that may be described by a simple function. Although previous researchers have expended much time on these solutions in the past, their usefulness for practical problems is very limited.

1.2.2. Method of Characteristics

In this method, the equations of motion are combined and solutions of the "characteristic" equation are sought by considering the propagation of small disturbances from an initial state. Development in this method is largely due to Dutch engineers and mathematicians. The method is well suited to problems in which an abrupt surface transition or critical sections appear (such as the development of a tidal bore). The method of solving the characteristic equations, however, may be fairly difficult. If friction is neglected, analytical or graphical solutions are relatively simple to obtain.

In most practical applications some form of finite-difference method must be employed and then the disadvantage of this method lies in the highly irregular grid formed by "characteristics".

The alternatives given by the irregular grid are to either accept the grid and interpolate for the final results or to interpolate at each node to find points from which characteristics emerge to intersect at the desired location. In either case, an unwieldy interpolation process may be avoided by using finite-difference methods directly from the partial differential equations.

1.2.3. Finite-Difference Methods

These methods use finite-difference approximations to the partial derivatives appearing in the equations of motion and solve the resulting system of algebraic equations. It is the direct finite-difference methods that hold the most promise for the solution of engineering problems when a digital computer may be used to solve large systems of algebraic equations. In this case, "direct" means the conversion of the partial differential equations to a finite-difference formulation, without use of the characteristic equations. Two recent reviews of the various available methods are given in [19, 22].

There are two finite-difference methods, and the distinction between them lies in the method in which the finite-differences are formulated and resulting methods for the solution of the equations. The implicit method requires that all of the equations be solved simultaneously in order to advance the solution one time step. The explicit method proceeds down the open channel solving only one equation at one time. There are a large number of finite-differencing schemes available for use with each method.

Because of the bookkeeping and equation solving requirements of the implicit scheme, the explicit method is much simpler to use; however, the explicit schemes are restricted in the size of the computational time step required to ensure a stable computational procedure. Numerical stability is achieved when small errors introduced in the computation diminish rather than increase in magnitude with succeeding computations. If too large a time step is used, the true solution to the equations may well be completely masked by the errors. The restriction in Δt is given by the well known Courant condition, which is:

$$\Delta t < \frac{\Delta x}{u + \sqrt{g A/B}}$$

where, B = width of water surface; A = cross-sectional area; g = acceleration due to gravity; u = velocity; and Δx = the distance interval used. If friction is important, Garrison et al [16] have shown that the maximum Δt may be further limited by the following stability criterion:

$$\Delta t \leq \left[\frac{\Delta x}{u + \sqrt{g A/B}} \right] \left[1 - \frac{gn^2 |u| \Delta t}{2.2 (A/B)^{4/3}} \right]$$

where n is the Manning friction factor.

Examination of the stability criteria will show that, for typical river applications, time steps on the order of a few seconds may be required. For problems in large river systems which may involve tidal cycles or input hydrographs extending over several days, these small time steps cause the explicit method to be very wasteful of computer time.

Two other disadvantages of the explicit schemes are due to the equation solving method. The computations start at one end of the river system and proceed from one end to the other. Therefore, the boundary conditions must be placed at either end of the channel in order that the computations may start. This restriction is not necessary with the implicit method and, as will be shown later, the implicit method allows boundary conditions to be placed anywhere within the system, subject of course to physical constraints.

The other restriction on the explicit method is the difficulty in handling flow in networks. As the computations proceed downstream and reach a junction, some arbitrary decision must be made about how much of the flow enters each branch. The profiles in each branch must then be computed separately and the results, at the end of the branches, compared. If these results are incompatible, then the computation must start again at the top of the branch and this procedure iterated in some fashion until the flows match properly. In the implicit method, this problem does not arise since all of the equations are solved simultaneously.

1.4. SCOPE OF THE REPORT

This report develops all of the theory required to construct and use a numerical model for simulating unsteady flow conditions in networks of open channels.

The first section, although very short, gives a formal method for describing the flow relationships in any network and is necessary for understanding the computer storage scheme. The implicit method is then described and extended from use in single channels to networks of channels. Equations of motion are derived and finite-difference approximations are developed.

The numerical properties of the linearized finite-difference scheme are analyzed by considering the numerical stability by means of

the Von Neuman technique. An expression for the truncation error is developed and the conditions for conservation of mass are investigated. All of these properties are found to depend upon the weighting factor, θ , used in the finite-difference scheme. On the basis of this investigation, a value of θ very close to 0.5 is to be preferred.

A little known method for the solution of systems of non-linear equations is described. It is independent of the form of the equations and deserves wider recognition, since it is applicable to any system of non-linear equations, not only the equations of fluid motion.

Examples are given of the application of the computer model developed from the theory presented in this report. These examples show the versatility of the method. An appendix gives a user's guide to the model in enough detail for anyone to use the model and to modify the model if required for specific applications. Finally a complete Fortran listing of the computer program is given.

CHAPTER 2

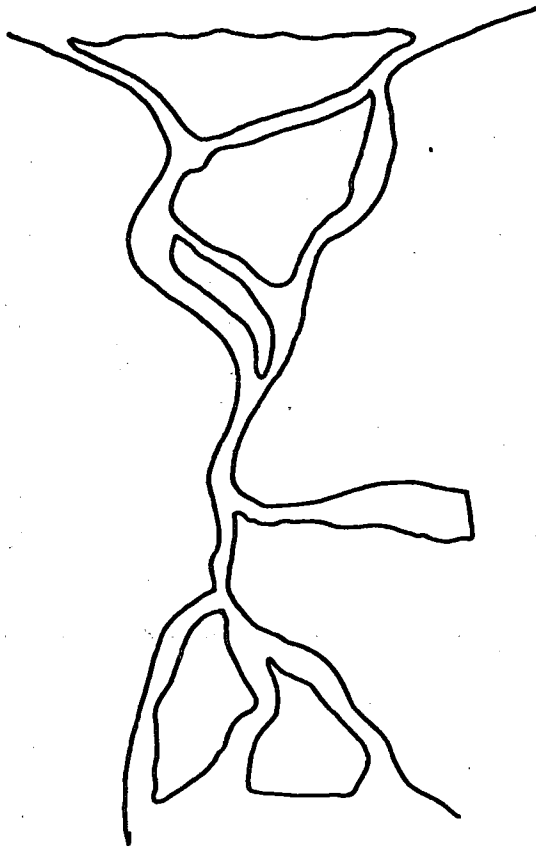
2.0 THE NETWORK AS A GRAPH

The essential features of open channel flow in a network may be illustrated by considering a network as if it were a mathematical entity known as a graph. A graph, in mathematical terms, may be defined as a connected set of lines on a plane surface. The points at which various lines meet or cross are known as nodes and if direction may be determined the graph is said to be directed. The relationships in the graph are purely topological in that distance relationships are not preserved.

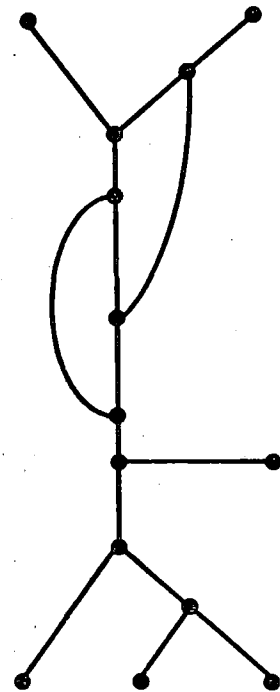
The graph representation of a network of open channels consists of a number of lines called branches representing the elementary reaches of the open channel network and a certain number of nodes, each of which identifies the location at which two or more branches intersect. To make the graph more general and at the same time more applicable to river or estuarine systems, additional nodes are allowed on the boundaries of the graph or at arbitrary locations on the graph. For example, the river system shown in figure 1(a) may be schematized into the graph shown in figure 1(b). The branches of the graph are shown as lines and the nodes are shown as dots in the figure.

In modeling unsteady flow situations, each branch is considered equivalent to an elementary reach in the channel network. The nodes are placed at locations where flow properties are required or are known. At each node it is necessary to have the cross-sectional geometry of the open channel and during the solution of the equations of motion of the system it is assumed that the system parameters will vary continuously between adjacent nodes. Therefore, in the schematization of an open channel network it is necessary to take into account the physical parameters of a system and these considerations are reflected in the additional nodes selected for inclusion on the graph.

It is possible to rearrange the graph so that each node has, at most, three branches connected to it. If more than three branches are connected to a particular node, as in figure 2(a), then the offending node may be split into two, or more, nodes each of which has three branches, as in figure 2(b). The physical distance between the new



(a)



(b)

FIGURE 1: A RIVER NETWORK AND ITS GRAPH REPRESENTATION

nodes will be zero. The purpose of this schematization is to identify flow relationships and to simplify these relationships to an extent which allows simple bookkeeping for a computer program.

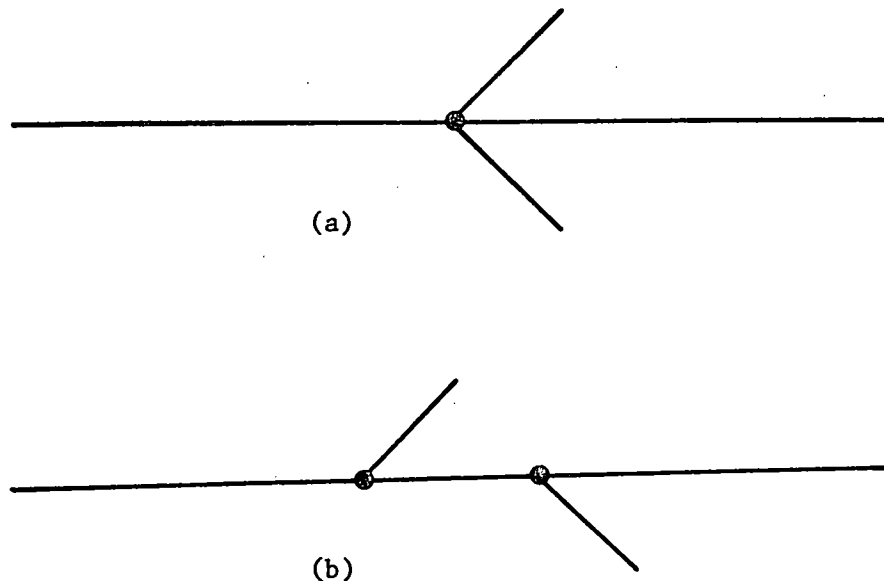


FIGURE 2. SPLITTING OF A JUNCTION NODE

This splitting of nodes allows the identification of exactly four separate types of nodes. Each network can be schematized using only these four types of nodes. They are:

1. A Bounding node, which is connected to only one branch.
2. An Interior node, which is connected to exactly two branches.
3. A Convergent node, which has two branches entering and one branch leaving.
4. A Divergent node, which has one branch entering and two branches leaving.

Types 3 and 4 are known collectively as Junction nodes. The

only difference between them lies in the direction chosen for the graph, and the distinction is made purely for the purpose of easing the computer programming difficulties. In an unsteady flow problem, any given junction node could be either type 3 or type 4 depending upon the direction of flow. This situation is automatically resolved by the computer program and a node is classified according to its type when all flow is in the positive direction. Making this distinction between convergent and divergent nodes eases the computer programming difficulties without making the theoretical treatment any more difficult.

CHAPTER 3

3.0 THE IMPLICIT METHOD

Of the two types of finite-difference schemes available for the solution of the equations of motion, the implicit scheme has been chosen for reasons which have been discussed briefly and will become clearer by the end of this section.

Consider a rectangular grid, not necessarily uniform, on the x - t plane as shown in figure 3. The function value and partial derivatives of a function, ϕ , at $x = x_i + \frac{\Delta x_i}{2}$ and $t = t_j + \frac{\Delta t_{xj}}{2}$

are given by:

$$\phi_i^j = \frac{1}{2}(\phi_i^j + \phi_{i+1}^j + \phi_i^{j+1} + \phi_{i+1}^{j+1})$$

$$\frac{\partial \phi}{\partial t} = \frac{1}{2\Delta t_i} \left[\phi_i^{j+1} - \phi_i^j + \phi_{i+1}^{j+1} - \phi_{i+1}^j \right]$$

$$\frac{\partial \phi}{\partial x} = \frac{1}{2\Delta x_i} \left[\phi_{i+1}^j - \phi_i^j + \phi_{i+1}^{j+1} - \phi_i^{j+1} \right]$$

The above are applied to a single branched river or channel which contains only bounding nodes. At time $t = t_{j+1}$, there are two unknowns for each node or grid point; so that a river system containing N reaches, corresponding to $N+1$ nodes, has $2(N+1)$ unknowns at time t_{j+1} . For each pair of adjacent nodes at time t_{j+1} , a continuity equation and a momentum equation may be written in finite-difference form relating the unknown values at time t_{j+1} to the known values at time t_j . Since there are two equations between each pair of adjacent nodes, there are a total of $2N$ equations in $2(N+1)$ unknowns. The addition of any two additional independent equations relating unknowns will produce a set of $2N+2$ equations with $2N+2$ unknowns which may be solved simultaneously to produce the solutions at time t_{j+1} .

This implicit method may easily be extended for use in a network of channels by considering the interior node shown in figure 4 and the junction node shown in figure 5(a). There are three nodes associated

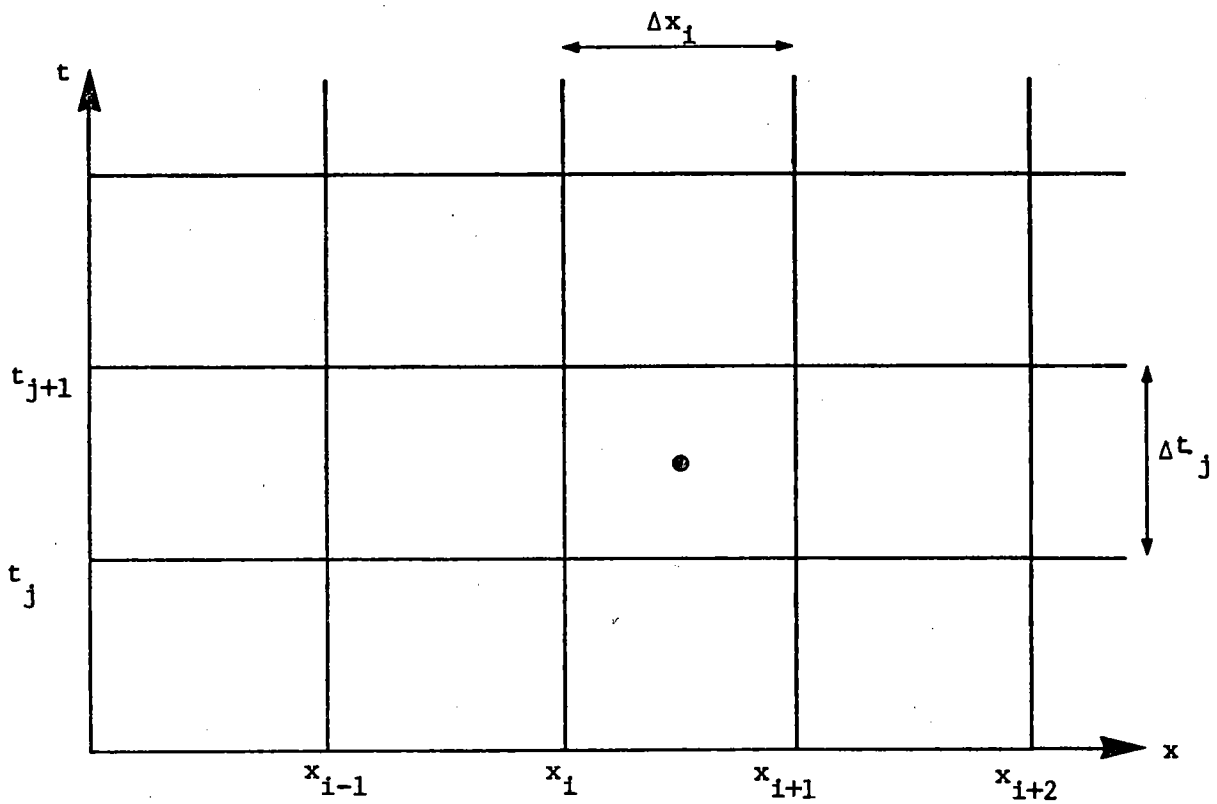


FIGURE 3: THE x - t SOLUTION PLANE

with the interior node shown in figure 4, two bounding nodes and the interior node itself. The system is comprised of two branches for which four equations may be written. The two boundary conditions supply the other two equations necessary for the computation of the six unknown quantities.

Consider the junction node shown in figure 5(a) and split this junction node into three nodes as shown in figure 5(b). There are now six nodes associated with the original junction node and hence twelve unknowns. The three branches provide six equations, the three boundary conditions provide an additional three equations giving a total of nine equations for the twelve unknowns.

The three remaining equations necessary for a solution are provided fairly easily. At any junction, a continuity equation will relate the amount of water entering the junction to the amount of water leaving the junction. This continuity equation provides one equation. Noting that the actual distance between the junction nodes is zero will provide two additional equations since the water level at the three "junction nodes" must be equal. Hence, there are a total of twelve equations in twelve unknowns and the system of equations is solvable.

Two advantages of the implicit method are immediately apparent at this point. The first is that the boundary conditions do not necessarily have to be specified at the bounding nodes. For the system of equations to be solvable, it is only necessary to have the requisite number of independent additional equations. As long as the boundary equations cannot be derived from the rest of the equations, they are mathematically necessary and sufficient for a solution. At any time step, then, any of the following combinations would give sufficient boundary conditions for a solution in a single branched channel:

1. any two water levels, at different locations
2. any two velocities, at different locations
3. any height and a discharge
4. any velocity and a discharge
5. any height and a stage discharge curve
6. any velocity and a stage discharge curve
7. any discharge and a stage discharge curve



FIGURE 4: AN INTERIOR NODE

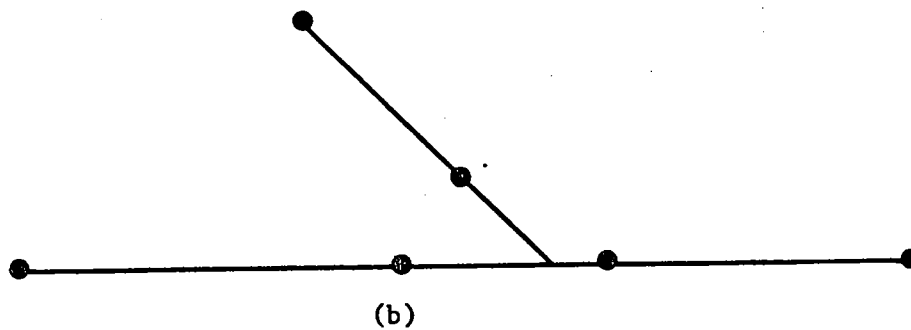
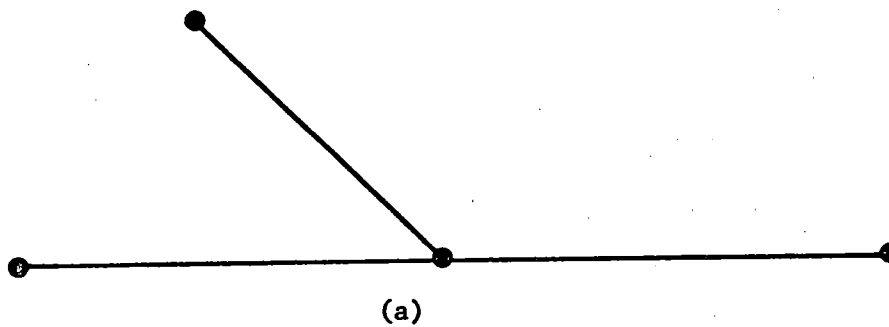


FIGURE 5: A JUNCTION NODE

The following will not give sufficient boundary conditions since the system of equations would not be independent:

1. any two discharges
2. any two stage discharge curves

It should be noted that the boundary equations must be expressed in terms of the same variables that are used for the momentum and continuity equations. This, however, causes no difficulty whatsoever, since the discharge is always an explicit function of velocity and cross-sectional area is always a function of the water level. Using these additional relationships, it is always possible to express the boundary conditions in terms of the appropriate variables.

Except for two cases, the simple algebraic requirement for two additional, independent boundary equations will also satisfy the physical requirements for boundary conditions. The first exception is the occurrence of a transition section in the channel. In almost all versions of the equations of motion, this transition will appear as a discontinuity in the water surface profile. Therefore, to handle cases such as this, additional equations would be required to describe the transition section itself.

The second exception is due to the physical process of measurement. If, in a very long channel, the boundary conditions are specified very close together, the inaccuracies in measurement may well be amplified both up and downstream. This, however, is a simple case of ill-conditioning and so care must be taken to ensure that the specification of boundary conditions is accurate enough to provide meaningful answers.

The second advantage of the implicit method is in the computation of flows around islands and in parallel channels. By solving all of the equations simultaneously, the flow in the various channels is derived automatically. The explicit method, which starts the computations at one end of a channel and proceeds upstream, does not have this advantage. In the explicit method various complicated iterative techniques are required in order to balance the flow in various branches. The result may be a savings in computation time and difficulty; if a satisfactory method is found for the solution of large systems of simultaneous non-linear equations.

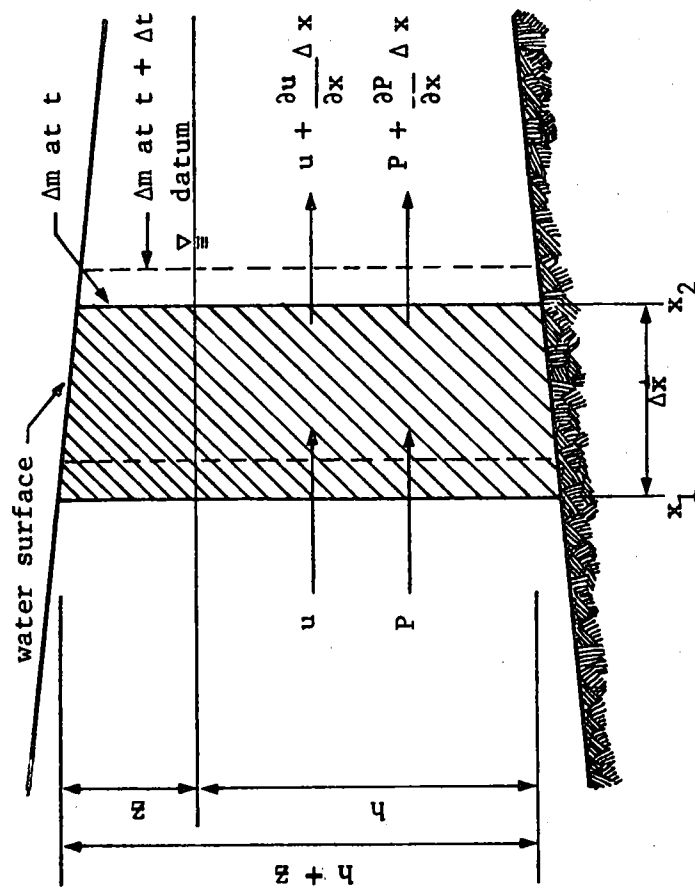


FIGURE 6: DEFINITION SKETCHES FOR AN IRREGULAR CHANNEL

CHAPTER 4

4.0 EQUATIONS OF MOTION

The equations of motion for one dimensional flow are derived in this section. These equations, known as the shallow water wave equations, or the St. Venant equations, consist of continuity and momentum equations for unsteady, non-uniform flow in non-prismatic open channels.

The schematization of a section of open channel is shown in figure 6. The x co-ordinate is measured horizontally along the longitudinal axis of the channel and z is the distance from an arbitrary horizontal datum to the water surface. The flow is assumed to be one dimensional and hence: channel curvature and Coriolis effects are neglected; the transverse water surface is horizontal; the density is homogeneous; and hydrostatic pressure prevails at all points in the channel.

Development of the equations of motion may proceed by a number of methods. The conservation of mass and momentum may be formulated by the material (Lagrangian) method or by the control volume (Eulerian) method. In the material method, the equations are derived by considering the motion of a given mass of fluid, Δm , for a small increment of time, Δt , in the vicinity of a fixed section. In the control volume method, the equations are derived by considering the flux of mass and momentum through a control volume fixed in space.

Although the control volume method provides a much more elegant method for deriving the equations of motion, the material method will be used here since it gives more insight into the physical processes involved.

4.1 CONTINUITY EQUATION

The continuity equation may be derived from the law of conservation of mass of a fluid element, Δm . Assuming that there is no transport in the storage section of the channel and that no lateral inflow or outflow is present the total time derivative of the mass in the main channel is zero, that is:

$$\frac{d}{dt} (\Delta m) = \frac{d}{dt} (\rho A \Delta x) = 0$$

where: ρ = density of the water
 $A = bh^* = b(h+z)$
 h^* = hydraulic mean depth
 b = topwidth of conveyance section

Two factors contribute to a change of mass in a moving element. These are: a change due to storage in a shallow portion of the cross-section which is equal to

$$\rho b_s \frac{\partial z}{\partial t} \Delta x$$

where b_s = topwidth of the storage section, and a change due to a lateral inflow or outflow caused by a small tributary or by bank infiltration, which is equal to

$$\rho q \Delta x$$

where q is the lateral inflow per unit of longitudinal length.

Hence, the continuity equation is

$$\frac{d}{dt} (\rho A \Delta x) = \rho b_s \frac{\partial z}{\partial t} \Delta x + \rho q \Delta x \quad (4.1)$$

In terms of partial derivatives, the total derivative is:

$$\frac{d}{dt} = \frac{\partial}{\partial t} + u \frac{\partial}{\partial x}$$

where u is the average velocity at a cross-section so that (4.1) may be written as

$$\Delta x \frac{\partial A}{\partial t} + u \Delta x \frac{\partial A}{\partial x} + A \frac{d}{dt} (\Delta x) + b_s \frac{\partial z}{\partial t} \Delta x - q \Delta x = 0 \quad (4.2)$$

The term $\frac{d}{dt} (\Delta x)$ may be handled as follows:

at time t , $\Delta x = x_2 - x_1$

and at time $t + \Delta t$, $\Delta x' = (x_2 + u \Delta t + \frac{\partial u}{\partial x} \Delta x \Delta t) - (x_1 + u \Delta t)$

hence: $\frac{d}{dt} (\Delta x) = \frac{\Delta x' - \Delta x}{\Delta t} = \frac{\partial u}{\partial x} \Delta x$

and after dividing by Δx , equation (4.2) becomes

$$\frac{\partial A}{\partial t} + u \frac{\partial A}{\partial x} + A \frac{\partial u}{\partial x} + b_s \frac{\partial z}{\partial t} - q = 0 \quad (4.3)$$

Noting that $A = bh^*$

$$\frac{\partial h^*}{\partial t} = \frac{\partial z}{\partial t}$$

$$\text{and } \frac{\partial b}{\partial t} = 0$$

equation 4.3 becomes

$$(b + b_s) \frac{\partial z}{\partial t} + ub \frac{\partial h^*}{\partial x} + uh^* \frac{\partial b}{\partial x} + bh^* \frac{\partial u}{\partial x} - q = 0 \quad (4.4)$$

4.2 MOMENTUM EQUATION

This equation is derived from Newton's Second Law, which states that the time rate of change of momentum is equal to the sum of the external forces acting on the moving fluid element.

Assuming again that there is no transport in the storage section of the channel, the longitudinal momentum of a fluid element Δm is given by

$$(\Delta m)u = (\rho A \Delta x) \frac{Q}{A} = \rho Q \Delta x$$

Newton's Second Law states that

$$\frac{d}{dt} (\rho Q \Delta x) = \rho \Delta x \frac{dQ}{dt} + \rho Q \frac{d}{dt} (\Delta x) = \Sigma F_x \quad (4.5)$$

using the relationships already derived for

$$\frac{d}{dt} \text{ and } \frac{d}{dt}(\Delta x); (4.5) \text{ becomes}$$

$$\rho \Delta x \left(\frac{\partial Q}{\partial t} + u \frac{\partial Q}{\partial x} \right) + \rho Q \frac{\partial u}{\partial x} \Delta x = \Sigma F_x \quad (4.6)$$

The change in momentum flux due to lateral inflow is extensively discussed by Dronkers [11]. There is a change in the longitudinal momentum flux due to flow entering or leaving the main channel from the storage area, but the effect is dependent upon whether the water level is rising or falling and hence whether the lateral inflow is entering or leaving the storage area. It will be assumed in this discussion that lateral flows enter or leave the conveyance section at right angles to the longitudinal momentum flux. The schematization of a channel into conveyance and storage sections is an approximate method for accounting for non-uniform velocity distributions and further refinements to account for these non-uniform velocity distributions are not warranted.

The above argument will only apply when the rate of lateral flow is very small compared to the flow in the main channel. If this rate is large, then the above argument does not apply and additional terms must be added to the momentum equation to account for the loss or gain in momentum due to the lateral inflow or outflow.

The sum of the external forces, ΣF_x , consists of P, the difference in hydrostatic pressure force on the two vertical cross-sections; P_B , the x component of the horizontal pressure due to convergent boundaries of the section; and F, the friction force exerted by the boundaries.

Hence, the three forces are:

$$\begin{aligned}\Sigma F_x &= P - (P + \frac{\partial P}{\partial x} \Delta x) + P_B - F \\ &= - \frac{\partial P}{\partial x} \Delta x + P_B - F\end{aligned}\quad (4.7)$$

The total force due to pressure on a vertical face is

$$P = \int_0^{h^*} \rho g (h^* - z') b' dz'$$

where b' is the channel width at elevation z' .

$$\text{so } \frac{\partial P}{\partial x} = \rho g \frac{\partial z}{\partial x} A + \rho g \int_0^{h^*} (h^* - z') \frac{\partial b'}{\partial x} dz' \quad (4.8)$$

$$\text{where the flow area } A = \int_0^{h^*} b' dz'$$

The boundary pressure force is

$$P_B = \rho g \int_0^{h^*} (h^* - z') \frac{\partial b'}{\partial x} \Delta x dz' \quad (4.9)$$

The average frictional shear stress on the boundary of the fluid element is:

$$\psi_o = \rho g h^* S_E$$

where h^* is the hydraulic mean depth and S_E is the slope of the energy gradient. Hence the frictional force is:

$$F = \psi_o (P_r \Delta x) = \rho g A S_E \Delta x$$

where b is the topwidth of the section.

The slope of the energy gradient may be evaluated from the Chezy equation in which

$$S_E = \frac{u|u|}{C^2 h^*}$$

$$\text{hence } F = \rho g A \Delta x \frac{u|u|}{C^2 h^*} \quad (4.10)$$

substituting into equation (4.7)

$$\Sigma F_x = -\rho g \frac{\partial z}{\partial x} A \Delta x - \rho g A \Delta x \frac{u|u|}{C^2 h^*} \quad (4.11)$$

noting that the integrals from (4.8) and (4.9) cancel each other.

Substituting (4.11) into equation (4.6) and dividing by the product $\rho A \Delta x$, the momentum equation is:

$$\frac{\partial Q}{\partial t} + u \frac{\partial Q}{\partial x} + Q \frac{\partial u}{\partial x} + g A \frac{\partial z}{\partial x} + g A \frac{u|u|}{C^2 h^*} = 0 \quad (4.12)$$

Since:

$$Q = Au = b h^* u = b(h + z)u$$

$$\frac{\partial b}{\partial t} = 0$$

$$\frac{\partial h}{\partial t} = 0$$

$$\frac{\partial h^*}{\partial t} = \frac{\partial z}{\partial t}$$

equation (4.12), after some algebraic manipulation and use of the chain rule, becomes:

$$\frac{\partial u}{\partial t} + \frac{u}{h^*} \frac{\partial z}{\partial t} + 2u \frac{\partial u}{\partial x} + \frac{u^2}{b} \frac{\partial b}{\partial x} + \frac{u^2}{h^*} \frac{\partial h^*}{\partial x} + g \frac{\partial z}{\partial x} + g \frac{u|u|}{C^2 h^*} = 0 \quad (4.13)$$

which is the final form of the momentum equation.

4.3 JUNCTION EQUATIONS

The schematization of a junction has resulted in the generalized junction shown in figure 7. For purposes of this discussion, the junction is considered to be convergent, but the argument could easily be applied to a divergent junction.

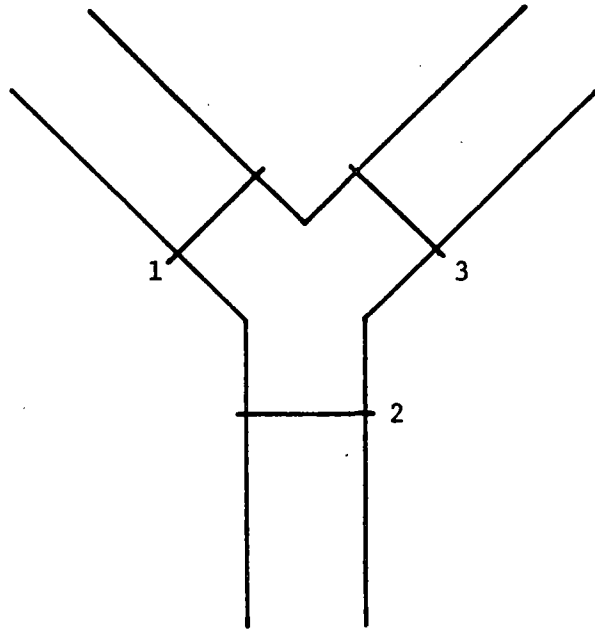


FIGURE 7: A CONVERGENT JUNCTION

The continuity equation is quite simple to write for a junction. Since the nodes are very close together, the storage in the junction may be ignored and so:

$$Q_1 + Q_3 = Q_2 \quad (4.14)$$

That is, all of the water flowing into the junction must flow out.

The conditions for a generalized momentum equation at a junction are very much more involved. If u_1 , u_2 , u_3 are the velocities at nodes 1, 2 and 3 respectively, a difference in water level may occur between the water levels at the various nodes. Since the distances between the nodes are very small, the terms $\frac{\partial Q}{\partial t}$, $Q \frac{\partial u}{\partial x}$ and the resistance term may be neglected in the momentum equation. If energy losses between node 1 and node 2 caused by non-uniformity of flow and acceleration or deceleration are taken into account by means of the factor η , equation (4.12) becomes:

$$\eta u \frac{\partial Q}{\partial x} + gA \frac{\partial z}{\partial x} = 0 \quad (4.15)$$

Then dividing through by A and putting into finite-difference form, (4.15) becomes

$$Z_1 - Z_2 = \frac{\eta}{2g} (u_2^2 - u_1^2) \quad (4.16)$$

and therefore, the energy heads will be equal across the junction.

Unfortunately this argument does not hold in practice. If the velocity at node 2 is smaller than at node 1, a considerable dissipation of energy may take place and η could be considerably smaller than 1. Alternatively, if $u_2 > u_1$, the value of η may be very close to 1. Thus over a tidal cycle, or any unsteady flow condition, the value of η may be a time dependent function and in practice its value may be unknown and very difficult to determine.

The result of this discussion is to indicate the problems in the development of generalized momentum equations for all junctions. If the differences in velocity through a junction are small and if centrifugal accelerations do not play a large part in energy dissipation, then the two momentum equations required for the junction may be written as:

$$Z_1 = Z_2 \quad (4.17)$$

$$Z_1 = Z_3 \quad (4.18)$$

If very accurate values for the water levels and velocities in a junction are required, both the Bernoulli forces and the centrifugal forces must be taken into account.

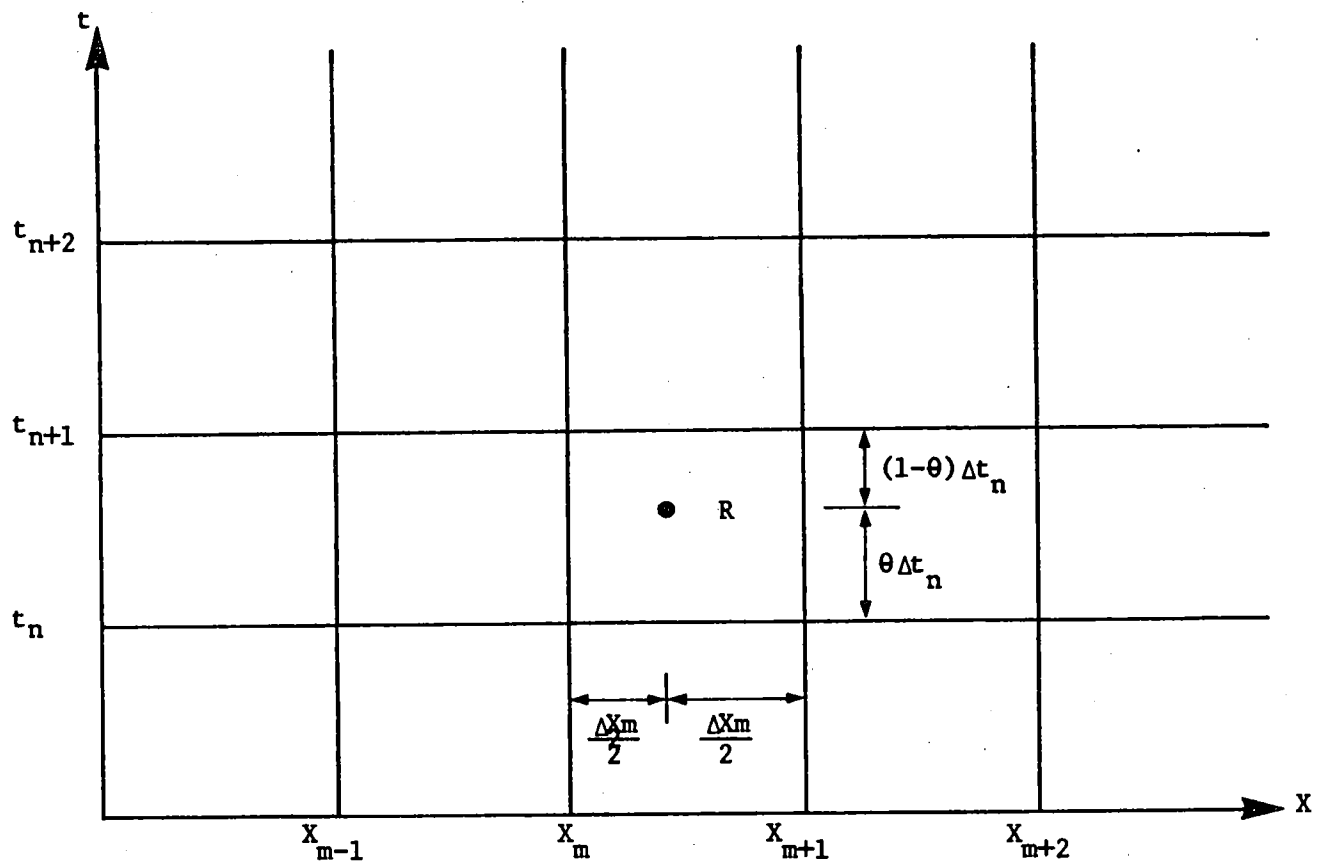


FIGURE 8: THE x - t SOLUTION PLANE SHOWING THE WEIGHTING FACTOR, θ

CHAPTER 5

5.0 FINITE-DIFFERENCE SCHEME

Since there is no known analytic solution to the partial differential equations of flow except for a very few specific cases, some numerical procedure must be found. Of the various implicit finite-difference techniques that have been used, the weighted four point scheme appears to have a number of advantages. Some of these advantages are: that the scheme may be used with a non-uniform distance grid; that convergence and stability may be controlled by varying the weighting factor as explained in chapter 6; and that the values of the unknown z and u are obtained at each grid point.

If we consider a rectangular grid placed on the x - t plane, the locations of points at which solutions of the unknowns are to be obtained are given by the intersections of the lines making up the net. Lines drawn parallel to the t -axis represent nodal points on an open channel and are spaced according to the corresponding nodal spacing on the open channel. They need not be a constant distance apart. Lines drawn parallel to the x -axis represent time steps and need not be a constant distance apart.

Figure 8 shows a grid drawn upon the x - t plane. The function values and partial derivatives of the unknowns will be evaluated at the point R , which is located at $(x_m + \frac{\Delta x_m}{2}, t_n + \theta \Delta t_n)$ where θ represents a weighting factor $0 \leq \theta \leq 1.0$. The function values and the partial derivatives of any unknown, K , may then be expressed in terms of the values at the four grid points surrounding the point R in the following manner.

$$K \approx \theta \left(\frac{K_m^{n+1} + K_{m+1}^{n+1}}{2} \right) + (1-\theta) \left(\frac{K_m^n + K_{m+1}^n}{2} \right) \quad (5.1)$$

$$\frac{\partial K}{\partial t} \approx \frac{1}{2\Delta t} (K_m^{n+1} + K_{m+1}^{n+1} - K_m^n - K_{m+1}^n) \quad (5.2)$$

$$\frac{\partial K}{\partial x} \approx \theta \left(\frac{K_{m+1}^{n+1} - K_m^{n+1}}{\Delta x} \right) + (1-\theta) \left(\frac{K_{m+1}^n - K_m^n}{\Delta x} \right) \quad (5.3)$$

In advancing the solution of the equations from time step t_n to time step t_{n+1} , all of the values of K at time t_n are assumed to be known.

If $\theta = 1$, then the backward implicit scheme used by Dronkers [10] and by Baltzer and Lai [5] is obtained. If $\theta = 0.5$, then the box scheme used by Amien [1,2], Amien and Chu [3], Amien and Fang [4], and Fread [12, 13, 15] is obtained. If $\theta = 0$, then an explicit finite-difference scheme is obtained. The scheme is implicit for all $\theta > 0$. The numerical properties of this scheme are discussed in detail in the following chapter.

5.1 THE CONTINUITY EQUATION

The continuity equation is given as equation (4.4); substituting (5.1), (5.2), (5.3) into this equation yields:

$$\begin{aligned}
& \frac{1}{2} \left[(b+b_s)_m + (b+b_s)_{m+1} \right] \left[\frac{1}{2\Delta t} (z_m^{n+1} + z_{m+1}^{n+1} - z_m^n - z_{m+1}^n) \right] + \\
& + \left[\theta \left(\frac{u_m^{n+1} + u_{m+1}^{n+1}}{2} \right) + (1-\theta) \left(\frac{u_m^n + u_{m+1}^n}{2} \right) \right] \left[\frac{b_m + b_{m+1}}{2} \right] * \\
& * \left[\theta \left(\frac{h_{m+1}^{*n+1} - h_m^{*n+1}}{\Delta x} \right) + (1-\theta) \left(\frac{h_{m+1}^{*n} - h_m^{*n}}{\Delta x} \right) \right] + \\
& + \left[\theta \left(\frac{u_m^{n+1} + u_{m+1}^{n+1}}{2} \right) + (1-\theta) \left(\frac{u_m^n + u_{m+1}^n}{2} \right) \right] \left[\frac{b_{m+1} - b_m}{\Delta x} \right] * \\
& * \left[\theta \left(\frac{h_m^{*n+1} + h_{m+1}^{*n+1}}{2} \right) + (1-\theta) \left(\frac{h_m^{*n} + h_{m+1}^{*n}}{2} \right) \right] + \\
& + \left(\frac{b_m + b_{m+1}}{2} \right) \left[\theta \left(\frac{h_m^{*n+1} + h_{m+1}^{*n+1}}{2} \right) + (1-\theta) \left(\frac{h_m^{*n} + h_{m+1}^{*n}}{2} \right) \right] * \\
& * \left[\theta \left(\frac{u_{m+1}^{n+1} - u_m^{n+1}}{\Delta x} \right) + (1-\theta) \left(\frac{u_{m+1}^n - u_m^n}{\Delta x} \right) \right]
\end{aligned}$$

$$- q = 0$$

(5.4)

This rather unwieldy equation may be simplified by realizing that the only unknowns are the values of z and u with time superscripts of $n + 1$. Therefore, collecting terms

$$\begin{aligned}
HT &= h_m + z_m^n \\
HTS &= h_{m+1} + h_m + (1-\theta) (z_m^n + z_{m+1}^n) \\
HN &= h_{m+1} - h_m + (1-\theta) (z_{m+1}^n - z_m^n) \\
VP &= u_m^n + u_{m+1}^n \\
Z\phi &= (1-\theta) VP \\
DV &= (1-\theta) (u_{m+1}^n - u_m^n) \\
DZ &= (1-\theta) (z_{m+1}^n - z_m^n) \\
ZP &= z_m^n + z_{m+1}^n
\end{aligned} \tag{5.5}$$

substituting these values into (5.4) and dropping the time superscript yields:

$$\begin{aligned}
&\frac{1}{4\Delta t} \left[(b+b_s)_m + (b+b_s)_{m+1} \right] (z_m + z_{m+1} - ZP) + \\
&+ \frac{1}{4\Delta x} [\theta(u_m + u_{m+1}) + Z\phi] [b_m + b_{m+1}] [\theta(z_{m+1} - z_m) + HN] \\
&+ \frac{1}{4\Delta x} [\theta(u_m + u_{m+1}) + Z\phi] [b_{m+1} - b_m] [\theta(z_{m+1} + z_m) + HTS] \\
&+ \frac{1}{4\Delta x} [b_m + b_{m+1}] [\theta(z_m + z_{m+1}) + HTS] [\theta(u_{m+1} - u_m) + DV] \\
&- q = 0
\end{aligned} \tag{5.6}$$

which is the simple form of the continuity equation to be used in the computer program.

5.2 THE MOMENTUM EQUATION

The momentum equation is given as equation (4.13). Substituting (5.1), (5.2) and (5.3) into the momentum equation yields:

$$\begin{aligned}
& \frac{1}{2\Delta t} \left(u_m^{n+1} + u_{m+1}^{n+1} - u_m^n - u_{m+1}^n \right) + \\
& + \frac{\left[\theta \left(u_m^{n+1} + u_{m+1}^{n+1} \right) + (1-\theta) \left(u_m^n + u_{m+1}^n \right) \right]}{\left[\theta \left(h_m^{*n+1} + h_{m+1}^{*n+1} \right) + (1-\theta) \left(h_m^{*n} + h_{m+1}^{*n} \right) \right]} \left[\frac{1}{2\Delta t} \left(z_m^{n+1} + z_{m+1}^{n+1} - z_m^n - z_{m+1}^n \right) \right] + \\
& + 2 \left[\theta \left(\frac{u_m^{n+1} + u_{m+1}^{n+1}}{2} \right) + (1-\theta) \left(\frac{u_m^n + u_{m+1}^n}{2} \right) \right] \left[\theta \left(\frac{u_{m+1}^{n+1} - u_m^{n+1}}{\Delta x} \right) + (1-\theta) \left(\frac{u_{m+1}^n - u_m^n}{\Delta x} \right) \right] + \\
& + 2 \frac{\left[\theta \left(\frac{u_m^{n+1} + u_{m+1}^{n+1}}{2} \right) + (1-\theta) \left(\frac{u_m^n + u_{m+1}^n}{2} \right) \right]^2}{(b_m + b_{m+1})} \left(\frac{b_{m+1} - b_m}{\Delta x} \right) + \\
& + \frac{1}{2} \frac{\left[\theta \left(u_m^{n+1} + u_{m+1}^{n+1} \right) + (1-\theta) \left(u_m^n + u_{m+1}^n \right) \right]^2}{\left[\theta \left(h_m^{*n+1} + h_{m+1}^{*n+1} \right) + (1-\theta) \left(h_m^{*n} + h_{m+1}^{*n} \right) \right]} * \\
& * \left[\theta \left(\frac{h_{m+1}^{*n+1} - h_m^{*n+1}}{\Delta x} \right) + (1-\theta) \left(\frac{h_{m+1}^{*n} - h_m^{*n}}{\Delta x} \right) \right] + \\
& + g \left[\theta \left(\frac{z_{m+1}^{n+1} - z_m^{n+1}}{\Delta x} \right) + (1-\theta) \left(\frac{z_{m+1}^n - z_m^n}{\Delta x} \right) \right] + \\
& + g \left[\theta \left(\frac{u_m^{n+1} + u_{m+1}^{n+1}}{2} \right) + (1-\theta) \left(\frac{u_m^n + u_{m+1}^n}{2} \right) \right] * \\
& * \frac{\left| \theta \left(\frac{u_m^{n+1} + u_{m+1}^{n+1}}{2} \right) + (1-\theta) \left(\frac{u_m^n + u_{m+1}^n}{2} \right) \right|}{C^2 \left[\theta \left(\frac{h_m^{*n+1} + h_{m+1}^{*n+1}}{2} \right) + (1-\theta) \left(\frac{h_m^{*n} + h_{m+1}^{*n}}{2} \right) \right]}
\end{aligned}$$

Making the substitutions (5.5) and dropping the time superscripts gives:

$$\begin{aligned}
& \frac{1}{2\Delta t} (u_m + u_{m+1} - VP) + \frac{1}{2\Delta t} \left[\frac{(\theta(u_m + u_{m+1}) + Z\phi)}{(\theta(z_m + z_{m+1}) + HTS)} (z_m + z_{m+1} - ZP) \right] + \\
& + \frac{1}{\Delta x} \left[(\theta(u_m + u_{m+1}) + Z\phi)(\theta(u_{m+1} - u_m) + DV) \right] + \\
& + \frac{1}{\Delta x} \left[(\theta(u_m + u_{m+1}) + Z\phi) \right]^2 \left[\frac{b_{m+1} - b_m}{b_{m+1} + b_m} \right] + \\
& + \frac{1}{2\Delta x} \left[\frac{(\theta(u_m + u_{m+1}) + Z\phi)}{(\theta(z_m + z_{m+1}) + HTS)} \right]^2 \left[\theta(z_{m+1} - z_m) + HN \right] + \\
& + \frac{g}{\Delta x} [\theta(z_{m+1} - z_m) + DZ] + \\
& + \frac{g_2}{2C^2} \frac{[\theta(u_m + u_{m+1}) + Z\phi] |\theta(u_m + u_{m+1}) + Z\phi|}{[\theta(z_m + z_{m+1}) + HTS]} \tag{5.8}
\end{aligned}$$

which is the momentum equation in finite-difference form.

CHAPTER 6

6.0 NUMERICAL PROPERTIES OF THE FINITE-DIFFERENCE SCHEME

In chapter 4 the partial differential equations of motion of fluid flow in open channels were derived. In chapter 5 a set of finite-difference equations were derived which approximate the partial differential equations. In the limit as the grid size tends to zero, these difference equations become the differential equations. In practice, however, the limit is never taken and the computation does not take place with an infinite number of decimal places. This chapter examines the stability, the computational error, and the convergence of the finite-difference scheme.

6.1 STABILITY

A solution to the set of partial differential equations is stable if and only if numerical errors introduced into the computation are not amplified during the course of computation to produce an unbounded solution.

One technique for investigation of stability was developed by J. Von Neumann. This technique follows the Fourier expansion of a line of errors as time progresses. It is only applicable to linear systems and is based on the hypothesis that linear operators with variable coefficients are stable if and only if all of their localized operators (i.e. where the coefficients are taken as constants), are stable.

The equations of motion of a system are linearized by neglecting certain terms on the basis that their magnitude is relatively small. Considering a broad channel with no lateral inflow, a small perturbation in depth h above a mean depth H_o and velocity u above a mean velocity V_o yields,

$$\frac{\partial h}{\partial t} + H_o \frac{\partial u}{\partial x} = 0 \quad (6.1)$$

$$\frac{\partial u}{\partial t} + g \frac{\partial h}{\partial x} + ku = 0 \quad (6.2)$$

$$\text{where } k = \frac{g V_o n^2}{2.2 H_o^{4/3}}$$

and n = Mannings roughness coefficient.

Substituting the weighted, four point implicit finite-difference scheme given by equations 5.1 to 5.3, into the linearized equations of motion yields,

$$\begin{aligned} & \frac{h_m^{n+1} + h_{m+1}^{n+1} - h_m^n - h_{m+1}^n}{2\Delta t} + \theta H_o \frac{(u_{m+1}^{n+1} - u_m^{n+1})}{\Delta x} + \\ & + (1-\theta) H_o \frac{(u_{m+1}^n - u_m^n)}{\Delta x} = 0 \end{aligned} \quad (6.3)$$

$$\begin{aligned} & \frac{u_m^{n+1} + u_{m+1}^{n+1} - u_m^n - u_{m+1}^n}{2\Delta t} + \theta \left[\frac{g(h_{m+1}^{n+1} - h_m^{n+1})}{\Delta x} + k \frac{(u_m^{n+1} + u_{m+1}^{n+1})}{2} \right] + \\ & + (1-\theta) \left[\frac{g(h_{m+1}^n - h_m^n)}{\Delta x} + k \frac{(u_m^n + u_{m+1}^n)}{2} \right] = 0 \end{aligned} \quad (6.4)$$

as the exact finite-difference scheme.

Since the finite-difference scheme represents a linear system, only one term in the Fourier series expansion of the errors need be considered. Therefore the errors are given by

$$\delta h(x,t) = h^*(x,t) e^{i(\sigma x + \beta t)} \quad (6.5)$$

$$\delta u(x,t) = u^*(x,t) e^{i(\sigma x + \beta t)} \quad (6.6)$$

where δh and δu are the errors in the depth and velocity respectively. These errors are functions of time and space: h^* and u^* are the exact solutions of the difference equations; $i = \sqrt{-1}$, the imaginary unit; $\beta = 2\pi/T$, the wave frequency; $\sigma = 2\pi/L$, the wave number; T = the wave period and; L = the wave length.

These errors may be expressed at the discrete node points on the x - t plane.

$$\begin{aligned} \delta h_m^n &= h_m^* e^{i(\sigma m \Delta x + \beta n \Delta t)} \\ \delta u_m^n &= u_m^* e^{i(\sigma m \Delta x + \beta n \Delta t)} \\ \delta h_{m+1}^{n+1} &= h_m^* e^{i(\sigma(m+1)\Delta x + \beta(n+1)\Delta t)} \end{aligned} \quad (6.7)$$

etc.

The errors are assumed to be perturbations superimposed on the exact solution of the exact finite-difference equations. Therefore the solution provided by the system at a point (m,n) would not be $(h^*)_m^n$ but would be

$$(h^*)_m^n + (h^*)_n^m e^{i(\sigma m \Delta x + \beta n \Delta t)} \quad (6.8)$$

To obtain an expression involving the error terms, all the terms in the form of equation (6.8) are substituted into equations (6.3) and (6.4) and then equations (6.3) and (6.4) are subtracted. In other words, the exact finite-difference equations are subtracted from the finite-difference equations which contain the error. The result after dividing by, $e^{i\sigma m \Delta x} e^{i\beta n \Delta t}$ and substituting $\lambda = e^{i\beta \Delta t}$ is:

$$h^* \left[\lambda (e^{i\sigma \Delta x} + 1) - (e^{i\sigma \Delta x} + 1) \right] + u^* \left[H_0 \frac{\Delta t}{\Delta x} (2\theta \lambda + 2 - 2\theta) (e^{i\sigma \Delta x} - 1) \right] = 0$$

and:

$$h^* \left[g \frac{\Delta t}{\Delta x} (2\theta \lambda + 2 - 2\theta) (e^{i\sigma \Delta x} - 1) \right] + u^* (e^{i\sigma \Delta x} + 1) \left[\lambda - 1 + k \Delta t (\theta \lambda + 1 - \theta) \right] = 0 \quad (6.9)$$

Then dividing further by $e^{i\sigma \Delta x} + 1$ and substituting

$$i \tan\left(\frac{\sigma \Delta x}{2}\right) = \frac{e^{i\sigma \Delta x} - 1}{e^{i\sigma \Delta x} + 1}$$

results in:

$$(\lambda - 1)h^* + \left[(2\theta \lambda + 2 - 2\theta) H_0 \frac{\Delta t}{\sigma \Delta x} i \tan\left(\frac{\sigma \Delta x}{2}\right) \right] u^* = 0 \quad (6.10)$$

and

$$\left[(2\theta \lambda + 2 - 2\theta) g \frac{\Delta t}{\Delta x} i \tan\left(\frac{\sigma \Delta x}{2}\right) \right] h^* + \left[\lambda - 1 + k \Delta t (\theta \lambda + 1 - \theta) \right] u^* = 0 \quad (6.11)$$

Equations (6.10) and (6.11) represent two homogeneous linear equations in h^* and u^* .

Since h^* and u^* are not both identically equal to zero, the determinant of this system must vanish, hence:

$$(\lambda - 1) \left[\lambda - 1 + \theta (\lambda - 1) b + b \right] + 4 \left[\theta (\lambda - 1) + 1 \right]^2 a = 0 \quad (6.12)$$

Where:

$$a = g H_0 \left(\frac{\Delta t}{\Delta x} \right)^2 \tan^2\left(\frac{\sigma \Delta x}{2}\right) \text{ and } b = k \Delta t.$$

Equation (6.12) is a quadratic in $\lambda-1$ and may be solved to obtain

$$\lambda = 1 - \frac{8a\theta+b}{2(1+\theta b+4a\theta^2)} \pm i \frac{\sqrt{16a^2b^2}}{2(1+\theta b+4a\theta^2)}$$

now if $\lambda = r+is$ then $|\lambda| = \sqrt{r^2 + s^2} = \sqrt{\lambda\lambda^*}$, where λ^* is the complex conjugate of λ , and so

$$|\lambda| = \sqrt{\frac{1+(2\theta-2)^2 a + (\theta-1)b}{1 + 4\theta^2 a + \theta b}} \quad (6.13)$$

For the finite-difference equations to be stable, the error at time $t + \Delta t$ must be smaller than the error at time t .

Now λ is the growth factor for the propagation of all error waves through time. Therefore, if the system is to be stable in the sense that errors tend to zero monotonically then $|\lambda| < 1$, independent of either a or b . If this condition is satisfied, then the finite-difference scheme will be stable, independent of the values of Δx and Δt , since a and b are functions of Δx and Δt . For $|\lambda|$ to be strictly less than 1 the numerator in equation (6.13) must be less than the denominator.

Although the numerator may or may not be smaller than the denominator, depending upon the values of a and b , to ensure unconditional stability the numerator must always be smaller than the denominator. This will be ensured if the coefficients in the numerator are always smaller than the corresponding coefficients in the denominator. Since $(\theta-1) < \theta$, then this requirement is satisfied if

$$(2\theta+2)^2 < 4\theta^2$$

which means that $\theta > \frac{1}{2}$. Therefore, if $\theta > 0.5$, the linear finite-difference equations are unconditionally stable. If $0 < \theta < 0.5$, the linear finite-difference equations may be conditionally stable depending upon the values of Δx and Δt .

When $\theta = \frac{1}{2}$, substitution in equation (6.13) gives

$$|\lambda| = \sqrt{\frac{1+a-b/2}{1+a+b/2}}$$

Hence, as long as $b > 0$, the centered finite-difference equations are unconditionally stable. However, if $b = 0$, then there are no frictional effects in the river system and the centered finite-difference scheme becomes weakly stable.

6.2 APPROXIMATION

The finite-difference equations are an approximation to the exact partial differential equations. From the definition of a partial derivative, it is easily seen that as the space-time grid size approaches zero, the solution of the finite-difference equations must approach the solution of the partial differential equations. However, in practice, the space-time grid size used is not zero and hence the difference between the exact solution and the solution of the difference equations must be determined. This error can only be examined qualitatively by deriving the functional form of the error.

The differential equation is computed about the centre point of the x-t grid, i.e. at point $(x_m + \frac{\Delta x}{2}, t_n + \frac{\Delta t}{2})$.

Expanding K_m^n and K_m^{n+1} about the point $(x_m, t_n + \Delta t/2)$ in a standard Taylor series gives

$$\begin{aligned} (K_m^n)^{n+\frac{1}{2}} &= K_m^{n+\frac{1}{2}} - \frac{\Delta t}{2} \left(\frac{\partial K}{\partial t} \right)_m^{n+\frac{1}{2}} + \frac{\Delta t^2}{8} \left(\frac{\partial^2 K}{\partial t^2} \right)_m^{n+\frac{1}{2}} - \frac{\Delta t^3}{48} \left(\frac{\partial^3 K}{\partial t^3} \right)_m^{n+\frac{1}{2}} + \dots \\ (K_m^{n+1})^{n+\frac{1}{2}} &= K_m^{n+\frac{1}{2}} + \frac{\Delta t}{2} \left(\frac{\partial K}{\partial t} \right)_m^{n+\frac{1}{2}} + \frac{\Delta t^2}{8} \left(\frac{\partial^2 K}{\partial t^2} \right)_m^{n+\frac{1}{2}} + \frac{\Delta t^3}{48} \left(\frac{\partial^3 K}{\partial t^3} \right)_m^{n+\frac{1}{2}} + \dots \end{aligned}$$

Therefore, combining these two series results in

$$\left[\frac{K_m^{n+1} - K_m^n}{\Delta t} \right]_m^{n+\frac{1}{2}} = \left(\frac{\partial K}{\partial t} \right)_m^{n+\frac{1}{2}} + \frac{\Delta t^2}{24} \left(\frac{\partial^3 K}{\partial t^3} \right)_m^{n+\frac{1}{2}} + \frac{\Delta t^4}{3840} \left(\frac{\partial^5 K}{\partial t^5} \right)_m^{n+\frac{1}{2}} + \dots \quad (6.14)$$

and

$$\left[\frac{K_{m+1}^{n+1} - K_{m+1}^n}{\Delta t} \right]_{m+1}^{n+\frac{1}{2}} = \left(\frac{\partial K}{\partial t} \right)_{m+1}^{n+\frac{1}{2}} + \frac{\Delta t^2}{24} \left(\frac{\partial^3 K}{\partial t^3} \right)_{m+1}^{n+\frac{1}{2}} + \frac{\Delta t^4}{3840} \left(\frac{\partial^5 K}{\partial t^5} \right)_{m+1}^{n+\frac{1}{2}} + \dots \quad (6.15)$$

Now, expanding (6.14) and (6.15) in a two-dimensional Taylor series about the point $(x_m + \Delta x/2, t_n + \Delta t/2)$

$$\begin{aligned} \frac{K_m^{n+1} - K_m^n}{\Delta t} &= \frac{\partial K}{\partial t} + \frac{\Delta t^2}{24} \left(\frac{\partial^3 K}{\partial t^3} \right) - \frac{\Delta x}{2} \left[\left(\frac{\partial^2 K}{\partial x \partial t} \right) + \frac{\Delta t^2}{24} \left(\frac{\partial^4 K}{\partial x \partial t^3} \right) + \dots \right] + \\ &+ \frac{\Delta x^2}{8} \left[\left(\frac{\partial^3 K}{\partial x^2 \partial t} \right) + \frac{\Delta t^2}{24} \left(\frac{\partial^5 K}{\partial x^2 \partial t^3} \right) + \dots \right] - \dots \end{aligned} \quad (6.16)$$

$$\begin{aligned} \frac{K_{m+1}^{n+1} - K_m^n}{\Delta t} &= \frac{\partial K}{\partial t} + \frac{\Delta t^2}{24} \left(\frac{\partial^3 K}{\partial t^3} \right) + \frac{\Delta x}{2} \left[\left(\frac{\partial^2 K}{\partial x \partial t} \right) + \frac{\Delta t^2}{24} \left(\frac{\partial^4 K}{\partial x \partial t^3} \right) + \dots \right] + \\ &+ \frac{\Delta x^2}{8} \left[\frac{\partial^3 K}{\partial x^2 \partial t} + \frac{\Delta t^2}{24} \left(\frac{\partial^5 K}{\partial x^2 \partial t^3} \right) + \dots \right] + \dots \end{aligned} \quad (6.17)$$

Where all of the derivatives are understood to be written around the point $(x_m + \frac{\Delta x}{2}, t_n + \frac{\Delta t}{2})$

Using similar techniques, expansions may be obtained for:

$$\begin{aligned} \frac{K_{m+1}^{n+1} - K_m^n}{\Delta x} &= \frac{\partial K}{\partial x} + \frac{\Delta x^2}{24} \left(\frac{\partial^3 K}{\partial x^3} \right) - \frac{\Delta t}{2} \left[\frac{\partial^2 K}{\partial t \partial x} + \frac{\Delta x^2}{24} \left(\frac{\partial^4 K}{\partial t \partial x^3} \right) + \dots \right] + \\ &+ \frac{\Delta t^2}{8} \left[\frac{\partial^3 K}{\partial t^2 \partial x} + \frac{\Delta x^2}{24} \left(\frac{\partial^5 K}{\partial t^2 \partial x^3} \right) + \dots \right] - \dots \end{aligned} \quad (6.18)$$

$$\begin{aligned} \frac{K_{m+1}^{n+1} - K_m^{n+1}}{\Delta x} &= \frac{\partial K}{\partial x} + \frac{\Delta x^2}{24} \left(\frac{\partial^3 K}{\partial x^3} \right) + \frac{\Delta t}{2} \left[\frac{\partial^2 K}{\partial t \partial x} + \frac{\Delta x^2}{24} \left(\frac{\partial^4 K}{\partial t \partial x^3} \right) + \dots \right] + \\ &+ \frac{\Delta t^2}{8} \left[\frac{\partial^3 K}{\partial t^2 \partial x} + \frac{\Delta x^2}{24} \left(\frac{\partial^5 K}{\partial t^2 \partial x^3} \right) + \dots \right] + \dots \end{aligned} \quad (6.19)$$

Again, all derivatives are understood to be evaluated at the point

$$(x_m + \frac{\Delta x}{2}, t_n + \frac{\Delta t}{2})$$

Using these series expansions, the approximation to the derivatives of the function K may be expressed in a Taylor series. Thus:

$$\begin{aligned} \frac{\partial K}{\partial t} &\approx \frac{K_m^{n+1} + K_{m+1}^{n+1} - K_m^n - K_{m+1}^n}{2\Delta t} \\ &= \frac{\partial K}{\partial t} + \frac{\Delta t^2}{24} \left(\frac{\partial^3 K}{\partial t^3} \right) + \frac{\Delta x^2}{8} \left[\frac{\partial^2 K}{\partial x^2 \partial t} + \frac{\Delta t^2}{8} \left(\frac{\partial^5 K}{\partial x^2 \partial t^3} \right) \right] + \dots \end{aligned} \quad (6.20)$$

$$\begin{aligned} \frac{\partial K}{\partial x} &\approx \theta \left[\frac{K_{m+1}^{n+1} - K_m^{n+1}}{\Delta x} \right] + (1-\theta) \left[\frac{K_{m+1}^n - K_m^n}{\Delta x} \right] \\ &= \frac{\partial K}{\partial x} + (2\theta-1) \frac{\Delta t}{2} \left[\frac{\partial^2 K}{\partial t \partial x} + \frac{\Delta x^2}{24} \left(\frac{\partial^4 K}{\partial t \partial x^3} \right) \right] + \frac{\Delta t^2}{8} \left[\left(\frac{\partial^3 K}{\partial t^2 \partial x} \right) + \frac{\Delta x^2}{24} \left(\frac{\partial^5 K}{\partial t^2 \partial x^3} \right) \right] + \end{aligned}$$

$$+ \frac{\Delta x^2}{24} \left(\frac{\partial^3 K}{\partial x^3} \right) + \dots \quad (6.21)$$

These expressions may be substituted into the finite-difference equations for the linearized equations of motion to give

$$\begin{aligned} & \frac{\partial h}{\partial t} + \frac{\Delta t^2}{24} \frac{\partial^3 h}{\partial t^3} + \frac{\Delta x^2}{8} \left(\frac{\partial^3 h}{\partial x^2 \partial t} + \frac{\Delta t^2}{24} \frac{\partial^5 h}{\partial x^2 \partial t^3} \right) + \\ & + H_0 \left[\frac{\partial u}{\partial x} + (2\theta-1) \frac{\Delta t}{2} \left(\frac{\partial^2 u}{\partial t \partial x} + \frac{\Delta x^2}{24} \frac{\partial^4 u}{\partial t \partial x^3} \right) + \frac{\Delta t^2}{8} \left(\frac{\partial^3 u}{\partial t^2 \partial x} + \frac{\Delta x^2}{24} \frac{\partial^5 u}{\partial t^2 \partial x^3} \right) + \right. \\ & \left. + \frac{\Delta x^2}{24} \frac{\partial^3 u}{\partial x^3} + \dots \right] = 0 \end{aligned} \quad (6.22)$$

for the continuity equation and

$$\begin{aligned} & \frac{\partial u}{\partial t} + \frac{\Delta t^2}{24} \frac{\partial^3 u}{\partial t^3} + \frac{\Delta x^2}{8} \left(\frac{\partial^2 u}{\partial x^2 \partial t} + \frac{\Delta t^2}{8} \frac{\partial^5 u}{\partial x^2 \partial t^3} \right) + \\ & + g \left[\frac{\partial h}{\partial x} + (2\theta-1) \frac{\Delta t}{2} \left(\frac{\partial^2 h}{\partial t \partial x} + \frac{\Delta x^2}{24} \frac{\partial^4 h}{\partial t \partial x^3} \right) + \frac{\Delta t^2}{8} \left(\frac{\partial^3 h}{\partial t^2 \partial x} + \frac{\Delta x^2}{24} \frac{\partial^5 h}{\partial t^2 \partial x^3} \right) + \right. \\ & \left. + \frac{\Delta x^2}{24} \frac{\partial^3 h}{\partial x^3} + ku + \dots \right] = 0 \end{aligned} \quad (6.23)$$

for the momentum equation.

The truncation errors E_1 and E_2 may be obtained by subtracting the exact partial differential equations (6.1) and (6.2) from (6.22) and (6.23) respectively, so that

$$\begin{aligned} E_1 = & (2\theta-1)H_0 \frac{\Delta t}{2} \left(\frac{\partial^2 u}{\partial t \partial x} + \frac{\Delta x^2}{24} \frac{\partial^4 u}{\partial t \partial x^3} \right) + \\ & + \frac{\Delta t^2}{8} \left[\frac{\partial^3 h}{\partial t^3} + H_0 \left(\frac{\partial^3 u}{\partial t^2 \partial x} + \frac{\Delta x^2}{24} \frac{\partial^5 u}{\partial t^2 \partial x^3} \right) \right] + \\ & + \frac{\Delta x^2}{8} \left[\left(\frac{\partial^3 h}{\partial x^2 \partial t} + \frac{\Delta t^2}{24} \frac{\partial^5 h}{\partial x^2 \partial t^3} + \frac{H_0}{3} \frac{\partial^3 u}{\partial x^3} \right) \right] + \dots \end{aligned} \quad (6.24)$$

$$E_2 = (2\theta-1) g \frac{\Delta t}{2} \left(\frac{\partial^2 h}{\partial t \partial x} + \frac{\Delta x^2}{24} \frac{\partial^4 h}{\partial t \partial x^3} \right) +$$

$$\begin{aligned}
& + \frac{\Delta t^2}{8} \left[\frac{\partial^3 u}{\partial t^3} + g \left(\frac{\partial^3 h}{\partial t \partial x} + \frac{\Delta x^2}{3} \frac{\partial^5 h}{\partial t^2 \partial x} \right) \right] \\
& + \frac{\Delta x^2}{8} \left[\left(\frac{\partial^2 u}{\partial x^2 \partial t} + \frac{\Delta t^2}{3} \frac{\partial^5 u}{\partial x^3 \partial t^3} + \frac{g}{3} \frac{\partial h^3}{\partial x^3} \right) + \dots \right] \quad (6.25)
\end{aligned}$$

Clearly, both E_1 and E_2 approach zero as Δx and Δt approach zero and so the difference scheme converges to the solution of the partial differential equations. In addition, both (6.24) and (6.25) may be written in the form $E = (2\theta-1) O(\Delta t) + O(\Delta t^2) + O(\Delta x^2)$ (6.26)

Therefore, the scheme has first order accuracy for linear equations with respect to Δt and second order accuracy with respect to Δx for any value of $0 \leq \theta \leq 1$. When $\theta = 0.5$, the scheme has second order accuracy and the farther θ departs from this value, the larger the truncation error becomes. Equation (26) also shows that the accuracy of the solution is independent of the relationship between Δx and Δt .

6.3 CONSERVATION OF MASS

The total mass in a bounded system should be conserved during numerical computation if account is taken of the increase or decrease of mass through the system boundaries. If mass is added during the course of computation, then wave amplitudes will increase with time and the computation will become unstable.

The continuity equation may be written in the form

$$\frac{\partial Q}{\partial x} + \frac{\partial A}{\partial t} = 0 \quad (6.27)$$

The finite-difference approximation to equation (6.27) is:

$$\frac{A_m^{n+1} + A_{m+1}^{n+1} - A_m^n - A_{m+1}^n}{2\Delta t} + \theta \left(\frac{Q_{m+1}^{n+1} - Q_m^{n+1}}{\Delta x} \right) + (1-\theta) \left(\frac{Q_{m+1}^n - Q_m^n}{\Delta x} \right) = 0 \quad (6.28)$$

Adding all the terms along the x-axis from $m = 1$ to $m = M$, gives

$$\frac{\Delta x}{\Delta t} \left[\sum_{m=1}^{m=M-1} \frac{A_m^{n+1} + A_{m+1}^{n+1}}{2} - \sum_{m=1}^{m=M-1} \frac{A_m^n + A_{m+1}^n}{2} \right] =$$

$$= \theta (Q_1^{n+1} - Q_M^{n+1}) + (1-\theta) (Q_1^n - Q_M^n)$$

which may be written as

$$\frac{\text{change of volume}}{\Delta t} = \theta (Q_1^{n+1} - Q_M^{n+1}) + (1-\theta) (Q_1^n - Q_M^n)$$

If $\theta = 0.5$ this becomes

$$\frac{\text{change in volume}}{\Delta t} = \text{inflow} - \text{outflow}$$

Thus, for one time step, the central finite-difference scheme conserves mass, so long as the mass which enters or leaves through the system boundaries is accounted for.

When $\theta = 1$

$$\frac{\text{change in volume}}{\Delta t} = Q_1^{n+1} - Q_M^{n+1}$$

and the system does not conserve mass to the same extent that it does when $\theta = 0.5$. The extent to which the system conserves mass is proportional to the extent to which θ departs from 0.5.

6.4 NUMERICAL RESULTS

The analytical results obtained in the preceding sections were obtained using the linearized equations of motion. The equations derived in chapter 4 are non-linear and so the results obtained in this chapter are not necessarily applicable. The conservation of mass analysis used the equation of continuity derived in chapter 4 and so the results from it are directly applicable.

The stability and convergence properties of a non-linear system may be investigated via numerical experiments. In this way, the effects of the non-linear terms may be calculated and some properties of the non-linear system discovered. This method has many drawbacks with the most obvious one being that not all cases may be investigated. The results obtained by numerical experiment are far from convincing.

The computer program described in this report has been run for a large variety of test cases, and the numerical results tend to agree closely with the results obtained from the analytic investigations.

It should be noted that the implicit scheme did tend to exhibit bounded oscillations with $\theta = 0.5$ when the wave period was quite short compared to the time step or when the wave was very large and abrupt. These oscillations died rapidly when θ was increased beyond 0.5. The numerical damping of a wave is proportional to θ . That is, as θ becomes close to 1.0, waves are damped out quite rapidly, a result predicted by the analysis.

In conclusion, the results of the analytic investigations are borne out by numerical experiments.

These results have also been confirmed by other investigators (13), and from their work, for unsteady flow situations, the "best" value of θ is approximately 0.55. For steady state conditions, rapid convergence from inaccurate initial conditions may be obtained if θ is increased to a value approaching 1.0.

CHAPTER 7

7.0 SOLUTION OF EQUATIONS

The major difficulty with the implicit method has always been the simultaneous solution of a large number of equations. For many years, this difficulty was insurmountable and very little work was done using the implicit method. In recent years, however, both Freud and Amien have produced a number of papers dealing with the implicit method. They have used the Newton iteration scheme to solve the non-linear equations of motion in a single branched channel.

In using the Newton iteration scheme, both Freud and Amien have made use of a particular formulation of the equations of motion and have used the fact that the coefficient matrix of the equivalent linear system is both banded and sparse. K.M. Brown [6, 7] has developed a modified Newton-like method for solving systems of non-linear algebraic or transcendental equations that is applicable to problems of open channel flow. The advantage of Brown's method is that the method makes no use of the form in which the equations are expressed. This means that a model may be designed which is totally independent of the particular formulation of the equations of motion to be used.

The following explanation of Brown's algorithm is taken from his two papers. It is worth noting that although this algorithm does not make any reference to the exact form of the equations of motion, it could be modified to do so. In practice, the coefficients of the equivalent linear system are sparse, with a maximum of six non-zero entries in each row. If this fact were taken advantage of, the operation of the algorithm could be speeded up dramatically.

7.1 NEWTON'S METHOD

Consider a system of N non-linear algebraic equations in N unknowns described by

$$\begin{aligned}
f_1(\vec{x}) &= f_1(x_1, x_2, x_3, \dots, x_N) = 0 \\
f_2(\vec{x}) &= f_2(x_1, x_2, x_3, \dots, x_N) = 0 \\
&\vdots \\
f_N(\vec{x}) &= f_N(x_1, x_2, x_3, \dots, x_N) = 0
\end{aligned} \tag{7.1}$$

Then, let $\vec{x}^i = (x_1^i, x_2^i, \dots, x_N^i)$ be the i^{th} approximation to the solution in a suitably recursive scheme. If the recursive scheme is Newton's method, then the iteration scheme is given by:

$$\vec{x}^{n+1} = \vec{x}^n - [\vec{J}(\vec{f}^n)]^{-1} \vec{f}^n, \quad n = 0, 1, 2, \dots \tag{7.2}$$

where $\vec{J}(\vec{f})$ is the Jacobian matrix $[\partial f_i / \partial x_j]$ and the superscript n means that all functions involved are to be evaluated at $\vec{x} = \vec{x}^n$.

For this iteration procedure, the following convergence theorem is well-known.

Theorem 1.

If

- (i) in a closed region R whose interior contains a root $\vec{x} = \vec{r}$ of (7.1), each f_i is twice continuously differentiable for $i = 1, \dots, N$.
- (ii) $\vec{J}(\vec{f})$ is non-singular at $\vec{x} = \vec{r}$, and
- (iii) \vec{x}^0 is chosen in R sufficiently close to $\vec{x} = \vec{r}$.

Then the iteration (7.2) is convergent to \vec{r} .

7.2 BROWN'S ALGORITHM

Brown's algorithm is essentially a modified Newton's method based on Gaussian elimination. The forward triangularization of the full Jacobian matrix is approximated by working with one row at a time, eliminating one variable for each row treated. If the conditions of Theorem 1 apply, and \vec{x}^n denotes the n^{th} approximation to a root $\vec{x} = \vec{r}$ of equations 7.1, then the method consists of the following steps:

STEP 1

Expand $f_1(\vec{x})$ in a Taylor series about the point \vec{x}^n and retain only the first order terms to obtain the linear approximation.

$$f_1(\vec{x}) \approx f_1(\vec{x}^n) + \frac{\partial f_1(\vec{x}^n)}{\partial x_1} (x_1 - x_1^n) + \frac{\partial f_1(\vec{x}^n)}{\partial x_2} (x_2 - x_2^n) + \dots$$

$$\dots + \frac{\partial f_1(\vec{x}^n)}{\partial x_N} (x_N - x_N^n) \quad (7.3)$$

STEP 2

Equate the right hand side to zero and solve for the variable, say x_N , whose partial derivative $\frac{\partial f_1(\vec{x}^n)}{\partial x_N}$ is largest in absolute value.

STEP 3

$$\text{Therefore: } x_N = x_N^n - \sum_{j=1}^{N-1} \left[\frac{\partial f_1(\vec{x}^n)}{\partial x_j} / \frac{\partial f_1(\vec{x}^n)}{\partial x_N} \right] (x_j - x_j^n) -$$

$$- f_1(\vec{x}^n) / \frac{\partial f_1(\vec{x}^n)}{\partial x_N} \quad (7.4)$$

Provided that the trial solution \vec{x}^n is sufficiently close to the root \vec{r} , $\vec{J}(\vec{x}^n)$ will be close to $\vec{J}(\vec{r})$ which is a non-singular matrix and hence at least one of the $\frac{\partial f_1(\vec{x}^n)}{\partial x_N}$ will be different from zero.

Hence a solution such as the above (7.4) can always be carried out.

$$\frac{\partial f_1(\vec{x}^n)}{\partial x_j} \text{ and } \frac{\partial f_1(\vec{x}^n)}{\partial x_j} / \frac{\partial f_1(\vec{x}^n)}{\partial x_N}, \text{ for } j = 1, \dots, N-1 \text{ and}$$

$$f_1(\vec{x}^n) / \frac{\partial f_1(\vec{x}^n)}{\partial x_N} \text{ are saved for later use.}$$

STEP 4

Rename the left side of equation (7.4) $b_N(x_1, x_2, \dots, x_{N-1})$ and define $b_N^n = b_N(x_1^n, x_2^n, \dots, x_{N-1}^n)$. (7.5)

STEP 5

Define a function g_2 of the $N-1$ variables x_1, \dots, x_{N-1} by

$$g_2(x_1, x_2, \dots, x_{N-1}) =$$

$$= f_2(x_1, x_2, \dots, x_{N-1}, b_N(x_1, x_2, \dots, x_{N-1})) \quad (7.6)$$

and

$$g_2^n = f_2(x_1^n, x_2^n, \dots, x_{N-1}^n, b_N^n). \quad (7.7.)$$

STEP 6

Expand (7.7) in a Taylor Series, this time about the point $(x_1^n, x_2^n, \dots, x_{N-1}^n)$, linearize and solve for the variable, say x_{N-1} , whose corresponding partial derivative is largest in absolute value to obtain

$$x_{N-1} = x_{N-1}^n - \frac{N-2}{J-1} \left[\frac{\partial g_2(\vec{x}^n)}{\partial x_j} / \frac{\partial g_2(\vec{x}^n)}{\partial x_{N-1}} \right] (x_j - x_j^n) - \frac{\partial g_2(\vec{x}^n)}{\partial x_{N-1}} \quad (7.8)$$

where $\partial g_2(\vec{x})/\partial x_j$ is obtained by differentiating equation (7.6) using the chain rule to obtain

$$\begin{aligned} \frac{\partial g_2(\vec{x})}{\partial x_j} &= \frac{\partial f_2(\vec{x}^n)}{\partial x_j} + f_2(\vec{x}^n) \frac{\partial b_N}{\partial x_j} \Big|_{x_1^n, x_2^n, \dots, x_{N-1}^n} \\ &= \frac{\partial f_2(\vec{x}^n)}{\partial x_j} + f_2(\vec{x}^n) \left[- \frac{\partial f_1(\vec{x}^n)}{\partial x_j} / \frac{\partial f_1(\vec{x}^n)}{\partial x_N} \right] \end{aligned}$$

STEP 7

Redefine the left hand side of (7.8) to be b_{N-1} , which is a function of the $N-2$ remaining variables so that

$$b_{N-1}(x_1, x_2, \dots, x_{N-2}) = \text{right hand side of (7.8)} \quad (7.9)$$

STEP 8

Again, save all of the ratios formed. Theorem 2 will state that this process is well-defined in that there does exist at least one non-zero partial derivative at each step in the process.

STEP 9

Define $g_3 = f_3(x_1, x_2, \dots, x_{N-2}, b_{N-1}, b_N)$ where b_{N-1} and b_N are obtained by back substitution in the linearized equations (7.5) and (7.9).

STEP 10

Repeat the process of expansion in Taylor Series, linearization and elimination of one variable, saving all of the ratios formed at each step. In this way one variable at a time is replaced by a b_j with g_k being expanded about the point $(x_1^n, x_2^n, \dots, x_{N-k+1}^n)$.

STEP 11

At the last step in the elimination process

$$g_N = f_N(x_1, b_2, b_3, \dots, b_N)$$

where the b_j 's have been obtained by back substitution in the $N-1$ rowed triangularized linear system which has been built up and now has the form:

$$b_i = x_i^{n-i+1} - \left[\frac{\partial g_{n-i+1}}{\partial x_j} / \frac{\partial g_{n-i+1}}{\partial x_j} \right] (b_j - x_j^n) - \left[\frac{\partial g_{n-i+1}}{\partial x_i} / \frac{\partial g_{n-i+1}}{\partial x_j} \right]$$

$$\text{for } i = N, N-1, \dots, 2 \quad (7.10)$$

STEP 12

Expanding and solving for x_1 results in:

$$x_1 = x_1^n - \left[\frac{g_N^n}{\frac{\partial g_N^n}{\partial x_1}} \right]$$

Use this value of x_1 as the improved approximation, x_1^{n+1} , to the first component, r_1 , of the root \vec{r} , call it b_1 and back solve the system (10) to get improved approximations to the other components r_j . In the back-solving process, x_j^{n+1} will equal the corresponding b_j .

It should be noted that numerical examples show that this process is NOT equivalent to Newton's Method.

7.3 MATRIX FORMULATION

In the above description of the algorithm, the variables are eliminated in the order x_N, x_{N-1}, \dots, x_2 . This was done to facilitate a definite identification of the particular variable being eliminated. If the chain rule is used to expand each derivative $\partial g_1 / \partial x_1$ which appears in the algorithm, the following matrix representation of the forward part of the method is obtained:

$$(\vec{H}) (\vec{x}^{n+1} - \vec{x}^n) = (-\vec{g})$$

where $(\vec{H}) = (h_{ij})$ is given by

$$(h_{ij}) = \frac{\partial f_1}{\partial x_j} \quad \text{for } i = 1, j = 1, 2, \dots, N$$

and if $i = 2, \dots, N, j = 1, \dots, N$

$$(h_{ij}) = \frac{(-1)^{i+1} \begin{vmatrix} \frac{\partial f_1}{\partial x_j} & \frac{\partial f_1}{\partial x_{N-i+2}} \dots & \frac{\partial f_1}{\partial x_{N-1}} & \frac{\partial f_1}{\partial x_N} \\ \frac{\partial f_2}{\partial x_j} & \frac{\partial f_2}{\partial x_{N-i+2}} \dots & \frac{\partial f_2}{\partial x_{N-1}} & \frac{\partial f_2}{\partial x_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_{i-1}}{\partial x_j} & \frac{\partial f_{i-1}}{\partial x_{N-i+2}} \dots & \frac{\partial f_{i-1}}{\partial x_{N-1}} & \frac{\partial f_{i-1}}{\partial x_N} \end{vmatrix}}{\begin{vmatrix} \frac{\partial f_1}{\partial x_{N-i+2}} & \dots & \frac{\partial f_1}{\partial x_N} \\ \vdots & \dots & \vdots \\ \frac{\partial f_{i-1}}{\partial x_{N-i+2}} & \dots & \frac{\partial f_{i-1}}{\partial x_N} \end{vmatrix}}$$

and the argument of each f_j is the progressive argument generated by the algorithm.

By expanding the determinant in the numerator, it is easily seen that $h_{ij} = 0$ for $j > N-i+1$. Hence, the matrix (H) is transverse upper triangular. The resultant matrix h_{ij} is in exactly the same form as the matrix which results from transverse triangularization of the Jacobian matrix J using Gaussian Elimination with partial pivoting.

The argument of each f_{ij} in the triangularization form of J is simply x^n ; however, at the root, the two arguments coincide.

For the 3 x 3 cases, the matrix has the explicit form

$$H = \begin{vmatrix} f_{11} & f_{12} & f_{13} \\ -\frac{\begin{vmatrix} f_{11} & f_{13} \\ f_{21} & f_{23} \end{vmatrix}}{f_{13}} - \frac{\begin{vmatrix} f_{12} & f_{13} \\ f_{22} & f_{23} \end{vmatrix}}{f_{13}} & 0 \\ \begin{vmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{vmatrix} & 0 & 0 \\ \begin{vmatrix} f_{12} & f_{13} \\ f_{22} & f_{23} \end{vmatrix} & & \end{vmatrix}$$

and the arguments used for the function evaluations are:

x^n for f_{1j} , $j = 1, 2, 3$

$(x_1^n, x_2^n, b_3(x_1^n, x_2^n))$ for f_{2j} , $j = 1, 2, 3$

$(x_1^n, b_2(x_1^n), b_3(x_1^n, b_2(x_1^n)))$ for f_{3j} , $j = 1, 2, 3$

Brown [7] proves that under the hypothesis of Theorem 1 and for the type of iteration described above, the following theorem is true:

Theorem 2 There exists a non-vanishing partial derivative at the i th step of the elimination process.

Essentially, the elimination process consists of expanding each function in a Taylor Series about r^1 , linearizing, and solving for

the variable whose corresponding partial derivative is largest in value. Theorem 2 states that there will always be a largest partial derivative. At each step of the elimination, all of the values of the variables found by the previous elimination steps are used in the appropriate equation. Back substitution into the system of equations results in a solution at the first step of the iteration. The whole elimination process is restarted to obtain the second step of the iteration.

The following local convergence theorem for iteration functions states that the above procedure will, indeed, converge to a solution.

Theorem 3 Let the functions F_1, F_2, \dots, F_N be defined in a region R , and let them satisfy the following conditions:

- i) The first partial derivations of F_1, \dots, F_N exist and are continuous in R .
- ii) The system.

$$\vec{x} = \vec{F}(\vec{x})$$

has a solution r in the interior of R such that $\vec{J}(\vec{F})_r = 0$, the zero matrix.

Then there exists a number $\epsilon > 0$ such that an algorithm of the form

$$\vec{x}^{n+1} = F(\vec{x}^n) \quad n = 0, 1, 2, \dots$$

converges to \vec{r} for any choice of the starting point, \vec{x}^0 , which satisfies $\|\vec{F} - \vec{x}^0\| < \epsilon$, where $\|\cdot\|$ denotes the Euclidian norm.

It can be proven that the Jacobian matrix will equal the zero matrix after a sufficient number of iterations. Hence, this proves that the algorithm will converge.

CHAPTER 8

8.0 APPLICATIONS OF THE MODEL

A computer model, MOD*, has been written to perform all of the bookkeeping and calculations discussed in the previous chapters. A user's guide to MOD* and a complete listing of the FORTRAN program is given in the appendices. This program has been tested on a wide variety of problems and seems to work as expected. Three of the test cases are presented here as illustrations of what may be done with the model. The three problems are: Stoker's famous problem of a flood routed through the junction of the Ohio and Mississippi rivers; a test case made up of a network; and the Saint Clair river system for a variety of flow conditions. The three test cases illustrate most of the considerations in setting up and using the numerical model.

8.1 STOKER'S PROBLEM

Stoker [21] used an explicit finite difference scheme to calculate the propagation of a flood wave through the junction of the Ohio and Mississippi rivers. His simplified model corresponded in a very rough way to the geometry of the actual river system. Figure 9 shows a schematic plan of the junction. Upstream of the junction, both rivers are assumed to be 1,000 feet wide, rectangular, and have a constant bottom slope of 0.5 ft./mile. The downstream Mississippi River has a width of 2,000 feet, is rectangular and has a constant bottom slope of 0.49 ft./mile. This change in slope is necessary to allow a constant initial depth of water of 20 feet for uniform flow. A constant friction factor of 0.03 for Manning's n is used throughout the river.

The initial conditions used are that the water depth is 20 feet throughout the river and the initial velocity is 3.49 feet/second. At the top of the Mississippi River, 50 miles from the junction, the depth of water is kept at a constant 20 feet. At the top of the Ohio River, the boundary condition (Figure 10) is given as a 20 foot increase in depth (from an initial 20 foot depth to a final 40 foot depth) over a four hour period. This rate of rise, 5 feet per hour, is an extreme case since one of the biggest floods ever recorded in the Ohio River, which was in 1947, had a maximum rate of rise of only 0.7

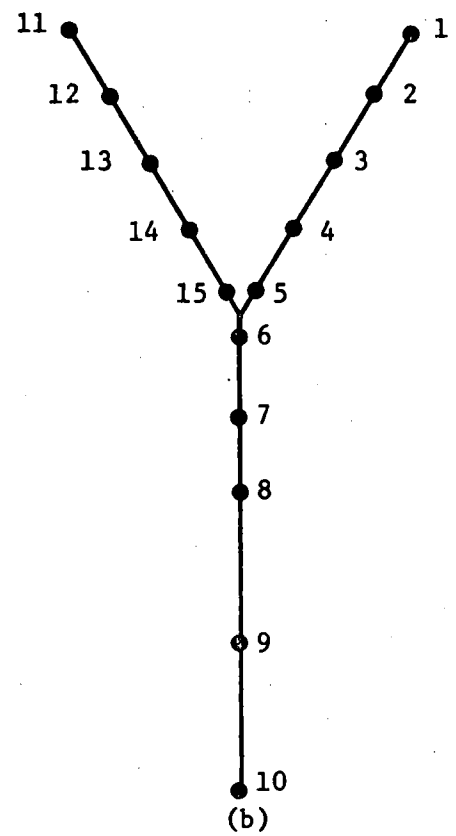
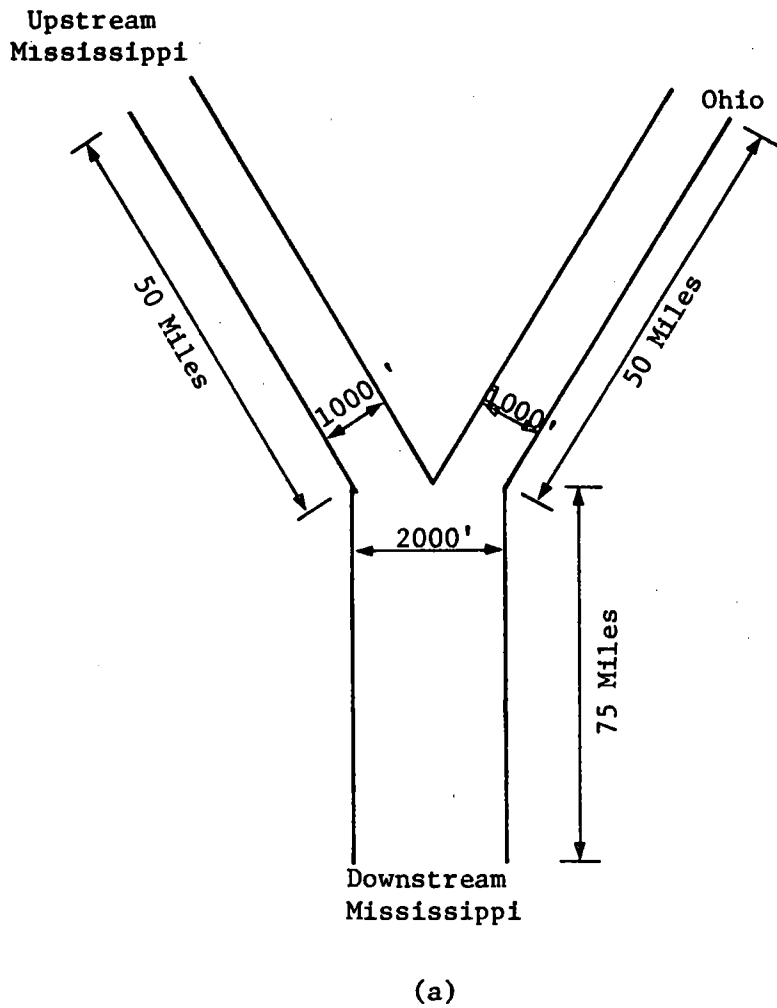


FIGURE 9: SCHEMATIC PLAN OF THE OHIO-MISSISSIPPI RIVER JUNCTION FOR STOKER'S PROBLEM

feet/hour. The downstream boundary condition, 75 miles downstream from the junction, is given by a stage-discharge curve developed from Manning's formula.

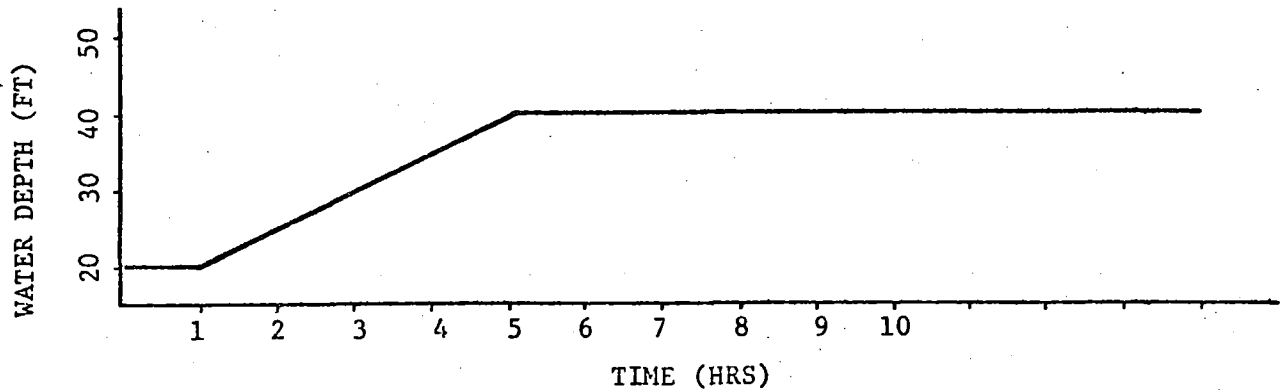


FIGURE 10: BOUNDARY CONDITIONS FOR STOKER'S PROBLEM

Stoker's solution to the problem used a variable reach length and a variable time step. The reach length was usually taken to be one mile. The time step was calculated, using the reach length and the expected speed of time of the disturbance, using the Courant stability criterion. This resulted in a time step between 0.024 hours and 0.17 hours. For the implicit model, the reach length used was 12.5 miles upstream of the junction and 25 miles downstream of the junction. The time step used throughout was one hour.

The results of the computer run are shown in Figure 11. Although Stoker does not publish numerical results but only publishes graphs [21], a comparison of the results shows excellent qualitative agreement. The steepening of the wave front as it moves through the system is more pronounced with the implicit model than it is from Stoker's results. This would be expected, since this model takes into account non-linear effects and Stoker solves only the linear equations

WATER DEPTH (FT.)

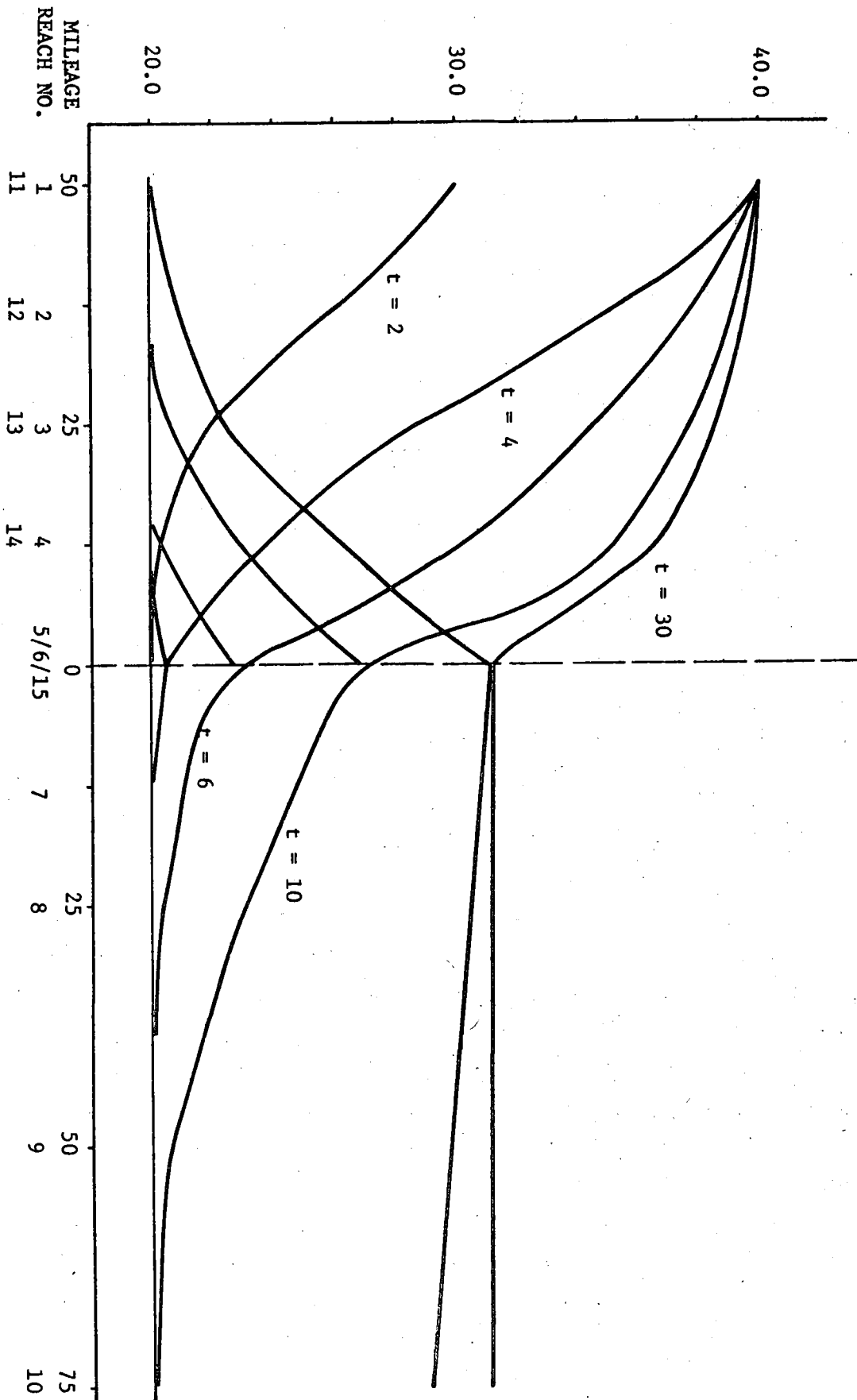


FIGURE 11: COMPUTED WATER LEVEL PROFILES FOR STOKER'S PROBLEM

of motion.

The most important aspect of this computer run, however, is the length of time step and reach length used. The reach length and time step used by Stoker are dictated by the Courant stability criterion. If he had used larger time steps or reach lengths, his numerical method would have become unstable. Since the implicit method is unconditionally stable, no such restrictions are necessary. The grid spacing is, then, dictated purely by the resolution required for the problem. In many cases, then, a much larger time or distance step, or both, may be used by the implicit method.

8.2 NETWORK PROBLEM

This second example shows the versatility of the implicit model. It consists of a complicated network arranged so that the flows differ in all of the branches. The schematization is shown in figure 12. This is not an actual problem in that it does not correspond to any specific river. It is intended, however, to indicate the type of network that may be handled by the implicit method.

The geometrical data are shown in Table 1. All of this data are entered on data cards and used in the program as explained in the User's Guide. A friction factor of 0.025 was used throughout, with Mannings' formula used to calculate the friction losses. The main branch consists of nodes 1 to 11. A branch containing nodes 12, 13, 14 and 15 with flow from node 3 to node 7 defines an island. A second branch containing nodes 16, 17 and 18 with flow to node 14, represents an embayment which has no flow but contributes to storage. A third branch containing nodes 19 and 20 and receiving flow from node 9, defines a delta region.

The model was run with a time step of one hour for a total of 5 time steps. Since only steady state conditions were required, a weighting factor of 1.0 was used to ensure rapid convergence to a steady state. The boundary conditions used were; a depth of 100 feet at nodes 1 and 11 and zero velocity at node 16. Initial conditions were a depth of 100 feet of water throughout the network and an initial velocity of 6 feet per second was used.

The results obtained are not too surprising and show the

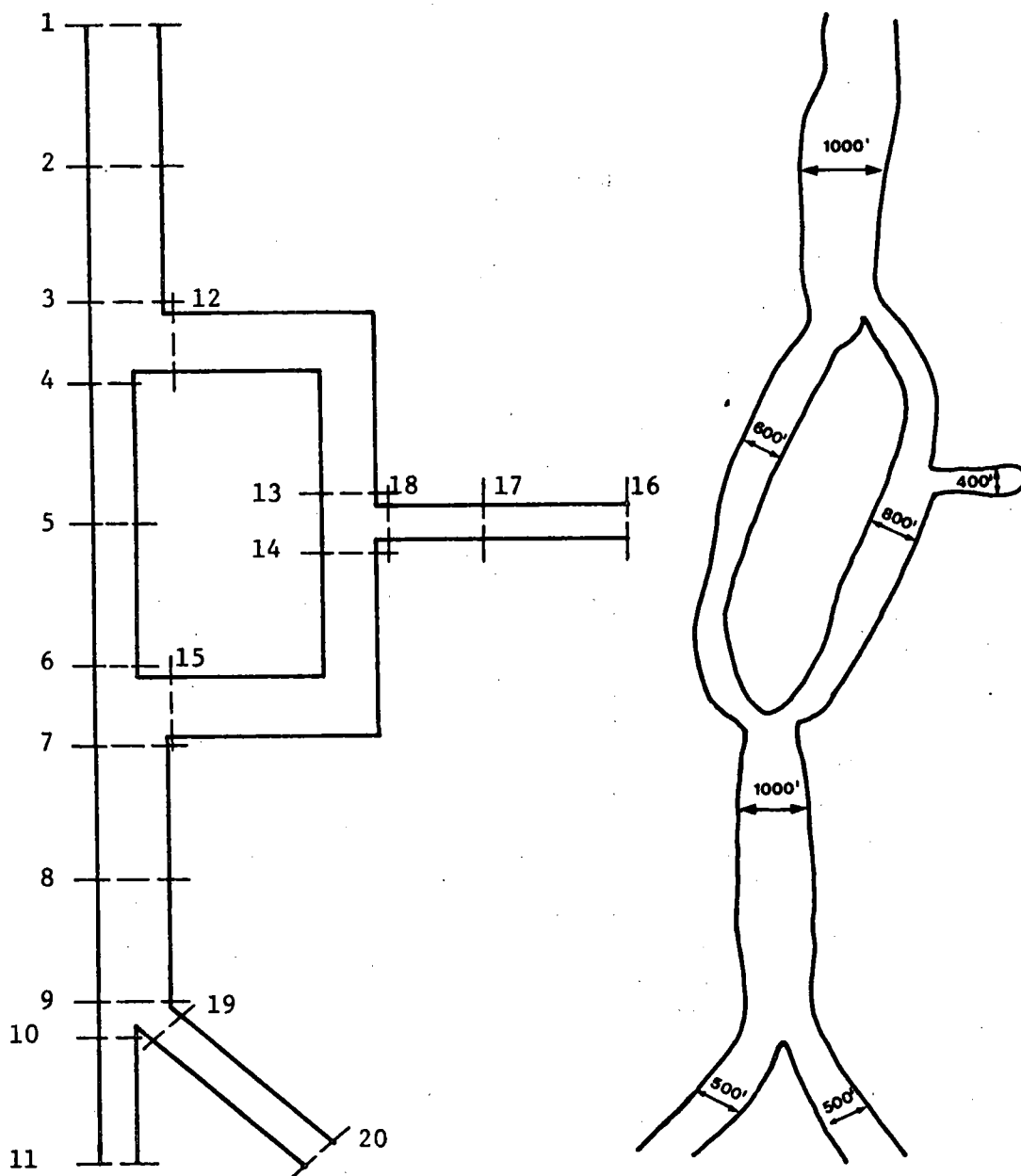


FIGURE 12: SCHEMATIC PLANS OF THE RIVER NETWORK PROBLEM

NODE NUMBER	WIDTH OF CHANNEL (FT.)	INVERT LEVEL BELOW DATUM (FT.)	REACH LENGTH (FT.)	MANNING'S N	COMMENTS
1	1000	0.00	25000	0.025	M
2	1000	1.25	25000	0.025	A
3	1000	2.50	800	0.025	I
4	600	2.50	25000	0.025	N
5	600	3.75	25000	0.025	C
6	600	5.00	800	0.025	H
7	1000	5.00	25000	0.025	A
8	1000	6.25	25000	0.025	N
9	1000	7.50	100	0.025	N
10	500	7.50	25000	0.025	E
11	500	8.75	1	0.025	L
12	800	2.50	25000	0.025	BRANCH AROUND ISLAND
13	800	3.75	400	0.025	
14	800	3.75	25000	0.025	
15	800	5.00	1	0.025	
16	400	3.75	25000	0.025	EMBAYMENT
17	400	3.75	25000	0.025	
18	400	3.75	1	0.025	
19	500	7.50	25000	0.025	DELTA
20	500	8.75	1	0.025	

TABLE 1: GEOMETRICAL DATA FOR NETWORK PROBLEM

NODE NUMBER	INITIAL		COMPUTED STEADY STATE		
	VELOCITY	WATER DEPTH	VELOCITY	WATER DEPTH	DISCHARGE
1	6.00	100.00	9.74	100.00	973989.7
2	6.00	100.00	9.76	99.78	973996.1
3	6.00	100.00	9.78	99.56	974014.6
4	6.00	100.00	6.99	99.56	417434.6
5	6.00	100.00	6.95	100.06	417441.0
6	6.00	100.00	6.92	100.59	417431.3
7	6.00	100.00	9.68	100.59	974005.2
8	6.00	100.00	9.70	100.40	973978.8
9	6.00	100.00	9.72	100.20	973960.9
10	6.00	100.00	9.72	100.20	486980.5
11	6.00	100.00	9.74	100.00	486977.3
12	6.00	100.00	6.99	99.56	556580.1
13	6.00	100.00	6.95	100.06	556588.7
14	6.00	100.00	6.95	100.06	556586.8
15	6.00	100.00	6.92	100.59	556573.9
16	0.00	100.00	0.00	100.06	0.0
17	0.00	100.00	0.00	100.06	0.0
18	0.00	100.00	0.00	100.06	0.0
19	6.00	100.00	9.72	100.20	486980.5
20	6.00	100.00	9.74	100.00	486977.3

Table 2: INITIAL AND COMPUTED FLOW CONDITIONS FOR NETWORK
PROBLEM

type of calculation that may be carried out using the model. The initial data abstraction and keypunching of data cards took about 2 hours and the computer time required for the computations was approximately 30 minutes. The versatility of the program is demonstrated in that, once data are available, setting up the data cards takes a very short time and results can be obtained quite quickly. The results of this computer run are shown in Table 2.

The run for 5 time steps produced quite good convergence to a steady state solution. Comparing the results from the fourth time step to the results of the fifth time step has shown that the height converged to within 0.01 feet and the velocities to within 0.07 feet per second. The maximum difference in the discharge was about 72.2 cusecs and this is well within the accuracy of the equation solving subroutine which was only working to 5 significant digits. The non-conservation of mass mentioned in the chapter on stability does not account for this 72.2 cusecs difference, since this example does not involve unsteady flow. The non-conservation of mass shown here is due entirely to the accuracy limits of the equation solving subroutine (which may be changed easily) and to the use of only five time steps.

8.3 THE ST. CLAIR RIVER

The third example given is that of flow on an actual river system. The actual river data are used and the results show the type of answers that can be expected from the implicit model presented here. In fact, the examples given are only a small portion of the work that has been done on the St. Clair/Detroit River system. This work will be published in a later report.

The St. Clair River joins Lake Huron and Lake St. Clair and is approximately 28 miles long, about 2,000 feet wide and approximately thirty feet deep. The river has been extensively dredged and is a major navigational waterway. The lower end of the river, where it empties into Lake St. Clair, is a complicated delta region with a very large amount of swampy ground and very shallow channels. Historically, the flow has varied between 100,000 and 240,000 cusecs with a 50 percent exceedence level of 184,000 cusecs. The U.S. Army Corps of Engineers have maintained an extensive water level gauging network on the river

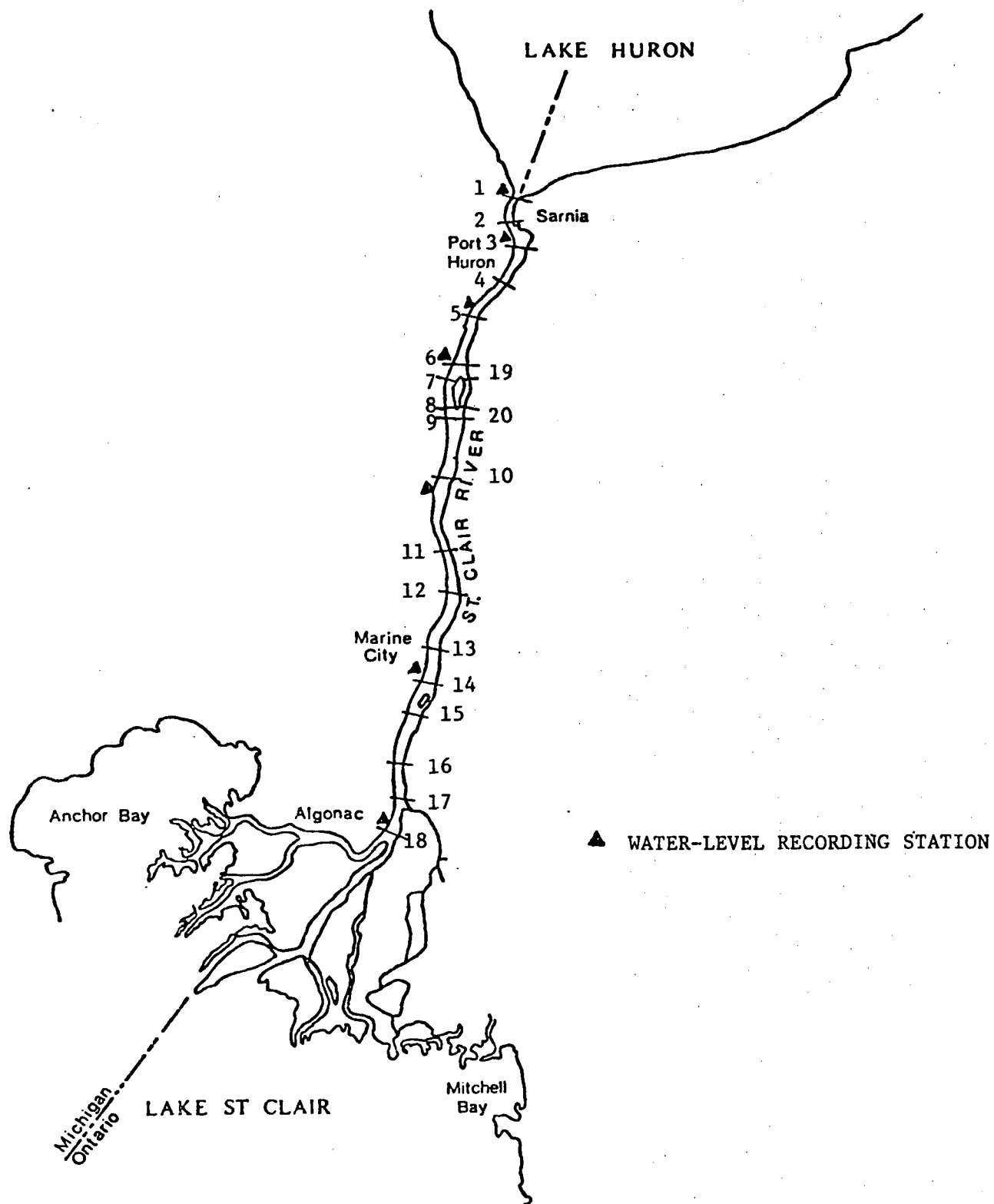


FIGURE 13: THE ST. CLAIR RIVER

for a number of years [10].

A map of the river giving the node numbering is shown in Figure 13. The geometrical data used in the model are given in Table 3. Examination of the field sheets shows that the cross-sectional shape of the river is approximated by a very wide rectangle and as a result, the hydraulic radius may be expressed as the mean depth (Cross-sectional Area/surface width). The river contains one major island, Stag Island, which is included in the model. The upper limit of the model is at Fort Gratiot and the lower limit is taken at Algonac, at the top of the delta region. No attempt was made to model the flow in the delta region since the section geometry of this region completely invalidates the hypothesis of one dimensional flow and since there are not enough water level data to properly calibrate a model.

A major problem in modeling a natural river lies in the choice of a friction factor. In the case of the St. Clair River, a great deal of information is available which allows for a very fine calibration. The fact that changing the friction factor in a given reach affects only the flow upstream of that reach is a useful point in calibration processes.

The calibration period chosen for the St. Clair River was the flow on 19 June, 1973. On this date, discharge measurements were being taken in the river by the U.S. Army Corps of Engineers at St. Clair, Michigan. The mean flow on this day was 215,000 cusecs. From this start, the calibration procedure was:

1. Use a discharge of 215,000 cusec and a known water level at Algonac.
2. Start with a Mannings friction factor of 0.020, which from practical experience is too low for this river.
3. Make a computer run and compare the calculated water levels with the observed water levels upstream.
4. Adjust the friction factor in the lowest sections of the model until the calculated water level at the second gauge upstream agrees with the observed water levels.

NODE NO.	TOPWIDTH	INVERT LEVEL BELOW DATUM	REACH LENGTH	MANNING'S n	WATER LEVEL GAUGE NAME
1	1000	34.99	7220	0.033	FT. GRATIOT
2	1300	35.10	3825	0.028	
3	2870	23.11	4200	0.023	MBR
4	2200	31.76	12700	0.023	
5	1950	28.70	12250	0.023	DRY DOCK
6	2060	29.81	3900	0.020	MARYSVILLE
7	2670	31.43	20200	0.020	
8	2200	23.09	100	0.020	
9	2200	27.10	18200	0.020	
10	1620	31.18	6050	0.025	ST. CLAIR, MICHIGAN
11	1700	35.62	9000	0.025	
12	2260	29.90	6200	0.025	
13	2780	27.07	9500	0.024	
14	3300	23.46	10950	0.023	MARINE CITY
15	2850	28.07	9200	0.023	
16	2260	33.14	9050	0.025	ROBERT'S LANDING
17	2240	33.12	6100	0.022	
18	2760	28.39	-	-	ALGONAC
19	1500	16.94	10200	0.020	
20	1530	17.07	6300	0.020	

TABLE 3: GEOMETRICAL DATA FOR ST. CLAIR RIVER SYSTEM
DATUM IS 576.80 (IGLD).

5. Move upstream in this fashion, adjusting friction factors for one section at a time until all of the computed water levels agree with the observed water levels within a certain predetermined accuracy;
6. Using two observed water levels and the determined friction factors, make another computer run to check that the discharge is correct. If it is not, adjust all friction factors up or

down by the same relative amount until the computed and observed discharges match properly.

Using this procedure, the model was calibrated for June 19, 1973 to within ± 0.08 feet as the maximum error in water level. This accuracy is certainly within the expected limits of any one dimensional model, assuming steady-state flow conditions and considering the practical accuracy of discharge measurements. The calibration process used about 20 computer runs to match up the 9 water level gauges on the river. Although the process of calibration could easily be incorporated into the computer program, "intelligent guesses" allow a great savings in computer time.

After the model was calibrated for the 19 June, 1973, two additional runs were made to show that the model would predict properly. The discharge for 19 June, 1973, was approximately 215,000 cusecs and this corresponds to an extremely high flow condition. The additional runs were: 18 July, 1968 which corresponds to a discharge of approximately 183,000 cusecs; a "medium" flow condition; and 13 November, 1964, a "low" flow condition with a discharge of approximately 147,000 cusecs. The observed water levels, computed water levels and differences are shown in Table 4. Unfortunately, not all of the water level gauges were in operation all of the time. However, a good indication of the accuracy of the model is given by this table.

Note that the two tables are referred to different datums. The abstraction of data was done and the model run with an arbitrary horizontal datum of 576.80 feet. The results, however, are presented at a datum of 570.00 feet to avoid the use of minus signs in the table.

To show that the model properly predicts transients, the model was run for a period during which a large transient passed through the St. Clair River. During December 3 - 5, 1970, a storm was in progress and this storm resulted in a three foot storm surge on Lake Huron at Fort Gratiot. At that time, three gauges were in operation in the St. Clair River; one at Fort Gratiot, one at St. Clair, Michigan, and one at Algonac. Data are available from these gauges at hourly intervals.

A run was made using, as boundary conditions, the observed water levels at Fort Gratiot and at Algonac. The schematization was

NODE NO.	JUNE 19, 1973			JULY 18, 1968			NOV 13, 1964			MANN- ING'S n
	OBS.	COMP.	DIFF.	OBS.	COMP	DIFF.	OBS.	COMP	DIFF.	
1	10.21	10.21	-	7.91	7.91	-	5.00	5.00	-	0.033
2		9.74			7.49			4.63		0.028
3	9.68	9.72	+0.04	7.51	7.45	-0.06	4.72	4.57	-0.15	0.023
4		9.64			7.38			4.51		0.023
5	9.33	9.27	-0.06	7.09	7.05	-0.04		4.23		0.023
6	8.87	8.95	+0.08	6.73	6.77	+0.04	3.97	3.97	-	0.020
7		8.95			6.77			3.97		0.020
8		8.70			6.53			3.74		0.020
9		8.70			6.53			3.74		0.020
10	8.13	8.13	-	6.03	6.03	-	3.30	3.30	-	0.025
11		7.99			5.91			3.19		0.025
12		7.69			5.72			3.01		0.025
13		7.65			5.60			2.89		0.024
14	7.36	7.42	+0.06		5.38			2.66		0.023
15		7.19			5.16			2.45		0.023
16	6.92	7.00	+0.08	4.96	5.00	+0.04		2.31		0.025
17		6.85			4.87			2.19		0.022
18	6.75	6.75	-	4.78	4.78	-	2.10	2.10		0.020
19	8.87	8.95	+ .08		6.77			3.97		0.020
20		8.70			6.53			3.74		0.020
COMP DISC	215,000			183,000			147,000			

TABLE 4: OBSERVED AND COMPUTED WATER LEVELS FOR THE ST. CLAIR RIVER
UNDER VARIOUS DISCHARGES. DATUM IS 570.00 FT. (IGLD).

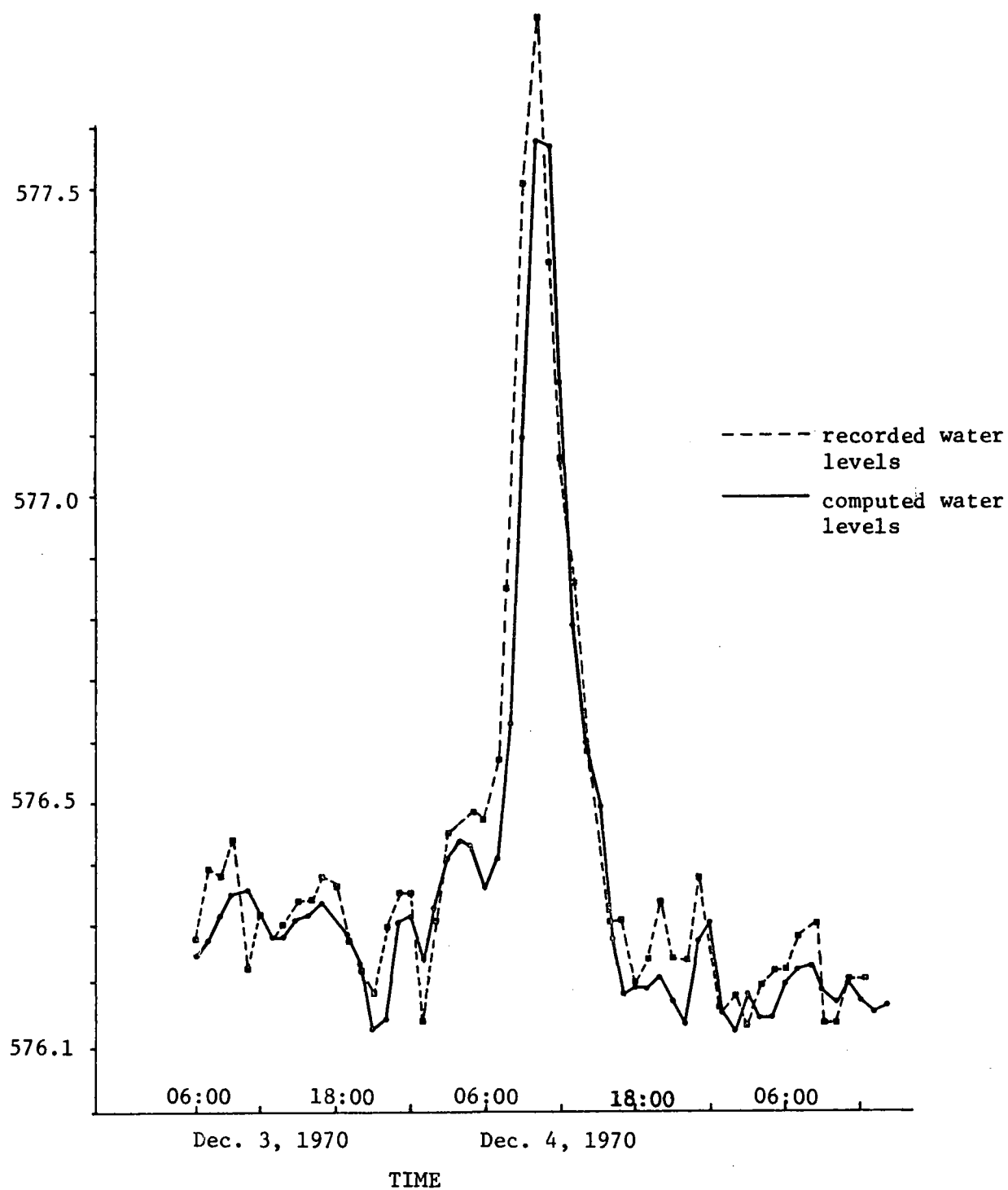


FIGURE 14: COMPUTED VS. OBSERVED WATER LEVELS IN ST. CLAIR RIVER

revised to use only 13 nodes and all data from the steady state runs were retained. Figure 14 shows the observed values compared with the calculated values for node 10 (St. Clair, Michigan). The agreement is quite close, considering that only hourly water levels are available. This time step is clearly too long to give adequate resolution in the phase of the transient.

CHAPTER 9

9.0 SUMMARY AND CONCLUSIONS

9.1 SUMMARY

A scheme has been devised for solving any set of equations describing open channel flow for any network of open channels. These equations must describe one dimensional subcritical flow of a homogenous fluid. The shallow water equations were derived from first principles and the derivation explicitly shows the assumptions made for flow in elementary reaches and in junctions. The stability of the finite difference scheme was examined for the linear case and a method for the solution of a generalized set of nonlinear algebraic equations was described. The application of these techniques to three sample problems was outlined.

A summary of the results follows:

1. By describing a network of open channels in terms of graph-theoretic concepts, the flow relationships in any network may be described in terms of relationships between only four types of nodes.
2. The implicit method is extended to handle flow in networks of open channels.
3. The derivation of the shallow water equations for an elementary reach assumes homogenous flow, one dimensionality, and fixed section geometry only.
4. The derivation of the junction equations assumes that the junction nodes are very close together, that the Bernoulli terms of the momentum equation are not important, and that turbulence and centrifugal accelerations may be neglected.
5. A generalized expression for the stability factor $|\lambda|$ is developed using the Von Neumann technique applied to the linearized finite difference equations.
6. When the weighting factor, θ , for the finite difference scheme is $0.5 \leq \theta \leq 1.0$, the implicit four point finite difference scheme is unconditionally for the linear case. When $0.0 \leq \theta \leq 0.5$, the implicit four point finite difference scheme is conditionally stable.

7. The truncation error, E , demonstrates that the implicit four point scheme is consistent since it approaches zero as the time and distance steps are reduced.
8. The truncation error demonstrates that for linear equations the box scheme, $\theta=0.5$, has second order accuracy and this accuracy decreases to first order as θ approaches unity.
9. The box scheme conserves mass during one time step.
10. The equation solving algorithm requires that no critical section appears within the network, that flow is physically possible, and that an initial guess to the solution be provided "reasonably" close to the actual solution.
11. The equation solving algorithm does not depend upon the form of the equations and so any set of equations, such as kinematic wave equations, may be used with this computer model.
12. The model has been successfully applied to a number of specific problems. The numerical examples bear out the analytical predictions.

9.2 ADVANTAGES AND DISADVANTAGES

The advantages of this particular model over others reported in the literature are:

1. The computer program is independent of the network topology.
2. The program is independent of the location of the boundary conditions.
3. The program is independent of the form of the governing equations.
4. The program handles all types of nonlinear terms.

Balanced against these advantages is the single disadvantage that the computer program takes an inordinately long time to run. The equation solving subroutine makes no use of the fact that the coefficient matrix is sparse and requires approximately $2N^2 + \frac{N}{3}$ function evaluations at each iteration, where N is the number of unknowns for which solution is required. If advantage 3 is not necessary, then a method exists for using sparse matrix techniques to solve the equations. This method has been tested with a two dimensional implicit model [8] and will result in a time saving of at least one order of magnitude. In the two dimensional model, 1,200 equations are

solved in 1/10 of a second in the IBM 370. The solution technique used in that model requires N function evaluations for N unknowns and hence is an order of magnitude faster than the technique used in this model.

9.3 CONCLUSIONS

The computer model developed in this report is a perfectly feasible method for solving unsteady flow problems in networks of open channels. This method is not dependent upon the particular form of the equations of motion. The restrictions on the program are very simple and require only that flow may be assumed to be one-dimensional and that no critical section appears in the network. The practical aspects of running the computer program and, more importantly, changing the program to handle different cases, are discussed fully in the Appendices.

REFERENCES

1. Amien, M., "An Implicit Method for Numerical Flood Routing", Water Resources Research, Vol. 4, No. 4, 1968, pp. 719-726.
2. Amien, M., "Computation of Flow through Masonboro Inlet, N. C.", Journal of the Waterways Harbours and Coastal Engineering Division, ASCE, Vol. 101, No. WW1, Proceedings Paper 11124, February, 1975, pp. 93-107.
3. Amien, M., and Chu, H. L., "Implicit Numerical Modeling of Unsteady Flows", Journal of the Hydraulics Division, ASCE, Vol. 101, No. HY6, Proceedings Paper 11378, June, 1975, pp. 717-731.
4. Amien, M., and Fang, C. S. "Implicit Flood Routing in Natural Channels", Journal of the Hydraulics Division, ASCE, Vol. 96, No. HY12, Proc. Paper 7773, Dec., 1970, pp. 2481-2500.
5. Baltzer, R. A., and Lai, C., "Computer Simulation of Unsteady Flows in Waterways", Journal of the Hydraulics Division, ASCE, Vol. 94, No. HY4, Proceedings Paper 6048, July, 1968, pp. 1083-1117.
6. Brown, K. M., "Solution of Simultaneous Non-Linear Equations, Algorithm 316", Communications of the ACM, Vol. 10, No. 11, Nov., 1967, pp. 728-729.
7. Brown, K. M., "A Quadratically Convergent Newton-Like Method Based Upon Gaussian Elimination", SIAM Journal of Numerical Analysis, Vol. 6, No. 4, Dec., 1969, pp. 560-569.
8. Chandrashekhhar, M., Muir, L. R., and Unny, T. E., A Numerical Two Dimensional Flow Model for River Systems, Proceedings of MODELING 75 Symposium, ASCE Specialty Conference, September, 1975, San Francisco, U. S. A.
9. Chow, V. T., Open Channel Hydraulics, McGraw-Hill Book Co., Inc., New York, N. Y., 1959.
10. Cox, P., U. S. Army Corps of Engineers, Detroit District, Personal Communication.
11. Dronkers, J. J., Tidal Computations in Rivers and Coastal Waters, North-Holland Publishing Co., Amsterdam, 1964.

12. Fread, D. L., "Effects of Time Step Size in Implicit Dynamic Routing", Water Resources Bulletin, Vol. 9, No. 2, April, 1973, pp. 338-351.
13. Fread, D. L., "Technique for Implicit Dynamic Routing in Rivers with Tributaries", Water Resources Research, Vol. 9, No. 4, August, 1973, pp. 918-926.
14. Fread, D. L., "Numerical Properties of Implicit Four-Point Finite Difference Equations of Unsteady Flow", NOAA Technical Memorandum NWS HYDRO-18, March, 1974.
15. Fread, D. L., "Computation of Stage-Discharge Relationships Affected by Unsteady Flow", Water Resources Bulletin, Vol. 11, No. 2, April, 1975.
16. Garrison, J. M., Granju, J. P., and Price, J. T., "Unsteady Flow Simulation in Rivers and Reservoirs", Journal of the Hydraulics Division, ASCE, Vol. 95, HY5, 1969.
17. Henderson, F. M., Open Channel Flow, The Macmillan Co., New York, N. Y., 1966.
18. Leendertse, J. J., "Aspects of a Computational Model for Long-Period Water Wave Propagation", Rand Memorandum RM-5294-PR, Rand Corp., Santa Monica, Calif., 1967.
19. Price, R. K., "Comparison of Four Numerical Methods for Flood Routing", Journal of the Hydraulics Division, ASCE, Vol. 100, No. HY7, Proc. Paper 10659, July, 1974, pp. 879-899.
20. Quinn, F. Y., and Wylie, E. B., "Transient Analysis of the Detroit River, by the Implicit Method", Water Resources Research, Vol. 8, No. 6, Dec., 1972, pp. 1461-1469.
21. Stoker, J. T., Water Waves, Interscience, N. Y., 1957.
22. Strelkoff, T., "Numerical Solution of Saint-Venant Equations", Journal of the Hydraulics Division, ASCE, Vol. 96, No. HY1, Proc. Paper 7043, Jan., 1970, pp. 223-252.

APPENDIX 1 USER'S GUIDE

This section deals with the computer implementation of the theoretical investigation described in the previous section.

The computer program is designed to handle all problems of open channel flow in networks where the following assumptions are valid:

1. Flow is one-dimensional.
2. Geostrophic and wind-driven circulation are not important.
3. A quadratic resistance law such as Manning's or Chezy is appropriate.
4. The geometry of the open channel network is constant with time, that is, no deposition or scouring occurs.
5. Hydrostatic pressure prevails.
6. The section geometry can be schematized as rectangular in cross-section.
7. Flow is entirely sub critical.
8. Flow is homogenous in density.

The program implementing this theoretical investigation is MOD*. It is programmed in FORTRAN IV for a CDC 3170 computer using a MASTER operating system. All remarks about the program refer to this version.

The program requires 74 quarter pages of storage space on the CDC 3170. This is equivalent to 37,888 real words. The word size on the CDC 3170 is 48 bits. Integer size is 24 bits. Double precision is not required for this word size but probably would be required for a smaller word size. Compilation time for the program is approximately 21 seconds.

A card reader and a 136 character line printer are the only required peripherals.

USING THE MODEL

1. Schematization

The first step in using the model is to schematize the open channel network as a directed graph. A node will appear on the graph at every location where a solution is required. The nodes are numbered from 1 to N, where N = number of nodes in the schematization. A main channel is chosen and the nodes in this main channel are numbered in sequence from upstream to downstream. For example, figures 9, 14 and 15 show some examples of graphs which are numbered correctly.

2. Abstraction of Data

All elevations used in the model must be referred to a HORIZONTAL datum. This datum may be at any level except that two conventions must be followed with regard to sign.

1. Z = water surface elevation above datum. If the initial water surface is above datum, the numerical value must be positive.
2. H = elevation of channel invert (bottom) below datum. If the channel invert is below the horizontal datum the numerical value must be positive. If the channel invert is above the datum, the numerical value must be negative.

Top width of the channel is determined from the cross-sectional appearance of the channel. The hydraulic mean depth is calculated by dividing the cross-sectional area by the top width. If it is necessary to allow arbitrary section geometry see the section on Program Restrictions further on this paper.

3. Input Cards

When all data are abstracted, a separate DATA card should be filled out for each node in the system. The format of this DATA card is as follows:

Columns 1 - 4 - <u>DATA</u> ,	Format A4
6 - 7 - Section number; the number of the node to which this card refers.	Format I2
9 - 12 - the initial velocity at this node.	Format F4.2

- 14 - 18 - the initial water level at this node with respect to the arbitrary horizontal datum, following the convention stated above.
- 20 - 24 - the top width at this section. Format F5.1
- 26 - 30 - the distance between the horizontal datum and the channel invert. Format F5.2
- 32 - 38 - the length of reach between this node and the next node downstream. Format F7.1
- 40 - 44 - the lateral inflow per unit length of reach between this node and the next node downstream. This value must be positive in the case of inflow and negative in the case of outflow. Format F5.3
- 46 - 49 - the average friction coefficient between this node and the next node downstream. If this value is greater than 1.0 the Chezy friction equation is used and if the value is less than 1.0, Mannings friction formula is used. Format F4.0

The MAIN and BRANCH cards set up the geometrical properties of the open channel system and provide boundary conditions for the model.

The format for the MAIN card is:

Columns 1 - 4 - MAIN

- 14 - 15 - the largest node number in the main channel. Format I2
- 24 - 25 - node number at which the first boundary condition is given. Format I2
- 33 - 35 - the type of the first boundary condition. May be VELØ, HEIG, DISØ if the boundary condition is fixed velocity, or height or discharge respectively. Format A4
- 44 - 45 - the node number of the second boundary condition. Format I2
- 53 - 55 - the type of the second boundary condition. May be VELØ, HEIG, DISØ. Format A4

Only one MAIN card may appear in a program.

One BRANCH card must appear for each distinct Branch in the network. The format is:

Columns	1 - 4 -	<u>BRAN</u>	Format A4
13 - 14 -	the smallest node number appearing in the branch itself.		Format I2
19 - 20 -	the largest node number appearing in the branch itself.		Format I2
29 - 32 -	code word describing whether the top end of the branch receives flow from another part of the network or stands alone as a boundary condition.		
	If the code word is <u>FROM</u> , FORMAT A4, then the flow is from another part of the river. The node number from which the flow is received is placed in Column 34 - 35, FORMAT I2.		
	If the code word is <u>FIXE</u> , FORMAT A4, then that node stands alone as a boundary condition. The type of boundary condition is given as <u>VELØ</u> , <u>HEIG</u> , or <u>DISØ</u> in columns 35 - 38, FORMAT A4.		
52 - 55 -	code word describing whether the bottom end of the branch provides flow to another part of the network or stands alone as a boundary condition.		
	If the code word is <u>TOØØ</u> , FORMAT A4, the node number to which flow is delivered is placed in Columns 57 - 58, FORMAT I2.		
	If the code word is <u>FIXØ</u> , FORMAT A4, the type of boundary condition is given as <u>VELØ</u> , <u>HEIG</u> , or <u>DISØ</u> in columns 58 - 61 FORMAT A4.		

The WEIGHT card provides miscellaneous data required by the computer program. All values on this card are provided by default in the Main Program. Therefore, the card does not have to be provided if the default values are acceptable. Any of the values on the card may be left blank if desired. The format is:

Columns	1 - 4 -	<u>WEIG</u>	Format A4
---------	---------	-------------	-----------

- 9 - 11 - weighting factor for the finite-difference scheme.
Default value is 0.75, Format F3.3
- 22 - 25 - acceleration due to gravity. Default value is
32.172. Format F4.2
- 35 - 36 - maximum number of iterations to be performed by
NONLIN. Default value is 10, Format I2
- 47 - 48 - number of significant digits to which NONLIN
produces solutions. Default value is 5. Format I2

The WEIGHT card may appear anywhere before the 1st RUN card in the Data Deck.

The RUN card controls the running of the Model in a steady-state configuration. All DATA, MAIN, BRANCH and WEIGHT cards must appear before the RUN card. The format is:

Columns 1 - 4 - RUN

- 18 - 19 - the highest node number appearing in the
schematization, Format I2
- 31 - 33 - the length of time step to be used. Format I3
- 35 - 37 - the units used for the time step. May be SEC, MIN
or HRS. Format A3
- 49 - 53 - the time at which the run is to be stopped.
Format I5
- 55 - 57 - the units used for the stop time. May be SEC, MIN
or HRS. Format A3

The VABC card allows time varying boundary conditions to be applied to the model. A RUN card must precede the VABC card so that the model may come to a proper steady state with all of the arrays properly initialized. The types and locations of boundary condition equations may not be changed with the VABC card. The card format is:

Columns 1 - 4 - VABC

- 15 - 17 - length of time step to be used under variable
boundary conditions. May be different from the time
step appearing on the RUN card. Format I3
- 19 - 21 - the units used for the time step. May be SEC, MIN
or HRS. Format A3

- 32 - 36 - Stop time. The length of time through which the model calculates; begins with the first calculation done on the Run card. Format I5
- 38 - 40 - Units used for stop time. Format A3
- 48 - 76 - Boundary conditions codes. Up to 5 individual nodes may have time dependent Boundary conditions. These are inserted on the card as pairs of integers. The first value in each pair is the node number at which the boundary condition is specified and the second is a code which provides the boundary condition information. The format for these is 5(I2, 1X, I2, 1X). The codes used are:
- 01 - height is given by a function
 - 02 - velocity is given by a function
 - 03 - discharge is given by a function
 - 04 - height is read in from data cards
 - 05 - velocity is read in from data cards
 - 06 - discharge is read in from data cards

Calculation of boundary conditions specified by codes 01, 02, and 03 is done in subroutine CALCBC and must be user supplied. The reading in of data cards specified by codes 04, 05, and 06 is done in subroutine READBC and must also be user supplied.

The COMMENT cards allow comments to be interspersed with the data. It causes no action in the program. Format is:

Columns 1 - 4 - Cb/bb

5 - 80 - any alpha numeric string

The PRINT card prints out tables of values at the end of a run. Each table starts on a new page and is neatly listed for plotting purposes. This card is not required if the tables are not wanted. The four tables printed out are: heights, velocities, discharges and Transit Time. Each table contains calculated values for each reach for each time step. The PRINT card allows the heights to be referenced to

a geodetic elevation and allows the time to be printed out in seconds, minutes or hours. The format is:

Columns	1 - 4 -	<u>PRIN</u>	Format A4
19 - 20 -	the highest node number for the table. The program will print values for all nodes from 1 up to the specified value.		Format 12
30 - 32 -	the increment in which the time is to be printed out. May be SEC, MIN, or HRS. Default value is in seconds.		Format A3
43 - 47 -	the elevation of the arbitrary datum. This value is added to the elevations determined by the program which are referred to the arbitrary horizontal datum of 0.00.		Format F5.2

4. Practical Considerations

There are a number of practical considerations which must be taken into account when running the program. These are listed here for easy reference.

UNITS: Any system of units may be used for the model as long as they are consistent. For example, all length units may be in centimetres, metres, kilometres, feet or inches. If the units are not in feet, however, the appropriate gravitational constant must be inserted on the WEIGHT card.

TIME: Any convenient time units may be used on input. The program works in seconds and all output from subroutines RUN or VABC are similarly in seconds.

WEIGHTING FACTOR: The weighting factor is exhaustively discussed in Chapter 6. Briefly, however, the finite-difference scheme is numerically stable for $0.50 \leq \text{WHT} \leq 1.0$. The rate of convergence of the scheme is directly proportional to the weight. In steady state flow, the weight affects only the rate of convergence of the numerical scheme. In time-varying flow, the weight may introduce phase lags and the smallest

practicable weight (0.55) should be used. The scheme will not conserve mass if the weight is different than 0.50. The extent to which the scheme does not conserve mass is directly proportional to the departure of the weight from 0.50 and should not exceed about 5% even if the weight is increased to 1.0.

TIME STEP: The time step to be used in the computation is fairly critical for efficient computer usage. The numerical scheme is stable regardless of the time step. Too large a time step results in inaccuracy of resolution and too small a time step results in wasted computer time. The most useful guide is to use a time step approximately equal to the shortest reach length divided by kinematic wave speed. The kinematic wave speed is $v + \sqrt{gh}$ where v = initial velocity in the reach, g = acceleration due to gravity and h = depth of water in the reach.

NUMBER OF NODES: The time required to complete the computations for one time step is a function of $(4N^2 + \frac{2N}{3})$, where N = number of nodes in the schematization. Therefore, a large number of nodes in the schematization can easily result in excessive use of computer time. It is recommended that no more than 14 nodes be used at any one time if possible. A revision of the equation solving subroutine may be possible. This revision would make use of the fact that the coefficient matrices are sparse and execution time should be speeded up dramatically.

INITIAL CONDITIONS OR BOUNDARY CONDITIONS: The initial conditions used by the subroutine NONLIN for the solution of the equations at a particular time step are those values computed at the previous time step. The initial conditions for the first time step are input on the DATA cards. These initial conditions may be simply calculated using either Manning's equations or

Chezy equations or may simply be estimated. If the system is physically possible the model will converge to a steady state within 8 or 9 time steps. A separate run to obtain good initial conditions is recommended before any variable boundary conditions are used.

The boundary conditions for the initial steady state run are input on the DATA cards as initial values. For example, if an initial height above datum at node 01 is specified on the DATA card, that height is the boundary condition and it remains constant throughout the time specified on the RUN card. A discharge boundary condition is specified by requiring the initial discharge, i.e. the product of the initial velocity and the cross-sectional area, be equal to the required discharge. The initial height and velocity may change throughout the run, but the discharge will remain constant.

NON-RECTANGULAR SECTION GEOMETRY: The program is presently set up for rectangular cross-sections. This need not be the case in a natural channel and can be overcome. If the assumption of a rectangular channel does not hold, then the following changes must be made before utilizing the model:

1. The equations of motion must be used in their most general form and finite-difference equations rewritten.
2. Use of a subroutine such as PROPS, from the CIVLIB package, must be incorporated to calculate hydraulic radius and cross-sectional area within the function subroutine FUN.

The programming changes are easily made; however, the price will be paid through increased computation time. The degree of accuracy obtained through these refinements is seldom warranted for large scale natural systems.

5. Output

- 1: START RUN
- 2: Should be a printout of all data cards up to and including the first RUN card.

- 3: One line containing weighting factor, gravitational constant, maximum number of iterations used, and number of significant digits used.
- 4: MODEL SET UP
 One line containing indexes (up to twice the number of sections).
 One line containing the equation types used.
 One line containing the links
- 5: For each time step.
 All calculations complete at time = _____seconds-----
 No. of iterations = _____
 One line containing velocities at each section
 One line containing heights above datum at each section
- 6: Print card, if present
- 7: Four tables containing height vs. time, velocity vs. time, discharge vs. time, and transit time vs. time.
- 8: ENDRUN
- 9: STOP

DIAGNOSTICS:

Apart from the diagnostics put out by the computer the model also does a certain amount of error checking. These diagnostics are listed in the order in which they occur in the listing of the program.

<u>Message</u>	<u>Meaning</u>
Error - Incorrect Statement Type	The last data card being read in has errors in the first four characters. Repunch the card.
More than 100 time steps called for.	The stop time on the VABC card is too large. Either arrange the time steps and the stop time so that less than 100 time steps are required, or dimension VEL (I,J), RHGT (I,J), DIS (I,J), FR (I,J) to the appropriate

Message

Solution matrix singular at
Iteration_____, time is_____
seconds ---- run stopped.

Warning -- Solution may not be
accurate, time = _____

Warning -- Check weights --
Solution may be unstable

Incorrect key word in boundary
conditions Stop Run.

Improper equation type -- Run
stopped

Data point number = _____ in
subroutine_____ Run stopped.

Meaning

dimensions and remove card num-
ber VBC 450 from subroutine VABC

The equations cannot be solved
with this combination of initial
conditions, boundary conditions
and equations. - Make sure you
have a physically realizable
situation.

Nonlin took the full number of
iterations allowed and has not
found a solution to the equations.
- Make sure you have a physically
realizable situation and increase
the number of iterations allowed.

The weighting factor in the finite-
difference scheme is too small and
the solution will probably be un-
stable. Increase the weighting
factor to 0.5 or greater.

Punching error on BRANCH card.

One of the subroutines DATA, MAIN,
BRANCH has set up the wrong equation
type into array IEQU. This warning
is called by subroutine FUN.

The named subroutine has determined
that you are attempting to use more
than 20 sections in the model.
- Re-dimension all of the COMMON
and change subroutine CHECK if you
must use more than 20 sections.

6. Narrative Description of Subroutines

MAIN PROGRAM:

Description: The main program provides default values for system parameters, reads in data cards, and calls subroutines.

Variables: WHT = weighting factor in time
 ISIGDIG = number of significant digits for equation
 solution
 MAXSA = maximum number of iterations by NONLIN
 GRAV = acceleration due to gravity
 ISW = switch used by VABC to disable first part
 of RUN
 TYPE = first 4 characters of data cards, used to
 call the appropriate subroutine. TYPE may
 be one of: DATA, CØØØ, BRAN, MAIN, PRIN,
 RUNØ, VABC, WEIG
 CARD = last 76 characters of data card. This array
 is decoded in the appropriate subroutine

Notes: WHT, ISIGDIG, MAXSA, and GRAV may be reset by subroutine WEIGHT.
 ISW will be reset by subroutine VABC.

Subroutine RUN

Description: This subroutine runs the model in a steady state condition. All equation types, initialization, and steady-state boundary conditions have been set up in other subroutines. The subroutine is in two parts. The first part decodes the RUN card, prints out the model setup, changes all time reference to seconds, and initializes variables. This section is done once for every appearance of a RUN card in the data deck. If RUN is called by VABC, ISW = 1 and this first part is skipped.

The second part of subroutine RUN is a large DO loop, iterating on the number of time steps. It places the present time value in arrays RHGT, VEL, DIS, and FR: calculates all coefficients involving only the initial

conditions for the present time step; calls NONLIN to solve the equations; checks that a proper solution has been obtained; inserts the solutions into the proper arrays; and prints out the solutions.

Variables:

ISW	=	A switch which disables the first part of the subroutine RUN. The first part of RUN is only required the first time RUN is called in a steady state condition.
IPR	=	highest section number in the model.
IDELT	=	integer value of time step.
INC	=	units used for time step. May be SEC, MIN, HRS.
ISTPT	=	integer value of stop time.
ISP	=	units used for stop time. May be SEC, MIN, HRS.
V	=	one dimensional array containing velocities. On entry to a time step contains initial values for that time step.
Z	=	one dimensional array containing water level above datum. On entry to a time step, it contains the initial values for that time step.
H	=	one dimensional array containing depth of water below datum at each section.
B	=	one dimensional array containing width of channel at each section.
QL	=	one dimensional array containing lateral inflow per unit length of reach for the reach between section I and section I + 1.
DK	=	one dimensional array containing friction coefficient for the reach between section I and section I + 1. If $DK > 1.0$, the Chezy resistance law is used; if $DK < 1$, the Manning resistance law is used.

X = one dimensional array used by NONLIN. Contains V and Z arrays as follows: V(I), Z(I), V(2), Z(2), V(3), Z(3) ... V(IPR), Z(IPR).
 WHT = weighting factor for the finite-differencing scheme.
 WHTN = 1. - WHT.
 IRUNS = counter giving present run number.
 ITIME = present time value.
 RHGT = two dimensional array containing water levels above datum at each section for each time step.
 VEL = two dimensional array containing velocities at each section for each time step.
 DIS = two dimensional array containing discharge at each section for each time step.
 FR = two dimensional array containing the Froude number at each section for each time step.

HTS
 HN
 VP
 ZØ
 DZ
 DV
 ZP
 CB
 CC
 CD
 CE

= various coefficients for the momentum and continuity equations using only initial values.

MAXIT
 MAXSA

= maximum number of iterations allowed to NONLIN.

ISING = a flag returned by NONLIN. If the Jacobian matrix is singular, ISING = 1 and no solution is possible. If ISING = 0 then a solution is obtained.

ISIGDIG = number of significant digits to which NONLIN solves the equations of motion.

Subroutine VABC

Description: This subroutine runs the model under time varying boundary conditions. A preliminary call to subroutine RUN ensures that all arrays are properly filled and that the model has converged to a proper steady state. The subroutine is in 3 parts.

The first part decodes the VABC card, changes all time references to seconds, and determines the number of runs required. The second part calls either CALCBC or READBC to input boundary conditions and stores them in the proper arrays. The third part extracts the proper boundary conditions for a particular time step, calls RUN for that time step, and iterates until the stop time has been reached.

Variables:

IDELT	=	the length of time step while VABC is used.
INCD	=	time increment used for inputting time step.
ISTOP	=	stop time, the run will stop when this time has been reached.
INCS	=	time increment used for inputting stop time.
ISECT	=	a one dimensional array containing the node numbers where the boundary conditions occur.
ICODE	=	a one dimensional array containing the types of variable boundary conditions at the corresponding node number.
IRUNS	=	the present run number.
NRUNS	=	the last run number.
ISW	=	a switch, set in this subroutine to 1, to disable the first part of subroutine RUN.
XBC	=	a two dimensional array containing the boundary conditions. This array is filled in subroutines READBC and CALCBC.
Z(IST)	=	the depth of water above datum at node number IST. A boundary condition.
V(IST)	=	the velocity at node number IST. A boundary condition.

Subroutine CALCBC

Description: A subroutine which must be changed by the user to allow the boundary conditions to be specified as a functional relationship. It calculates the boundary conditions for the complete run at one time and stores them in the proper array, i.e. the Ith boundary condition is calculated from time step IRUNS to time step NRUNS. The Ith boundary condition for run number IR is placed on XBC (IR,I).

Subroutine READBC

Description: Another subroutine which must be user supplied to allow the boundary condition to be read in from cards. The Ith boundary condition for run number IR is placed on XBC (IR,I).

Subroutine DATA

Description: This subroutine reads the DATA card. Each DATA card contains the section geometry, lateral inflow and resistance coefficient for a given section. This subroutine also sets up initial equation types in array IEQU and assigns initial value to LINK. The equation types and links are modified to take boundary conditions into account in subroutine MAIN and junction conditions in subroutine BRANCH.

Variables:

I	=	section number to which DATA card refers.
V(I)	=	initial velocity at the section.
Z(I)	=	initial water level above datum. + ve if bottom is below datum. - ve if bottom is above datum.
DELTX(I)	=	length of reach between section I and I + 1.
QL(I)	=	lateral inflow per unit reach length between section I and section I + 1.
DK(I)	=	resistance coefficient between section I and section I + 1.

IEQU = one dimensional array containing equation types. The two equations for the reach between section I and I + 1 are stored in IEQU (2*I-1) and IEQU (2*I). This subroutine sets IEQU (2*I-1) = 1 and IEQU (2*I) = 2. The boundary conditions are set in subroutine MAIN and the junction conditions are set in subroutine BRANCH.

LINK = a one dimensional array containing section numbers. If section I is involved with section K, for example, then LINK (2*I-1) or LINK (2*I) would contain the value K. These values are changed in subroutine MAIN and subroutine BRANCH.

Subroutine MAINB

Description: This subroutine processes the MAIN card and sets up the boundary conditions for the main channel. These boundary conditions may be set at any section in the main channel. Values of IEQU and LINK which are set in this subroutine override those set by subroutine DATA. All DATA cards must precede the MAIN card.

Variables:

I	=	largest section number in main branch. The program assumes that the main branch starts at section 1.
IU	=	section number at which the upstream boundary condition occurs.
ICA	=	type of boundary condition at upstream end. May be <u>VEL</u> ϕ , <u>HEIG</u> , <u>DIS</u> .
ID	=	section number at which the downstream boundary condition occurs
ICB	=	type of downstream boundary condition.

The subroutine assigns the boundary condition equation types to IEQU (I-1) and IEQU (I). The section numbers at which the boundary conditions hold are placed in LINK (I-1) and LINK (I).

Subroutine BRANCH

Description: This subroutine processes the BRANCH card and sets up boundary conditions and/or junction conditions for the branch. One Branch card must appear for each distinct Branch in the model. The values of IEQU and LINK which are set in this subroutine override those set by subroutine DATA. All DATA cards must precede the BRANCH card.

Three basic types of branches are allowed in this model; every branch has at least one junction with the main channel or with another branch. The other end of the branch may be a junction or it may be a free boundary. Figure 14 shows these three types of branches.

IEQU and LINK are filled as follows:

1. At a junction with flow from/to IN, the two momentum equations (fixed height) are Type 6. These are stored in IEQU (2*IN-1) and IEQU (2*IN). LINK (2* IN-1) and LINK (2*IN) contain the two section numbers to which section IN is joined.
2. The remaining equation at a junction is stored in IEQU (2*IU-1) and IN is stored in LINK (2*IU-1) if IU is the upstream end of the Branch. If ID is the downstream end of the Branch, the continuity equation is stored in IEQU (2*ID) and IN is stored in LINK (2*ID).
3. If the upstream end of the Branch is a boundary condition, then the appropriate equation type is stored in IEQU (2*IU-1) and IU is stored in LINK (2*IU-1) if the upstream section number is IU.
4. If the downstream end of the Branch is a boundary condition, the section number is ID. Then, the appropriate equation type is stored in IEQU (2*ID) and ID is stored in LINK (2*ID).
5. In all uses the displaced continuity equation between sections IU and IU +1 is stored in IEQU (2*ID-1) and IU is stored in LINK (2*ID-1).

Note that the BRANCH card must come after the DATA cards since equation types set in subroutine BRANCH must override those set by subroutine DATA.

Variables: IS = upstream section number - the smallest
 section number in the branch.
 IE = downstream section number - the largest
 section number in the branch.
 ICA = a code describing the equation type for the
 upstream end of the branch.
 - if ICA = 4HFIXE then the upstream section
 has a boundary condition of type ITA.
 - if ICA = 4HFROM then the upstream section
 receives flow from section INL.
 ICB = a code describing the downstream equation
 type.
 - if ICB = 4HFIXE then the downstream section
 has a boundary condition of type ITB.
 - if ICB = 4HTØØØ then the downstream section
 donates flow to section number ØUT.

Subroutine WEIGHT

Description: This subroutine reads the WEIGHT card if one is present in the data deck. If no WEIGHT card is present or if any values on the card are zero or blank the default values assigned by the MAIN program are used. These default values are:

WHT = 0.75
GRAV = 32.2
MAXIT = 10.
ISIGDIG = 5

Subroutine PRINT

This subroutine prints out the arrays, RHGT, VEL, DIS, FR from section 1 up to the section number called from the PRINT card.

Function Subroutine FUN

Description: This function subroutine contains all of the equations describing the model and is called iteratively by subroutine NONLIN. At present, the model is set up with 8 types of equations.

Variables: K = argument set by NONLIN in the call to FUN.
IEQU(K) = equation type to be used with this call by NONLIN.
LINK(K) = section number involved in Kth call. The type of equation used is contained in IEQU(K).

Equations: The eight equation types used are:

1. Continuity equation between sections LINK(K) and LINK(K) + 1.
2. Momentum equation between sections LINK(K) and LINK(K) + 1.
3. Boundary condition - fixed velocity at section LINK(K).
4. Boundary condition - fixed height at section LINK(K).
5. Boundary condition - fixed discharge at section LINK(K).
6. Junction condition - fixed water levels at section $(K + 1)/2$ and LINK(K).
7. Divergent Junction condition - continuity equation involving sections $(K + 1)/2$, LINK(K) and LINK(K) + 1.
8. Convergent Junction condition - continuity condition involving $(K + 1)/2$, LINK(K) and LINK(K) + 1.

Subroutine NONLIN

Description: This subroutine solves the non-linear equations of motion of the system. It calls both FUN and subroutine BACK. This subroutine was originally presented in ALGOL[6]. It was transformed into FORTRAN for this program and reference should be made to the original literature for discussion of the methods used.

Variables:

- N = maximum number of equations to be solved,
i.e. 2* number of nodes
- MAXIT = maximum number of iterations allowed NONLIN.
The number actually used is returned.
- NUMSIG = number of significant digits to which answer
is to be accurate
- X = one dimensional array contains initial values
to be used as trail solutions. On return, X
contains the solution determined by NONLIN.
- FUN = name of function subroutine which contains the
equations $FUN(K) = K_{th}$ function evaluated at
 $X(1), X(2), \dots, X(N)$.
- SINGLE = value returned by NONLIN.
SINGLE = 0 if Jacobian matrix singular
SINGLE = 1 if proper solution obtained.

POINT (N,N) =	}	
ISUB (N-1) =		
TEMP(N) =		
PART (N) =		
COE (N,N-1) =		

working arrays

Subroutine BACK

This subroutine is called by NONLIN and is an integral part of that subroutine. It solves an upper triangularized linear system generated by NONLIN.

```

PROGRAM MODEL
C ***
C *** PROGRAM FOR THE ONE DIMENSIONAL MODELLING OF A RIVER NETWORK
C *** SOLVING THE COMPLETE NON-LINEAR ,SHALLOW WATER EQUATIONS
C *** USING A WEIGHTED IMPLICIT FINITE-DIFFERENCE SCHEME.
C ***
C *** DATE: AUGUST,1975
C *** COMPUTER CDC 3170 MASTER O.S.
C *** BY: L. R. MUIR
C *** P.O. BOX 5050
C *** BURLINGTON, ONTARIO
C *** CANADA
C ***
C *** MODEL READS CARDS AND DETERMINES WHAT TYPE THEY ARE
C *** INTEGER SIZE 1 AND * OPTION IS USED
C ***
C
      INTEGER TYPE,CARD(19),BRAN,COM,DAT,MAN,RU,PRI
      INTEGER POINT,VARBC
      COMMON/10/X(40),N,MAXSA,POINT(40,40),ISUB(39),TEMP(40),
*      PART(40),COE(40,41),ISIGDIG
      COMMON/20/IEQU(40),V(20),B(20),H(20),Z(20),DELT(20),QL(20),
*      DK(20),IPR,LINK(40),LIN(20),IDELT,ISTPT,RMS,GRAV,WHT
      COMMON/30/RHGT(100,21),VEL(100,21),DIS(100,21),FR(100,21)
      COMMON/50/ISW,IRUNS
      DATA BRAN/4HBRAN/,COM/4HC /,DAT/4HDATA/,MAN/4HMAIN/,
*      RU/4HRUN /,PRI/4HPRIN/,IWHT/4HWEIG/,VARBC/4HVABC/
C
C *****
C
C *** DEFAULT VALUES FOR WEIGHT CARD
      WHT=0.75
      MAXSA=10
      ISIGDIG=5
      GRAV=32.172
      ISW=0
      PRINT100
C
      02 READ200,TYPE,CARD
      IF(IFEOF(60).EQ.-1)GO TO 15
      ICARD=ICARD+1
      PRINT300,ICARD,TYPE,CARD
C ***
C *** TYPE DETERMINATION SECTION
C ***
      IF(TYPE.EQ.DAT)GO TO 07
      IF(TYPE.EQ.COM)GO TO 02
      IF(TYPE.EQ.BRAN)GO TO 08

```

```
IF(TYPE.EQ.MAN)GO TO 09
IF(TYPE.EQ.PRI)GO TO 16
IF(TYPE.EQ.RU)GO TO 10
IF(TYPE.EQ.VARBC)GO TO 11
IF(TYPE.EQ.IWHT)GO TO 17
PRINT400
GO TO 15
```

C

```
07 CALL DATA(CARD)
GO TO 02
08 CALL BRANCH(CARD)
GO TO 02
09 CALL MAINB(CARD)
GO TO 02
10 CALL RUN(CARD)
GO TO 02
11 CALL VABC(CARD)
GO TO 02
15 PRINT500
STOP
16 CALL PRINT(CARD)
GO TO 02
17 CALL WEIGHT(CARD)
GO TO 02
100 FORMAT(1HQ,/,10X,*START RUN*,/)
200 FORMAT(20A4)
300 FORMAT(5X,I3,2X,20A4)
400 FORMAT(09X,* ERROR INCORRECT STATEMENT TYPE.*/)
500 FORMAT(///,10X,*END RUN*)
END
```

```

SUBROUTINE VABC (CARD)
C *** THIS SUBROUTINE RUNS THE MODEL WITH VARIABLE BOUNDARY CONDITIONS.
C *** THE VABC CARD MUST BE PRECEDED BY A RUN CARD TO ENSURE THAT ALL
C *** ARRAYS ARE PROPERLY INITIALIZED AND THAT THE MODEL HAS CONVERGED
C *** TO AN ACCEPTABLE STEADY STATE CONDITION.
C *** THIS SUBROUTINE WILL NOT ALLOW THE BOUNDARY CONDITIONS OR THE
C *** EQUATION TYPE TO BE CHANGED FROM THOSE INITIALLY SET UP BY SUB-
C *** ROUTINES DATA,MAINB,AND BRANCH.
C *** POSSIBLE BOUNDARY CONDITIONS ARE
C ***     VELOCITY -SPECIFIED OR CALCULATED
C ***     DISCHARGE-SPECIFIED OR CALCULATED
C ***     HEIGHT ABOVE DATUM- SPECIFIED OR CALCULATED.
C *** THE USER MUST SUPPLY INPUT SUBROUTINES TO STORE THE BOUNDARY
C *** CONDITIONS IN THE PROPER ARRAYS, SEE SUBROUTINES CALBC AND READBC
C ***     CODES FOR BOUNDARY CONDITIONS ARE
C ***     1     HEIGHT IS SPECIFIED BY A FUNCTION
C ***     2     VELOCITY IS SPECIFIED AS A FUNCTION
C ***     3     DISCHARGE IS SPECIFIED AS A FUNCTION
C ***     4     HEIGHT IS READ IN AS DATA
C ***     5     VELOCITY IS READ IN AS DATA
C ***     6     DISCHARGE IS READ IN AS DATA
C
COMMON /20/ IEQU(40),V(20),B(20),H(20),Z(20),DELTX(20),QL(20),DK
$(20),IPR,LINK(40),LIN(20),IDELT,ISTPT,RMS,GRAV,WHT
COMMON /30/ RHGT(100,21),VEL(100,21),DIS(100,21),FR(100,21)
COMMON /50/ ISW,IRUNS
COMMON /70/ XBC(100,5)
DIMENSION ISECT(5), ICODE(5)
INTEGER HEIG
DATA HEIG/4HHEIG/
DATA ISR/4HVABC/
C
C *****
C
C *** DECODE TIME STEP AND STOPPING TIME
C     DECODE (76,40,CARD) IDELT,INCD,ISTOP,INCS
C     IF(INCD.EQ.3HHRS) IDELT = IDELT*3600
C     IF(INCD.EQ.3HMIN) IDELT = IDELT*60
C     IF(INCS.EQ.3HHRS) ISTOP = ISTOP*3600
C     IF(INCS.EQ.3HMIN) ISTOP = ISTOP*60
C *** TIME STEP AND STOP TIME ARE NOW IN SECONDS.
C
C *** DECODE BOUNDARY CONDITION SECTION NUMBERS AND CODE
C
C     DECODE (76,50,CARD) (ISECT(I),ICODE(I),I=1,5)
C *** IF BC AT SECTION I IS TO BE CALCULATED USE CALCBC. IF BC DATA IS
C *** TO BE READ FROM CARDS USE READBC.

```

```

      IRUNS = IRUNS+1
      NRUNS = ISTOP/IDELT
      IF (NRUNS+IRUNS.LE.100) GO TO 10
      PRINT 60
      NRUNS = 100-IRUNS
C
C *** DETERMINE ALL BOUNDARY CONDITIONS AT ONCE
10  DO 20 I=1,5
      IF (ISECT(I).EQ.0) GO TO 20
      CALL CHECK(ISECT(I),ISR)
      IF (ICODE(I).LE.3) CALL CALCBC (ISECT(I),ICODE(I),NRUNS,I)
      IF (ICODE(I).GE.4) CALL READBC (ISECT(I),ICODE(I),NRUNS,I)
20  CONTINUE
C *** ISW=1 DISABLES FIRST PART OF SR RUN.
      ISW = 1
      CARD = 0
C
C *** CALL SUBROUTINE *RUN* ONCE FOR EACH TIME STEP
30  DO 35 I=1,5
      IF(ISECT(I).EQ.0) GO TO 35
      IST= ISECT(I)
      IF(ICODE(I).EQ.1) GO TO 33
      IF(ICODE(I).NE.4) GO TO 31
33  Z(IST) = XBC(IRUNS,I)
      GO TO 35
31  IF(ICODE(I).EQ.2) GO TO 34
      IF(ICODE(I).NE.5) GO TO 32
34  V(IST) = XBC(IRUNS,I)
      GO TO 35
32  V(IST)=XBC(IRUNS,I)/((H(IST)+Z(IST))*B(IST))
35  CONTINUE
      CALL RUN(CARD)
      IRUNS = IRUNS+1
      IF (IRUNS.LE.NRUNS) GO TO 30
      RETURN
C
40  FORMAT (10X,I3,1X,A3,10X,I5,1X,A3)
50  FORMAT (43X,5(I2,1X,I2,1X))
60  FORMAT(10X,*MORE THAN 100 TIME STEPS CALLED FOR*)
      END

```

```

SUBROUTINE RUN(CARD)
C *** THIS SUBROUTINE RUNS A STEADY STATE MODEL
C ***      ENDS AT IPR
C ***      TIME INCREMENT IDELT
C ***      STOP TIME      ISTPT
C ***
      INTEGER CARD(19),ITYP(10),CX(19),POINT,ISSS
      COMMON/10/X(40),N,MAXSA,POINT(40,40),ISUB(39),TEMP(40),
      * PART(40),COE(40,41),ISIGDIG
      COMMON/20/IEQU(40),V(20),B(20),H(20),Z(20),DELT(20),QL(20),
      * DK(20),IPR,LINK(40),LIN(20),IDELT,ISTPT,RMS,GRAV,WHT
      COMMON/30/RHGT(100,21),VEL(100,21),DIS(100,21),FR(100,21)
      COMMON/40/CA(19),CB(19),CC(19),CD(19),CE(19),HTS(19),
      * DV(19),VP(19),DZ(19),ZP(19),AS(19),ZO(19),WHTN,HN(19)
      COMMON/50/ISW,IRUNS
      DATA CX,ISEC,IMIN,IHRS/19*4H      ,3HSEC,3HMIN,3HRS/
      DATA ISR/4HRUN /

C *****
C *****
C *****
C *** ISW IS SET IN S.R. VABC, AFTER THE INITIAL CALL TO RUN BY
C *** THE RUN CARD, ALL SUBSEQUENT CALLS ARE MADE BY S.R. VABC
      IF (ISW.EQ.1) GO TO 04
      DECODE(76,100,CARD) IA,IB,INC,IC,ISP
100  FORMAT(13X,I2,11X,I3,1X,A3,11X,I5,1X,A3)
      IF(IA.NE.0)IPR=IA
      IF(IB.NE.0)IDELT=IB
      IF(IC.NE.0)ISTPT=IC
      CALL CHECK(IPR,ISR)
      N=2*IPR
C *** PRINT OUT THE INITIAL VALUES OF THE ARRAYS IEQU AND LINK
C *** TO ENSURE THE MODEL IS SET UP CORRECTLY
      PRINT 1000,WHT,GRAV,MAXSA,ISIGDIG
1000  FORMAT(1H0,2X,*WHT=*,F4.3,5X,*GRAV =*,F6.3,5X,*MAXSA =*,
      * I3,5X,*ISIGDIG =*,I3)
      PRINT 200,(I,I=1,N)
      PRINT 300,(IEQU(I),I=1,N)
      PRINT 400,(LINK(I),I=1,N)
200  FORMAT (1H0,10X,*MODEL SET UP*,/,* INDEX K*,4X,40(I2,1X))
300  FORMAT (1H ,* IEQU(K)*,4X,40(I2,1X))
400  FORMAT(1H ,* LINK(K)*,4X,40(I2,1X))
C *** INITIALIZE VARIABLES
      K=1
      IRUNS=0
      DO 03 I=1,IPR
      X(K)=V(I)
      X(K+1)=Z(I)

```

```

03 K=K+2
C
C *** CHANGE ALL INPUT TIME REFERENCES TO SECONDS
    IF(INC.EQ.ISEC) GO TO 06
    IF(INC.EQ.IMIN) GO TO 02
    IF(INC.EQ.IHRS) GO TO 01
01 IDELT=60*IDELT
02 IDELT=60*IDELT
06 DELT =FLOAT(IDELT)
    IF(ISP.EQ.ISEC) GO TO 09
    IF(ISP.EQ.IMIN) GO TO 08
    IF(ISP.EQ.IHRS) GO TO 07
07 ISTPT=60*ISTPT
08 ISTPT=60*ISTPT
09 CONTINUE
C
C *** INITIALIZE ARRAYS
    WHTN=1.0-WHT
    ITIME=0.0
14 IRUNS=IRUNS+1
04 ITIME=ITIME+IDELT
    RHGT(IRUNS,1)=ITIME
    VEL(IRUNS,1)=ITIME
    DIS(IRUNS,1)=ITIME
    FR(IRUNS,1)=ITIME
C *** CALCULATE COEFFICIENTS
    NM=IPR-1
    DO 12 I=1,NM
        VP(I)=V(I+1)+V(I)
        ZP(I)=Z(I+1)+Z(I)
        AS(I)=B(I)*H(I)+B(I+1)*H(I+1)+WHTN*(B(I)*Z(I)+B(I+1)*Z(I+1))
        OZ(I)=WHTN*(Z(I+1)-Z(I))
        ZO(I)=WHTN*(V(I+1)+V(I))
        OV(I)=WHTN*(V(I+1)-V(I))
        HTS(I)=H(I)+H(I+1)+WHTN*(Z(I)+Z(I+1))
        HN(I)=H(I+1)-H(I)+WHTN*(Z(I+1)-Z(I))
        CA(I)=DELTX(I)/DELT
        CB(I)= 4.0*DELTX(I)*QL(I)/(B(I)+B(I+1))
        CC(I)=DELT/DELTX(I)
        CD(I)=(B(I+1)-B(I))/(B(I+1)+B(I))
C *** IF DK(I) IS LESS THAN 1.0, MANNINGS FRICTION IS
C *** ASSUMED. IF DK(I) IS GREATER THAN 1.0 THEN CHEZY
C *** FRICTION IS ASSUMED
        IF(DK(I).LT.1.) GO TO 10
        CE(I)=DELT*GRAV/(DK(I)*DK(I))
        GO TO 12
10 CE(I)=(DELT*GRAV*DK(I)*DK(I))/
    $      (((H(I)+Z(I)+H(I+1)+Z(I+1))*0.33333)+1.7621)

```



```

12 CONTINUE
C
C *** SOLVE SYSTEM OF EQUATIONS TO
C ***     FOUR SIGNIFICANT DIGITS OR
C ***     MAXIT MAXIMUM NUMBER OF ITERATIONS
      MAXIT=MAXSA
      CALL NONLIN(MAXIT,ISING,ISIGDIG)
C
C *** CHECK TO SEE IF SOLUTION OBTAINED
      IF(ISING.NE.0) GO TO 13
      PRINT 500,MAXIT,ITIME
500 FORMAT(10X,*SOLUTION MATRIX IS SINGULAR AT ITERATION*,I4,
+ *, TIME IS *,I5,*SECONDS ---- RUN STOPPED*)
      STOP
13 IF(MAXIT.GE.MAXSA) PRINT 600,ITIME
600 FORMAT(10X,*WARNING SOLUTION MAY NOT BE ACCURATE AT TIME *, I5,*
*SECONDS*)
C
      K=-1
      DO 05 I=1,IPR
      K=K+2
      V(I)=X(K)
      Z(I)=X(K+1)
      RHGT(IRUNS,I+1)=Z(I)
      VEL(IRUNS,I+1)=V(I)
C *** CALCULATE DISCHARGE NO FROUDE NUMBER
      DIS(IRUNS,I+1)=V(I)*B(I)*(H(I)+Z(I))
      FR(IRUNS,I+1) = DELTX(I)/((V(I)+V(I+1))*30.)
05 CONTINUE
      PRINT 700,ITIME,MAXIT
      PRINT 800,(V(I),I=1,IPR)
      PRINT 900,(Z(I),I=1,IPR)
700 FORMAT(1H/,/,*, ALL CALCULATIONS COMPLETE FOR TIME = *,I5,* SECONDS
+ --- NO.OF ITERATIONS = *,I2,/)
800 FORMAT (1H0,*VEL*,5X,20F6.2)
900 FORMAT (1H ,*HEIGHT*,2X,20F6.2)
      IF(ITIME.LT.ISTPT) GO TO 14
      RETURN
      END

```

```

      SUBROUTINE WEIGHT(CARD)
C *** THIS SUBROUTINE READS THE WEIGHT CARD
C ***   WHT= WEIGHTING FACTOR IN FINITE DIFFERENCE SCHEME
C ***   GRAV= GRAVITATIONAL CONSTANT
C ***   MAXSA= MAXIMUM NO. OF ITERATIONS BY NONLIN
C ***   NUMSIG= NUMBER OF SIGNIFICANT FIGURES CALCULATED BY NONLIN
C *** DEFAULT VALUES IF NO WEIGHT CARD IS PRESENT OR IF VALUES ON
C *** WEIGHT CARD ARE BLANK ARE
C ***   WHT=0.55
C ***   GRAV= 32.2
C ***   MAXSA= 7
C ***   NUMSIG=5
C *** THESE DEFAULT VALUES ARE SET IN THE MAIN PROGRAM
C
      INTEGER CARD(19),POINT
      COMMON/10/X(40),N,MAXSA,POINT(40,40),ISUB(39),TEMP(40),
* PART(40),COE(40,41),ISIGDIG
      COMMON/20/IEQU(40),V(20),B(20),H(20),Z(20),DELT(20),QL(20),
* DK(20),IPR,LINK(40),LIN(20),IDELT,ISTPT,RMS,GRAV,WHT
C
      DECODE(76,100,CARD) RA,RB,IC,ID
100  FORMAT(4X,F3.2,10X,F4.2,8X,I2,10X,I2)
      IF(ID.GT.0)ISIGDIG=ID
      IF(IC.GT.0) MAXSA=IC
      IF(RB.GT.0.0) GRAV=RB
      IF(RA.GT.0.0) WHT =RA
      IF(RA.GT.0.5) GO TO 01
      PRINT 200
200  FORMAT(1H0,*WARNING CHECK THAT COURANT STABILITY CRITERION IS
* SATISFIED AS SOLUTION MAY BE UNSTABLE*)
01  RETURN
      END

```

```

SUBROUTINE CALCBC (ISECT,ICODE,NRUNS,I)
C *** THIS SUBROUTINE MUST BE USER SUPPLIED.
C *** IT IS SUPPOSED TO CALCULATE THE APPROPRIATE
C *** BOUNDARY CONDITIONS FOR THE MODEL AND PLACE
C *** THEM IN THE APPROPRIATE ARRAYS.
C *** THE I-TH BOUNDARY CONDITION IS TO BE CALCULATED FROM THE TIME STEP
C *** IRUNS+1 TO TIME STEP NRUNS AND PLACED IN THE ARRAY XBC( ,I).
C *** FUNCTION VALUES IN THIS SUBROUTINE MUST BE CHANGED BY THE
C *** USER TO SUIT HIS CASE.....
C *** ISECT = THE SECTION NUMBER AT WHICH THE I-TH BOUNDARY CONDITION
C *** OCCURS
C *** ICODE = THE TYPR OF BOUNDARY CONDITION
C ***      1 = VELOCITY
C ***      2 = HEIGHT
C ***      3 = DISCHARGE
C *** NRUNS = LTOTAL NUMBER OF NUNS TO BE COMPLETED
C *** I = BOUNDARY CONDITION NUMBER..IE.. THE FIRST,SECOND...FIFTH
C *** BOUNDARY CONDITION ON THE VABC CARD
COMMON /70/ XBC(100,5)
GO TO (10,20,30,40,50), I
10 XBC(2,1) = 25.0
   XBC(3,1) = 30.0
   XBC(4,1) = 35.0
   DO 15 JJ=5,NRUNS
15 XBC(JJ,1) = 40.0
   RETURN
20 RETURN
30 RETURN
40 RETURN
50 RETURN
END

```

```

      SUBROUTINE READBC(ISECT,ICODE,NRUNS,I)
C *** THIS SUBROUTINE MUST BE USER SUPPLIED
C *** IT IS SUPPOSED TO READ IN THE APPROPRIATE BOUNDARY
C *** CONDITIONS AND PLACE THEM IN THE PROPER ARRAY.
      COMMON /70/ XBC(100,5)
      DO 10 J=6,65,12
        II=J+11
        READ 100, (XBC(K,I), K=J,II)
100  FORMAT(20X,12(F4.2))
      10 CONTINUE
      DO 15 J=6,65
15  XBC(J,I) = XBC(J,I) - 76.80
      RETURN
      END

```

```

SUBROUTINE DATA(CARD)
C ***
C *** THIS SUBROUTINE READS IN ALL DATA PERTAINING TO
C *** SECTION I AND SETS UP INITIAL EQUATION TYPES.
C *** I= SECTION NUMBER
C *** V(I)=INITIAL GUESS AT VELOCITY
C *** Z(I)=INITIAL GUESS AT STAGE ABOVE DATUM. +VE IF ABOVE DATUM
C *** -VE IF BELOW DATUM
C *** B(I)= TOP WIDTH OF MAIN CHANNEL SECTION
C *** H(I)= DEPTH OF SECTION BOTTOM BELOW DATUM
C *** +VE IF BOTTOM IS BELOW DATUM
C *** -VE IF BOTTOM IS ABOVE DATUM
C *** DELX(I)= DISTANCE BETWEEN SECTION I AND SECTION I+1
C *** QL(I)=LATERAL INFLOW PER UNIT LENGTH BETWEEN
C *** SECTION I AND SECTION I+1.
C *** DK(I)=FRICTION FACTOR BETWEEN SECTION I AND SECTION I+1.
C *** IF DK(I).LT.1 ,MANNINGS FRICTION IS ASSUMED.
C *** IF DK(I).GT.1, CHEZY FRICTION IS ASSUMED.
C ***
      INTEGER CARD(19)
      COMMON/20/IEQU(40),V(20),B(20),H(20),Z(20),DELTX(20),QL(20),
      * DK(20),IPR,LINK(40),LIN(20),IDELT,ISTPT,RMS,GRAV,WHT
      DATA ISR/4HDATA/
C ***
C *** DECODE SECTION NUMBER
      DECCDE (4,100,CARD) I
      100 FORMAT(1X,I2,1X)
      CALL CHECK(I,ISR)
C**** DECODE SECTION PROPERTIES
      DECODE(76,200,CARD) V(I),Z(I),B(I),H(I),DELTX(I),QL(I),DK(I)
      200 FORMAT(4X,F4.2,1X,F5.2,1X,F5.1,1X,F5.2,1X,F7.1,1X,F5.3,1X,F4.0)
C *** SET UP INITIAL EQUATION TYPES IN IEQU ARRAY AND REACH NUMBERS
C *** IN LINK ARRAY.THESE NUMBERS ARE MODIFIED IN MAINB,BRANCH,AND
C *** VARBC SUBROUTINES.
      IN=2*I
      IV=2*I-1
      IEQU(IV)=1
      LINK(IV)=I
      IEQU(IN)=2
      LINK(IN)=I
      RETURN
      END

```

```

SUBROUTINE MAINB(CARD)
C *** THIS SUBROUTINE PROCESSES MAIN CARD
C *** IT SETS UP BOUNDARY CONDITIONS FOR THE MAIN BRANCH
C *** THE POSSIBLE BOUNDARY CONDITIONS ARE
C ***     FIXED HEIGHT -- EQUATION TYPE 3
C ***     FIXED VELOCITY-- EQUATION TYPE 4
C ***     FIXED DISCHARGE -- EQUATION TYPE 5
C *** BOUNDARY CONDITIONS MAY OCCUR AT ANY SECTION IN THE MAIN
C *** BRANCH OF THE RIVER SYSTEM.
C *** WHICH GOVERN A BRANCH.
C *** NOTE THAT THE CLOSER THE TWO BOUNDARY CONDITIONS ARE TO EACH
C *** OTHER THE HIGHER IS THE PROBABLE ERROR IN THE SYSTEM, AND THE
C *** HIGHER IS THE SENSITIVITY OF THE SYSTEM TO THE BOUNDARY CONDITIONS
C ***
      INTEGER CARD(19)
      COMMON/20/IEQU(40),V(20),B(20),H(20),Z(20),DELTX(20),QL(20),
      *   DK(20),IPR,LINK(40),LIN(20),IDELT,ISTPT,RMS,GRAV,WHT
      DATA INF,IOUTF,IHEIG,IVEL/4HINFL,4HOUTF,4HHEIG,4HVELO/
      DATA IDIS/3HDIS/
      DATA ISR/4HMAIN/
C *** DECODE MAIN CARD. MAIN BRANCH FROM SECTION 1 TO SECTION I,
      DECODE(52,100,CARD) I,IU,ICA,ID,ICB
      100 FORMAT(9X,I2,8X,I2,7X,A4,7X,I2,7X,A4)
      CALL CHECK(I,ISR)
C *** DECIDE ON EQUATION TYPES FOR BOTH BOUNDARY CONDITIONS
      IF(ICA.EQ.IHEIG) IE=3
      IF(ICA.EQ.IVEL) IE=4
      IF(ICA.EQ.IDIS) IE=5
      IF(ICB.EQ.IHEIG) IS=3
      IF(ICB.EQ.IVEL) IS=4
      IF(ICB.EQ.IDIS) IS=5
C *** UPSTREAM BOUNDARY CONDITION AT SECTION IU IS TYPE ICA.
C *** DOWNSTREAM BOUNDARY CONDITION AT SECTION ID IS TYPE ICB.
C *** UPSTREAM BC EQUATION TYPE PLACED IN IEQU(Z*I-1)
C *** DOWNSTREAM BC EQUATION TYPE PLACED IN IEQU(Z*I)
      IEQU(2*I-1)=IE
      LINK(2*I-1)=IU
      IEQU(2*I)=IS
      LINK(2*I)=ID
      RETURN
      END

```

```

SUBROUTINE BRANCH(CARD)
C *** THIS SUBROUTINE PROCESSES BRANCH CARD
C *** IT SETS UP THE BOUNDARY CONDITIONS WITHIN THE BRANCHES AND
C *** LINKS THE BRANCHES TO THE REST OF THE SYSTEM
C *** EQUATION TYPES SET IN THIS SUBROUTINE OVERRIDE THOSE
C *** SET UP IN SUBROUTINE *DATA*
C *** NOTE THAT SECTIONS INVOLVED IN A BRANCHING MUST NOT BE
C *** USED AS BOUNDARY CONDITIONS.
C *** JUNCTION EQUATIONS ARE AS FOLLOWS
C *** FIXED WATER LEVEL BETWEEN SECTIONS I,J = EQUATION TYPE 6
C *** INFLOW JUNCTION CONTINUITY EQUATION = EQUATION TYPE 7
C *** OUTFLOW JUNCTION CONTINUITY EQUATION = EQUATION TYPE 8
      INTEGER CARD(19),OUT
      COMMON/20/IEQU(40),V(20),R(20),H(20),Z(20),DELTX(20),QL(20),
      *   DK(20),IPR,LINK(40),LIN(20),IDELT,ISTPT,RMS,GRAV,WHT
      DATA IFIX,IFR,IVEL,IHEI/4HFIXE,4HFROM,4HVELO,4HHEIG/,ITO/4HTO /
      DATA IDIS/3HDIS/
      DATA ISR/4HBRAN/
C ***
      DECODE(76,100,CARD) IS,IE,ICA,ITA,ICB,ITB
      CALL CHECK(IE,ISR)
      CALL CHECK(IS,ISR)
      IEN=2*IE-1
C *** IF INFLOW TO BRANCH IS FIXED VEL OR HEIGHT OR DISCHARGE
C *** USE THIS SECTION TO SET EQUATION TYPE
      IF(ICA.NE.IFIX) GO TO 01
      IF(ITA.EQ.IVEL) IEQU(IEN)=4
      IF(ITA.EQ.IHEI) IEQU(IEN)=3
      IF(ITA.EQ.IDIS) IEQU(IEN)=5
      LINK(IEN)=IS
      GO TO 02
01 IF(ICA.NE.IFR) GO TO 05
      DECODE(76,200,CARD) INL
C *** THIS SECTION SETS EQUATION TYPES IF INFLOW IS FROM INL.
      CALL CHECK(INL,ISR)
      IEQU(2*INL-1)= 6
      IEQU(2*INL)=6
      LINK(2*INL-1)=INL+1
      LINK(2*INL)=IS
      IEQU(2*IS-1)=7
      LINK(2*IS-1)=INL
      LINK(2*IE-1) = IS
C ***
C *** IF OUTFLOW FROM BRANCH IS FIXED,SET EQUATION TYPES HERE
02 IF(ICB.NE.IFIX) GO TO 03
      IF(ITB.EQ.IHEI) IEQU(2*IE)=3
      IF(ITB.EQ.IVEL) IEQU(2*IE)=4

```

```

      IF(ITB.EQ.IDIS) IEQU(2*IE)=5
      LINK(2*IE)=IE
      GO TO 04
C *** THIS SECTION SETS EQUATION TYPES IF OUTFLOW IS TO OUT
      03 IF(ICB.NE.ITO) GO TO 05
      DECODE(76,300,CARD) OUT
      CALL CHECK(OUT,ISR)
      IEQU(2*OUT-3)=6
      LINK(2*OUT-3)=OUT
      IEQU(2*OUT-2)=6
      LINK(2*OUT-2)=IE
      IEQU(2*IE)=8
      LINK(2*IE)=OUT-1
      LINK(2*IE-1)=IS
      04 RETURN
C ** ERROR IN BOUNDARY CONDITIONS
      05 PRINT 400
      STOP
      100 FORMAT(8X,I2,4X,I2,8X,A4,2X,A4,13X,A4,2X,A4)
      200 FORMAT(29X,I2)
      300 FORMAT(52X,I2)
      400 FORMAT(10X,*INCORRECT KEY WORD IN BOUNDARY CONDITIONS*,///,
        * 10X,*STOP RUN*)
      END

```



```

SUBROUTINE PRINT(CARD)
C ***
C *** THIS SUBROUTINE PRINTS TABLES OF VALUES
C *** CALCULATED BY THE MODEL.
C *** THESE TABLES ARE:
C *** 1. WATER LEVELS
C *** 2. VELOCITY
C *** DISCHARGE
C *** TRANSIT TIME THROUGH THE REACH.
C *** THE TIME UNITS IN THE TABLE MAY BE IN SEC,MIN,OR HRS.
C *** THE DATUM OF THE WATER LEVEL MAY BE ADJUSTED IF NECESSARY.
C ***
      INTEGER CARD(19)
      COMMON/30/RHGT(100,21),VEL(100,21),DIS(100,21),FR(100,21)
      COMMON/50/ISW,IRUNS
      DATA ISR/4HPRIN/
C
      DECODE (76,110,CARD) IPT,INC,DATUM
      CALL CHECK(IPT,ISR)
      ITF=IPT+1
      IF(INC.EQ.3HSEC) GO TO 01
      IF(INC.EQ.3HMIN) FAC=60.
      IF (INC.EQ.3HHRS) FAC = 3600.
      DO 01 I=1,IRUNS
        RHGT(I,1) = RHGT(I,1)/FAC
        VEL(I,1) = VEL(I,1)/FAC
        DIS(I,1) = DIS(I,1)/FAC
        FR(I,1) = FR(I,1)/FAC
      01 CONTINUE
      PRINT 1100
      PRINT 1200,(I,I=1,IPT)
      DO 04 I=1,IRUNS
        DO 03 J=2,ITF
          03 RHGT(I,J) = RHGT(I,J) + DATUM
          PRINT 1300,(RHGT(I,J),J=1,ITF)
        04 CONTINUE
      PRINT 1400
      PRINT 1200,(I,I=1,IPT)
      DO 05 I=1,IRUNS
        PRINT 1300,(VEL(I,J),J=1,ITF)
      05 CONTINUE
      PRINT 1500
      IF(ITF.GT.10) GO TO 07
      PRINT 1700,(I,I=1,IPT)
      DO 06 I=1,IRUNS
        PRINT 1800,(DIS(I,J),J=1,ITF)
      06 CONTINUE

```

```

      GO TO 09
07 PRINT1700,(I,I=1,10)
   PRINT 1750,(I,I=11,IPT)
      DO 08 I=1,IRUNS
   PRINT1800,(DIS(I,J),J=1,11)
   PRINT1850,(DIS(I,J),J=12,ITF)
08 CONTINUE
09 PRINT1600
   PRINT 1900,(I,I=1,IPT)
      DO 10 I=1,IRUNS
   PRINT 2000,(FR(I,J),J=1,IPT)
10 CONTINUE
   RETURN
110 FORMAT(14X,I2,9X,A3,10X,F5.2)
1100 FORMAT(1H1,45X,*TABLE CONTAINING HEIGHTS VS TIME*,/)
1200 FORMAT(1H ,*TIME*,5X,20(I2,4X),/)
1300 FORMAT(1H ,F7.0,2X,20(F5.2,1X))
1400 FORMAT(1H1,45X,*TABLE CONTAINING VELOCITY VS TIME*,/)
1500 FORMAT(1H1,45X,*TABLE CONTAINING DISCHARGE VS TIME*,/)
1600 FORMAT(1H1,45X,*TABLE CONTAINING TRANSIT TIME IN REACH*,/)
1700 FORMAT(1H ,*TIME*,5X,10(5X,I2,5X),/)
1750 FORMAT(1H ,8X,10(10X,I2))
1800 FORMAT(1H ,F7.0,2X,10(F10.1,2X))
1850 FORMAT(1H ,11X,10(2X,F10.1))
1900 FORMAT(1H ,*TIME*,20(4X,I2))
2000 FORMAT(1H ,F7.0,1X,19F5.1)
      END
      FUNCTION FUN(K)
C ***
C *** THIS SUBROUTINE IS CALLED BY NON-LIN AND TABULATES THE
C      EQUATIONS OF MOTION OF THE SYSTEM
C *** NON-LIN DETERMINES VALUES OF THE VARIABLES SUCH THAT
C ***      FUN=0.000 IN EACH CASE
      INTEGER POINT
      COMMON/10/X(40),N,MAXSA,POINT(40,40),ISUB(39),TEMP(40),
*      PART(40),COE(40,41),ISIGDIG
      COMMON/20/IEQU(40),V(20),B(20),H(20),Z(20),DELTX(20),QL(20),
*      DK(20),IPR,LINK(40),LIN(20),IDELT,ISTPT,RMS,GRAV,WHT
      COMMON/40/CA(19),CB(19),CC(19),CD(19),CE(19),HTS(19),
*      DV(19),VP(19),DZ(19),ZP(19),AS(19),ZO(19),WHTN,HN(19)
C ***
C *** FIND EQUATION TYPE
C ***
      IR=IEQU(K)
      IF(IR.LT.1.OR.IR.GT.9) GO TO 10
C
      GO TO(01,02,03,04,05,06,07,08,09), IR
C *** CONTINUITY EQUATION

```

```

01 IN=LINK(K)
   J=2*IN-1
   CW=ZO(IN)+WHT*(X(J)+X(J+2))
   FUN=CA(IN)*(X(J+1)+X(J+3)-ZP(IN))+(HTS(IN)+WHT*(X(J+1)+X(J+3))) *
   * (UV(IN)+WHT*(X(J+2)-X(J)))+CW*(HN(IN)+WHT*(X(J+3)-X(J+1)))
   * + (HTS(IN)+WHT*(X(J+3)+X(J+1)))*CD(IN)+CW-CB(IN)
   RETURN
C *** MOMENTUM EQUATION
02 IN=LINK(K)
   J=2*IN
   CU=DZ(IN)+WHT*(X(J+2)-X(J))
   CU=2.*CU
   CV=ZO(IN)+WHT*(X(J-1)+X(J+1))
   CX=1./(HTS(IN)+WHT*(X(J)+X(J+2)))
   CW=HN(IN)+WHT*(X(J+2)-X(J))
   CZ=CV(IN)+WHT*(X(J+1)-X(J-1))
   CS=X(J-1)+X(J+1)-VP(IN)
   FUN=CC(IN)*GRAV*CU + 1.0*CS + CV*(CX*(1.0*(X(J)+X(J+2)-ZP(IN))
   * +CE(IN)*ABS(CV) + CC(IN)*CV*CW) + CC(IN)*(CV*CD(IN) + 2.*CZ))
   RETURN
C *** BOUNDARY CONDITION -FIXED VELOCITY
04 IN=LINK(K)
   IV=2*IN-1
   FUN=X(IV)-V(IN)
   RETURN
C *** BOUNDARY CONDITION -FIXED HEIGHT
03 IN=LINK(K)
   IV=2*IN
   FUN=X(IV)-Z(IN)
   RETURN
C *** BOUNDARY CONDITION -FIXED DISCHARGE
05 IN=LINK(K)
   IV=2*IN
   FUN=X(IV-1)*B(IN)*(H(IN)+X(IV))-V(IN)*B(IN)*(H(IN)+Z(IN))
   RETURN
C *** JUNCTION CONDITION -FIXED HEIGHT
06 IJUN= 2*LINK(K)
   IV=(K+1)/2
   IVI= 2*(IV)
   FUN= X(IVI)-X(IJUN)
   RETURN
C *** CONTINUITY AT DIVERGENT JUNCTION
C OBTAIN SECTION NUMBERS
07 IN=LINK(K)
   IBN=K/2+1
   IMN=IN+1
C CONTINUITY EQUATION
FUN= X(2*IN-1)*B(IN)*(H(IN)+X(2*IN))-X(2*IBN-1)*B(IBN)*(H(IBN)

```

```

      * +X(2*IBN))-X(2*IMN-1)*B(IMN)*(H(IMN)+X(2*IMN))
      RETURN
C *** CONTINUITY AT CONVERGENT JUNCTION
C      OBTAIN SECTION NUMBERS
      08 IN= LINK(K)+1
         IBN=K/2
         IMN=LINK(K)
C      CONTINUITY EQUATION
      FUN=X(2*IN-1)*B(IN)*(H(IN)+X(2*IN))-X(2*IBN-1)*B(IBN)*(H(IBN)
      * +X(2*IBN)) - X(2*IMN-1)*B(IMN)*(H(IMN)+X(2*IMN))
      RETURN
      09 CONTINUE
      RETURN
      10 PRINT 100
      100 FORMAT(1H0,*UNRECOGNIZABLE EQUATION TYPE ---RUN STOPPED*)
      STOP
      END

```

```

      SUBROUTINE CHECK(I,ISR)
C ***
C *** SUBROUTINE CHECKS TO SEE IF I IS BETWEEN 1-21
C ***
      IF(I.LE.21.AND.I.GE.1) RETURN
      PRINT 100,I,ISR
100  FORMAT(1H0, 'DATA POINT NUMBER =',I3,' IN SUBROUTINE ',A4,
+ * ----- STOP RUN*)
      STOP
      END

```

```

SUBROUTINE NONLIN(MAXIT,SINGLE,NUMSIG)
C ***
C ***      THIS SUBROUTINE SOLVES NON-LINEAR EQUATIONS.
C ***      CONVERGENCE IS ROUGHLY QUADRATIC
C ***
C ***      SOURCE      ALGORITHM 316  COMMUNICATIONS OF ACM
C ***                  VOLUME10,NUMBER11,NOVEMBER,1967.....INCLUDING THE
C ***                  SUGGESTIONS IN CCM.OF A.C.M. VOL.14,NO.7,JULY,U
C ***                  1971, P.493
C ***
C ***      IMPLEMENTED IN FORTRAN BY RYAN RAZ, FEBRUARY 1974
C ***
C ***      COMPUTER  CDC 3100 MASTER O.S.
C ***
C ***      ARGUEMENTS AND DIMENSIONS
C ***
C ***      N      NUMBER OF EQUATIONS
C ***      MAXIT  MAXIMUM NUMBER OF ITERATIONS TO BE MADE
C ***      NUMBER ACTUALLY MADE RETURNED IN MAXIT
C ***      NUMSIG  NUMBER OF SIGNIFICANT DIGITS DESIRED
C ***      X(N)   INITIAL GUESS TO SOLUTION, AFTER EXECUTION
C ***      IT IS THE SOLUTION
C ***      FUN    NAME OF FUNCTION  FUN(X,K)= K TH FUNCTION
C ***      EVALUATED AT X1,X2,...,XN
C ***      SINGLE VALUE RETURNED BY NONLIN
C ***      SINGLE= 0  JACOBIAN MATRIX  SINGULAR
C ***      = 1  SOLUTION OBTAINED
C ***      POINT(N,N),ISUB(N-1),TEMP(N),PART(N),COE(N,N+1)
C ***      WORKING ARRAYS
C ***
C ***      SUBROUTINE NEEDED  BACK
C ***
      INTEGER POINT,SINGLE,CONVRG,TALLY
      COMMON/10/X(40),N,MAXSA,POINT(40,40),ISUB(39),TEMP(40),
*      PART(40),COE(40,41),ISIGDIG
      LOGICAL SWT
C ***
      CONVRG=1
      SINGLE=1
      RELCON=10.**(-NUMSIG)
      DO 140 M=1,MAXIT
      DO 005 J=1,N
005 POINT(1,J)=J
      SWT=.TRUE.
      DO 100 K=1,N
      IF(SWT)GO TO 010
      CALL BACK(K)

```

```

010 F=FUN(K)
    FACTOR=.001
020 TALLY=0
    DO 040 I=K,N
        ITEMP=POINT(K,I)
        HOLD=X(ITEMP)
        H=FACTOR*HOLD
        IF(H.EQ..0)H=FACTOR
        X(ITEMP)=HOLD+H
        IF(SWT)GO TO 030
        CALL BACK(K)
030 FPLUS=FUN(K)
    PART(ITEMP)=(FPLUS-F)/H
    X(ITEMP)=HOLD
    IF(ABS(F/AMAX1(01E-100,ABS(PART(ITEMP))))).GT.01E20)TALLY=TALLY+1
040 CONTINUE
    IF(TALLY.LE.N-K)GO TO 050
    FACTOR=FACTOR*10.0
    IF(FACTOR.GT..5)GO TO 150
    GO TO 020
050 IF(K.LT.N)GO TO 060
    IF(ABS(PART(ITEMP)).LE.01E-100)GO TO 150
    COE(K,N+1)=0
    KMAX=ITEMP
    GO TO 95
060 KMAX=POINT(K,K)
    DERM=ABS(PART(KMAX))
    KPLUS=K+1
    DO 080 I=KPLUS,N
        JSUB=POINT(K,I)
        TEST=ABS(PART(JSUB))
        IF(TEST.LT.DERM)GO TO 070
        DERM=TEST
        POINT(KPLUS,I)=KMAX
        KMAX=JSUB
    GO TO 080
070 POINT(KPLUS,I)=JSUB
080 CONTINUE
    IF(ABS(PART(KMAX)).LT.01E-100)GO TO 150
    ISUB(K)=KMAX
    COE(K,N+1)=0
    PARTS=PART(KMAX)
    DO 090 J=KPLUS,N
        JSUB=POINT(KPLUS,J)
        COE(K,JSUB)=-PART(JSUB)/PARTS
090 COE(K,N+1)=COE(K,N+1)+PART(JSUB)*X(JSUB)
    95 COE(K,N+1)=(COE(K,N+1)-F)/PART(KMAX)+X(KMAX)
100 SWT=.FALSE.

```

```

X(KMAX)=COE(N,N+1)
IF(N.GT.1)CALL BACK(N)
IF(M.EQ.1)GO TO 130
DO 110 I=1,N
IF(ABS((TEMP(I)-X(I))/AMAX1(ABS(X(I)),1E-100)).GT.RELCON)GO TO 12
110 CONTINUE
CONVRG=CONVRG+1
IF(CONVRG.GE.3)GO TO 160
GO TO 130
120 CONVRG=1
130 CONTINUE
DO 140 I=1,N
140 TEMP(I)=X(I)
RETURN
150 SINGLE=0
160 MAXIT=M
RETURN
END

```



```

SUBROUTINE BACK(K)
C *** SUBPROGRAM NEEDED BY NONLIN
C *** BACK SOLVES A TRIANGULAR LINEAR SYSTEM
C ***
C ***
      INTEGER POINT
      COMMON/10/X(40),N,MAXSA,POINT(40,40),ISUB(39),TEMP(40),
      * PART(40),COE(40,41),ISIGDIG
      KM=K
01 KMAX=ISUB(KM-1)
      X(KMAX)=0
      DO 02 J=KM,N
      JSUB=POINT(KM,J)
02 X(KMAX)=X(KMAX)+COE(KM-1,JSUB)*X(JSUB)
      X(KMAX)=X(KMAX)+COE(KM-1,N+1)
      KM=KM-1
      IF(KM.GT.1)GO TO 01
      RETURN
      END
      FINIS

```



Environment
Canada

F&W.S.

O. & A.S.

MANUSCRIPT REPORT SERIES No. 1

UNSTEADY FLOW IN NETWORKS OF OPEN CHANNELS

L. R. MUIR



GB
651
M361
No. 1

OCEAN AND AQUATIC SCIENCES
CENTRAL REGION
CANADA CENTRE FOR INLAND WATERS
BURLINGTON, ONTARIO