ODIRMO
A One Dimensional model for
River Morphology

K. Vermeer

R/1985/H/9

Technische Hogeschool Delft
Afdeling der Civiele Techniek
Vakgroep Waterbouwkunde
Sectie Vloeistofmechanica
$x_0 = 3, a, \ldots = f(x, t)$

$0_p t_0 \rightarrow$ via geleidend: $a(x, 0), b(x, 0), n(x, 0)$, richting

$z(x, 0)$

alleen samendruk.

Nummering bouwen = handtekenen (beheren gemenseerd)

maximaal 6.

water en dichtstrm

beek water en = stroomnummer (wateringijn)

6 stels; transp. Upst.

(aldoossen bouwen

voor sedimen)

mag o.m. 6 by dijkplm en noemen vragen gas te def door

$m_1, m_2, m_3, m_4, m_5, m_6 \quad m_i = \text{alde}

\text{a}$

s => randaarsen $= f_1(t)$

9 => randaarsen $= f_2(t)$

Begin nummerade: alleen $z(x, 0)$

Write data to $h = \text{alde}$ resultaat is een doorsileen!
2. Software remarks

2.1 Contents of software package

The ODIRMO package consist of this report and two 5.25 inch diskettes, containing the program. One diskette is named:

ODIRMO version 1.01, source files

and the other:

ODIRMO version 1.01, code files

It is very important to make a backup copy of these two diskettes for your own use. Safeguard the original diskettes with care and don’t let them fall in illegal copying hands!

The two disks contain the following files:

ODIRMO source files:

<table>
<thead>
<tr>
<th>filename</th>
<th>contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMAND.COM</td>
<td>MS-DOS operating system</td>
</tr>
<tr>
<td>CONFIG.SYS</td>
<td>system information</td>
</tr>
<tr>
<td>DISKCOPY.COM</td>
<td>diskcopy facility</td>
</tr>
<tr>
<td>LISTT.COM</td>
<td>program to list TURBO-Pascal programs</td>
</tr>
<tr>
<td>TURBO.COM</td>
<td>the TURBO-Pascal compiler</td>
</tr>
<tr>
<td>TURBO.MSG</td>
<td>TURBO-Pascal error messages</td>
</tr>
<tr>
<td>ODIRMO.PAS</td>
<td>ODIRMO main program</td>
</tr>
<tr>
<td>INITODIR.PAS</td>
<td>routine InitOdirmo</td>
</tr>
<tr>
<td>READDATA.PAS</td>
<td>routine ReadDataFromFile</td>
</tr>
<tr>
<td>DISCHARG.PAS</td>
<td>routine DischargeCalc</td>
</tr>
<tr>
<td>WATERLEV.PAS</td>
<td>routine WaterlevelDownstream</td>
</tr>
<tr>
<td>BACKWATE.PAS</td>
<td>routine BackwaterCalc</td>
</tr>
<tr>
<td>ROUGHCAL.PAS</td>
<td>function Roughterm</td>
</tr>
<tr>
<td>SEDITRAN.PAS</td>
<td>routine SeditransCalc</td>
</tr>
<tr>
<td>SEDIPST.PAS</td>
<td>routine SeditransUpstream</td>
</tr>
<tr>
<td>RESULTS.PAS</td>
<td>routine Results</td>
</tr>
<tr>
<td>BEDLEVEL.PAS</td>
<td>routine BedlevelCalc</td>
</tr>
<tr>
<td>WRIEDAT.PAS</td>
<td>routine WriteDataToFile</td>
</tr>
<tr>
<td>CREADATA.PAS</td>
<td>CREADATA main program</td>
</tr>
<tr>
<td>INITCREA.PAS</td>
<td>routine InitCrea</td>
</tr>
<tr>
<td>ASKWIDTH.PAS</td>
<td>routine AskWidth</td>
</tr>
<tr>
<td>ASKBEDLE.PAS</td>
<td>routine AskBedLevel</td>
</tr>
<tr>
<td>ASKDISCH.PAS</td>
<td>routine AskDischarge</td>
</tr>
<tr>
<td>ASKTRANS.PAS</td>
<td>routine AskTransport</td>
</tr>
<tr>
<td>ASKWEATER.PAS</td>
<td>routine AskWaterlevel</td>
</tr>
<tr>
<td>ASKRough.PAS</td>
<td>routine AskRoughness</td>
</tr>
<tr>
<td>ASKGRAIN.PAS</td>
<td>routine AskGrainsize</td>
</tr>
<tr>
<td>ASKFORMU.PAS</td>
<td>routine AskFormula</td>
</tr>
<tr>
<td>ASKSOURC.PAS</td>
<td>routine AskSource</td>
</tr>
<tr>
<td>WHATNEXT.PAS</td>
<td>routine WhatNext</td>
</tr>
<tr>
<td>INPUTCHE.PAS</td>
<td>routine to check input</td>
</tr>
<tr>
<td>ENTERREA.PAS</td>
<td>routine to enter real numbers</td>
</tr>
</tbody>
</table>

- 14 -
ODIRMO

A One Dimensional model
for River Morphology

K. Vermeer
ODIRMO, A One Dimensional Morphological Model For Rivers

ERRATA

part II, page 4, line 3:
'morfologisch zijn' should be 'morfologisch model zijn'

part III, page 7, equation (3.2.8) should be:
\[
h^{n-1} = h^n - \frac{u^n}{2} \cdot \left((w^n - w^{n-1}) + 2(a^n - a^{n-1})\right) + \frac{\Delta x, F^n}{g, a} - \frac{u^{n-1} \cdot 2}{g, w^n} \cdot \left((w^{n-1} - w^{n-2}) + \Delta x, F^{n-1}\right)
\]

part III, page 9, equation (3.3.8) should be:
\[
h^{e-1} = h^e - \frac{1}{2} \cdot \left((r^{e-1}, Q)^2 \cdot (w^1, e - w^1, e-1) + 2(a^1, e - a^1, e-1)\right) + \frac{\Delta x, F^1, e}{g, w^1, e} - \frac{(r^{e-1}, Q)^2}{2} \cdot \left((w^1, e-1 - w^1, e-2) + \Delta x, F^1, e-1\right)
\]

The mistakes in eq. (3.2.8) and (3.3.8) are already corrected in ODIRMO.

Test Tg is not valid anymore!
Alt + F1

<table>
<thead>
<tr>
<th>Nummer</th>
<th>Functie</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lijst synoniemen</td>
</tr>
<tr>
<td>2</td>
<td>Zoeken / vervangen eind: F1</td>
</tr>
<tr>
<td>3</td>
<td>Weergeven functies</td>
</tr>
<tr>
<td>4</td>
<td>Blok aan / uit</td>
</tr>
<tr>
<td>5</td>
<td>Aut. nummering / marker tekst</td>
</tr>
<tr>
<td>6</td>
<td>Blok rechts aansluiten</td>
</tr>
<tr>
<td>7</td>
<td>Veleken en kolommen</td>
</tr>
<tr>
<td>8</td>
<td>Pagina opmaak / Blok op 1 pagina</td>
</tr>
<tr>
<td>9</td>
<td>Samenvoegen functies</td>
</tr>
<tr>
<td>10</td>
<td>Macro uitvoeren</td>
</tr>
</tbody>
</table>

Shift + F1

<table>
<thead>
<tr>
<th>Nummer</th>
<th>Functie</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Subscript / superscript /</td>
</tr>
<tr>
<td>3</td>
<td>Achterw. zoeken</td>
</tr>
<tr>
<td>4</td>
<td>Wissel doc. 1 ↔ doc. 2</td>
</tr>
<tr>
<td>5</td>
<td>Beide ruden instappen</td>
</tr>
<tr>
<td>6</td>
<td>Datum / tyd invoegen</td>
</tr>
<tr>
<td>7</td>
<td>(Blok) Centreren</td>
</tr>
<tr>
<td>8</td>
<td>Aantal - kopieer - print</td>
</tr>
<tr>
<td>9</td>
<td>Afbrekesteken aan / uit</td>
</tr>
<tr>
<td>10</td>
<td>Samenvoegen eind record</td>
</tr>
<tr>
<td></td>
<td>Bestand opvragen / tussenvoegen</td>
</tr>
</tbody>
</table>
Word Perfect

Functietoetsen
1 herstellen /ophoeren
2 voorwaarts zoeken
3 Help
4 inspringen tot <Return>
5 Directory
6 Veb zetten aan/uit
7 End Save
8 onderstrepen aan/uit
9 Einde veld
10 Save

Ctrl + F1 Shell progr.
2 Spelling controle
3 schermopbouw
4 blok copieren/verplaatsen
5 DOS-file in/uit
6 Tab
7 voetnoten
8 Print opmaak
9 Mail /sort
t0 Macro definieren

Ctrl - Home/End begin v. regel.
Ctrl - Return nieuwe pagina
Nico,

Dit is mijn enige exemplaar, dus liever niet te lang uitbitten, want dat is vol
Rob Nieuwhamer aan het leiden. Laten met het inbouwen van een modul
g.

Simon.

Floppy's met progr. beschikbaar.
Acknowledgements

This project was carried out at the section Fluid Mechanics between August 1984 and November 1985. During this period I have had the excellent assistance from Jan Ribberink, who was always available when I encountered a minor or major problem, Simon de Boer, who was incredibly patient explaining to me the 'big computer' of the University and tracing more or less obscure failures in my computer program and Nico Booy, who had several useful remarks on the subject of software development. I also wish to thank Prof. de Vries, who initiated this project, and who was astonishingly enthusiastic over my modest results and Prof. Vreugdenhil for introducing me to the numerical calculations of hydraulic problems.

I am also indebted to Mr. Struiksma and Mr. Dijkstra of the Delft Hydraulics Laboratory for their help with the RIVMOR calculations.

This thesis is typed with the Wordwise-plus word processor on a BBC microcomputer and printed with a Star Gemini matrix printer. Thanks to Arnold Knotthnerus for the use of his printer.

Kees Vermeer
November 1985
Contents

Part I : Summary
Part II : Definition Study
Part III : Functional and Technical design
Part IV : Program Description
Part V : Test Cases
Part VI : User Guide
## Part I: Summary

### Contents

1. Introduction ........................................... 2
2. Equations ............................................... 4
3. Proceedings in project ................................. 6
   3.1 General remarks .................................... 6
   3.2 Definition study .................................... 7
   3.3 Functional and technical design .................... 7
   3.4 Computer programs ................................. 9
      3.4.1 CREADATA ..................................... 9
      3.4.2 ODIRMO ....................................... 10
   3.5 Test cases ......................................... 12
4. Evaluation ............................................. 12
   4.1 Project ........................................... 12
   4.2 System Development Methodology (SDM) .............. 14
5. References ............................................ 14
1. Introduction

Generally in rivers not only water but also sediment is transported. This transport of sediment consists of transport over the riverbed, suspended transport and washload. Washload is transport of very small sediment particles. These particles are usually picked up in the region of origin of the river and transported all the way to the sea. As these particles generally do not influence the river morphology, they are neglected in this report. It is obvious, that the rate of sediment transport (bed-load and suspended transport) depends on several parameters (e.g. flow velocity, grain diameter, relative grain density, local bedform). In this report it is assumed, that the total sediment transport depends on the local conditions only. This means, that the transport will be calculated with a transport formula which depends on the local conditions (Engelund & Hansen, Meyer-Peter & Mueller, Ackers & White, power law). It is possible, that the transport also depends on conditions upstream, and that the concentration of suspended sediment is not in equilibrium with respect to the local conditions. In this case, the transport formulae used may not give reliable results and it may be necessary to introduce an other transport calculation (e.g. Galappatti (1983)).

Considering a river section, the bed level of this section is only constant when the transport of sediment at the upstream end equals the transport at the downstream end. The river section is in equilibrium. However, when the transport at the upstream end exceeds the transport downstream, an aggradation will occur in the considered section. In the opposite case a degradation will occur. These morphological changes can be caused by e.g. a water withdrawal (or supply), sediment withdrawal (or supply) or a river bend cut-off.

The aim of this project was to develop a one dimensional computer program which calculates those aggradations and degradations as a function of time for a given situation. Those calculations can be important for economic reasons. For instance an indication can be given how long it will take before dredging at a certain place will become necessary.

For the development of this model a standard method called SDM (System Development Methodology) was used.

This report consists of five major parts. In Part I a global description of the project is given. In Part II the results of the definition study are presented and in Part III the results of the functional and technical design. Part IV gives a description of the two developed computer programs (ODIRMO and CREADATA). In Part V the test cases for the model are discussed and the results of the tests are given. Part VI is a user guide for the model.
Part I: Summary

In chapter 2 of this part the reader is introduced to the basic equations of river morphology. In chapter 3 the proceedings in the project are treated; a short description is given of the System Development Methodology, the Definition Study and the Functional and Technical Design. Chapter 3 is concluded with some remarks on both computer programs ODIRMO (One Dimensional River Morphology) for the morphological computations and CREADATA for the preparation of input data and on the test cases used. In chapter 4 some conclusions and recommendations are given and the usefulness of the SDM method for technical problems is discussed. In chapter 5 some references are given.
2. Equations

The morphological behaviour of a river is described by the following five equations (quasi steady approach for the water motion):

Equation for the water motion:
\[
\frac{\partial}{\partial x} \left( u \frac{Q^2}{w} \right) + g.w.a. \frac{\partial h}{\partial x} + R = 0
\] (2.1)

Equation for water continuity:
\[
\frac{\partial Q}{\partial x} = 0
\] (2.2)

Equation for sediment transport:
\[ s = f(u, C, D50, \rho_s, \rho_w, v, g) \] (2.3)

Equation for sediment continuity:
\[
\frac{\partial w.s}{\partial x} + w. \frac{\partial z}{\partial t} + w.Y = 0
\] (2.4)

Equation for alluvial roughness:
\[ C = f(u, a, D50, \rho_s, \rho_w, v, g) \] (2.5)

In which:

- \( Q \) = discharge (m^3/s)
- \( u \) = flow velocity
- \( w \) = width (m)
- \( h \) = waterlevel (m)
- \( z \) = bedlevel (m)
- \( g \) = acceleration of gravity (m/s^2)
- \( a \) = waterdepth (h-z) (m)
- \( \alpha \) = correction factor
- \( R \) = friction term
- \( C \) = Chezy value (m^(1/2)/s)
- \( s \) = sediment transport per unit width (m^2/s)
- \( D50 \) = grain size (m)
- \( \rho_s \) = grain density (kg/m^3)
- \( \rho_w \) = water density (kg/m^3)
- \( v \) = kinematic viscosity (m^2/s)
- \( Y \) = extern sediment source
The five equations obviously are not independent. For instance, the water motion influences the alluvial roughness and vice versa and the sediment transport depends strongly on the water motion.

The biggest problems arise with respect to the equation for sediment transport and the equation for alluvial roughness. For the sediment transport several formulae are available (e.g. formulae of Engelund & Hansen, Meyer-Peter & Mueller and Ackers & White). Their validity is restricted and uncertain. They give the sediment transport as a function of water motion, sediment properties and alluvial roughness and are based on experimental data and theoretical investigations. For a certain problem it has to be decided which transport formula represents the sediment transport in the best way.

The alluvial roughness (represented by the Chezy value) is often given as a constant. This avoids the problem of calculating the Chezy value as a function of water motion, sediment properties and bed forms; in fact, this reduces the five equation problem to a four equation problem. However, a number of methods exists to calculate the Chezy value in a given situation (e.g. van Rijn roughness predictor [lit[7]])

Equation (2.1) combined with eq. (2.2) can be solved (with a numerical method) when the following boundary conditions are given:

- discharge (Q)
- downstream water level (Fr < 1)
  or: upstream water level (Fr > 1)
- initial bed level

In which:

\[ Fr = \frac{u}{(g a)^{0.5}} \]  \hspace{1cm} (2.6)

For the solution of eq. (2.4) knowledge is necessary of:

- initial bed level
- upstream sediment input
3. Proceedings of the project

3.1 General remarks

To ensure a systematical development of the model, a standard method for system development was chosen: The Standard Development Methodology (SDM) as described by Eilers ([2]). According to SDM the development of a model has to be divided in several parts.

At first a thorough definition study has to be carried out. In this definition study all aspects of the problem are examined and a minute description of the demands for the model is given. This stage is concluded with a report, called Definition Study, which should be discussed with all parties concerned.

In the second part the solution techniques are developed and their implementation in the model is given. This part is also concluded with a report, called Functional and Technical Design. This report is also discussed with all participants.

Next, a computer program is written for the problems defined in the Definition Study using the techniques developed in the Functional and Technical Design. The computer program is described in a report, and in a separate report a user guide for the program is given.

Finally the program is tested. In fact, the test cases should be defined in the definition study. In this project, the test cases are not treated according to the official SDM rules. The test cases used are given in part V of this report.

3.2 Definition Study

In the definition study the capabilities of the model were discussed. In the end, the following criteria were set for the calculation of water motion, for the boundary conditions and for the morphological calculations:

1. Quasi steady flow
2. Confluences possible
3. Bifurcations NOT possible
4. Sediment uniform, although variable per grid point
5. Composite river cross sections with floodplains
6. Rectangular cross sections
7. Sediment transport depends on local conditions only
8. In a later stage it should be possible to represent the influence of riverbends by a parameter.

9. a. Chezy roughness constant or variable per grid point
   b. Nikuradse roughness constant or variable per grid point ($C = 18.1 \log (12. a/k)$)
   c. Roughness predictor is also implemented (v.Rijn)

10. Initial condition: bedlevel given at each point or given at one point plus the slope
Part I: Summary

11. Internal boundaries:
   a. construction in river
   b. local sediment withdrawal or input
   c. local water withdrawal or input
   d. sudden change of width

12. Upstream boundary:
   a. sediment input constant
      constant bedlevel
      transport calculated with transport formula
      transport equal to downstream transport of flume
      (closed circuit in experiments)
   b. discharge continuous
      discharge function of time
      discharge periodic function of time

13. Downstream boundary:
    waterlevel constant
    waterlevel function of discharge

14. Source in continuity of sediment possible (e.g.
    geological motion of riverbed)

15. The program written on an (IBM) personal computer, but
    should also be implemented on the IBM mainframe computer
    of the Delft University of Technology

16. Pascal programming language

17. Plotting and printing of output (the plot - facilities
    are not present yet)

For an argumentation on the criteria the reader is referred
   to part II of this report.

3.3 Functional and Technical Design

In this part it was discussed how the structure of the
   program should be and how the calculations should be carried
   out. To solve the partial differential equations, all
   derivatives in the equations are approached with difference -
   schemes. The user should give the value of $\Delta x$ and $\Delta t$
   with the
   wanted accuracy and the stability demands in mind (see also
   part III, IV and VI). The structure of the morphological
   model is roughly given in fig. 3.3.1.
ReadDataFromFile reads the input data from a file. DischargeCalc calculates in every timestep the discharge in each branch of the problem. WaterlevelDownstream determines the downstream boundary condition for the waterlevel of a riverbranch. BackwaterCalc calculates the flow profile for a riverbranch. SeditransCalc calculates the sediment transport at each point in a branch with the desired transport formula. SeditransUpstrm determines the upstream boundary condition for the sediment transport for a branch according to the wishes of the user. BedlevelCalc calculates the new bedlevel for every timestep. WriteDataToFile stores a new datafile on disc. With this file, the computations can be continued.

As can be seen from fig. 3.3.1 the given morphological problem should be divided into one or more riverbranches. Each branch starts at an upstream - (or internal) boundary and ends at a downstream - (or internal) boundary. Various upstream and downstream boundary conditions can be given for each riverbranch. It is e.g. possible to define a discharge - waterlevel relation on an internal boundary in case a weir is present.

First, going from the upstream end to the downstream end of the problem, the discharge in each branch is determined.
Part I: Summary

Secondly, going from the downstream end to the upstream end, the flow profile calculation is carried out. Finally, going downstream again, the sediment transport and the new bedlevel are calculated. It will be clear, that the present structure of the program does not permit the presence of bifurcations in the problem. In that case an iterative calculation of the flow profiles is necessary to compute the distribution of water discharge at the bifurcation. Besides, in that case a distribution of the sediment transport at the bifurcation should be given too.

The iterative calculation of the bedlevel takes place with a predictor-corrector method. An explicit predictor and an implicit corrector (Crank-Nicholson) is used. It is not possible to use this numerical scheme at an internal or external boundary, therefore an adapted numerical scheme is used at the boundaries.

In the Functional and Technical Design report (part III) the results are shown of morphological calculations with some test programs written in BBC-BASIC on a BBC micro computer. With these programs the numerical method (predictor-corrector) of the bedlevel calculations was tested.

3.4 Computer programs

3.4.1 CREADATA

CREADATA is a completely menu-driven program to prepare the input data for the morphological computations with ODIRMO. It is written in TURBO-Pascal on an IBM PC. The program will run on all computers with MS-DOS or PC-DOS operating system.

It consists of a main program and eleven subroutines. The main program calls the subroutines subsequently. In these subroutines all necessary input data for ODIRMO are obtained from the user and written to a file on disk. The diskfile is in ASCII format and can be transferred to other computers if needed by several procedures.

The following data are asked from the user in the various subroutines (see Part IV and VI for more detailed information):

- name of the datafile
- number of riverbranches
- time of the calculations
- timestep used in the calculations
- timelevels of output
- length of each branch
- place step in each branch
- width
- initial bedlevel
- information about discharge
- sediment transport boundary conditions
- waterlevel boundary conditions
- grain size
- transport formula (Engelund & Hansen), (Meyer-Peter & Mueller), (Ackers & White) or (Power law)
- bed roughness

In the last subroutine all data are written to a file on disk and to a printer. Next, the user can start calculations with the data.

3.4.2 ODIRMO

ODIRMO is written in the programming language Pascal on an IBM PC at the Delft University of Technology, department of Civil Engineering, section Fluid Mechanics. An implementation on the IBM mainframe of the University has been made.

ODIRMO's aim is to calculate time dependent morphological changes in alluvial streams with uniform sediment. It therefore needs knowledge about boundary and initial conditions. The input should be prepared with the program CREADATA.

ODIRMO consists of a main program and a number of subroutines. The main program controls the flow of the calculations and the calling of subroutines. In chapter 3.3 the structure of the program is already given. For a more detailed description the reader is referred to Part IV of this report.

3.5 Test cases

To test the validity of the model some test calculations are carried out. They are numbered T1 to T11 and are roughly described below (detailed information is given in Part V of this report): T1:

Small sediment overload in a river section (small timescale). The results are compared with RIVMOR calculations. This case is taken from the report "Aggradation in rivers due to overloading" by Ribberink and v.d. Sande (lit[6]), in which the RIVMOR calculations are compared with analytical approaches.

In this case the major difference between RIVMOR and ODIRMO becomes very obvious. RIVMOR produces a much steeper wave-front than ODIRMO does. And ODIRMO gives some secondary waves behind the wave front.

T2:

As T1, only this time a large sediment overload. The conclusions are the same.
Part I: Summary

T3:
As T1, only this time a large time scale. In this case there is almost no difference between ODIRMO and RIVMOR results.

T4:
As T2, only this time a large time scale. No difference between ODIRMO and RIVMOR calculations.

T5:
Sediment overload. Calculations with ODIRMO were performed until the new equilibrium was reached.

T6:
As T6, but now with a sediment 'underload'.

T7:
Calculations with a floodplain. A dredged pit in the floodplain. This case shows the shortcomings of ODIRMO in the calculation of the flow profile with floodplains. The results are not so good compared with the RIVMOR results.

T8:
Calculations with a floodplain. A dredged trench in the mainstream. The results of these calculations are better than the results of case T7 (compared with RIVMOR).

T9:
Calculations with a discontinuity in the initial bedlevel. Three timesteps are carried out to compare the calculation of the flow profile and the development of the sedimentation wave front with RIVMOR.

T10:
River bend cut-off. This case is not compared with RIVMOR calculations. The results are given to indicate what kind of results a user might expect for his own case. Calculations are carried out with some different time- and placesteps. This can give an impression of the accuracy and the numerical diffusion.

T11:
Confluence of two rivers with a weir in the downstream section. As case T9, the calculations are not compared with other calculations.
4. Evaluation

4.1 Project

After one year of work two computer programs are created: ODIRMO and CREADATA. It would, however, not be surprising if they still contain big mistakes, irritating omissions or obscure failures. I apologize for any discomfort that may be caused by working with these programs, but I can not be held responsible for any harm or damage directly or indirectly caused by them.

From the test results can be concluded, that morphological effects are rather well represented by the model. A certain disadvantage is, that the accuracy and numerical diffusion of the used predictor corrector method strongly depends on the input from the user. The user has to have some insight in the numerical effects. An advantage of the scheme is, that it is stable for Courant numbers up to about 1.7.

The computation of the flow profile may cause trouble in some cases. Especially in cases with floodplains with strongly varying width and in cases with steep gradients in the waterlevel.

It can be rather irritating that it is not possible to edit an existing input datafile. There was no time to write a separate program for this possibility. Also the absence of plotting facilities is rather frustrating.

With the above in mind, the following recommendations for the future are given:

- improve flow profile calculation (with floodplains)
- implement a check on Courant number and numerical diffusion
- write the program EDITDATA
- add plotting facilities

and for expansion of the model:

- introduce the possibility of bifurcations
- introduce the possibility of non-uniform sediment
- implement Galappatti’s model for sediment transport

4.2 SDM Method

It is concluded that is was very useful to work according to a described standard method. The major benefits are:

1. There are some close defined stages in the development at which a report must be presented, so that all participants can give their comments on the proceedings so far and their demands for the following stages.
2. One is not likely to overlook a major aspect of the problem, as one is forced to systematically examine all aspects of the problem given.
At first sight, the disadvantage seems to be that the whole procedure is rather time consuming, but I am personally convinced that no time is really wasted. It should be kept in mind that a model developed in this way is very transparent and straightforward, and it will be easy to implement changes and to find remaining (minor?) failures in the program.

The followed procedure System Development Methodology, however, is probably not the best procedure for technical systems. It seems more appropriate for administrative problems. To my knowledge there not yet exists a procedure special suitable for technical problems.
5. References


[6] Ribberink, J.S. & van der Sande, J.T.M. (1984), Aggradation In Rivers Due To Overloading, Delft University of Technology, Dept. of Civil Engineering, Communications on Hydraulics rep.84-1 Also published in JHR, vol.23, 1985, No.3


[8] Vreugdenhil, C.B. (1979); Waterloopkundige Berekeningen I, Delft University of Technology, Dept. of Civil Engineering, Lecture notes b84

[9] Vreugdenhil, C.B. (1980); Waterloopkundige Berekeningen II, Delft University of Technology, Dept. of Civil Engineering, Lecture notes b85


[12] De Vries, M. (1981); Morphological Computations, Delft University of Technology, Dept. of Civil Engineering, Lecture notes f10A
Part II: Definition Study

Contents

1. INLEIDING 2
2. BESTAANDE SITUATIE 3
   2.1 Inventarisatie
   2.2 Evaluatie
3. PROBLEEMSTELLING 4
   3.1.1 Keuzepunten morfologisch model 5
   3.1.2 Keuzepunten hardware/software 6
   3.2.1 Criteria en randvoorwaarden 11
   3.2.2 Keuze argumentatie 12
   3.3 Probleemdefinitie 13
   3.4 Systeemschets 14
4. INV OEFING 13
   4.1 Acceptatiecriteria
   4.2 Testen in praktijk
5. WERKPLAN/TIJDSCHEMA 14
1. INLEIDING

Als in de loop van een (getij)rivier (of estuarium) ingegrepen wordt, door de mens, bijvoorbeeld door onttrekking of toevoer van water of bodemmateriaal, of door de natuur zelf, bijvoorbeeld door geologische processen, wordt de oorspronkelijke evenwichtssituatie in die rivier verstoord en zal de rivier gaan streven naar een nieuwe evenwichtssituatie. Dit gaat gepaard met veranderingen in de bodemligging ten gevolge van erosie en aanzanding. Het is voor de mens vaak van belang te weten, hoe die bodemveranderingen zich in de tijd afspelen, bijvoorbeeld alsproblemen voor de scheepvaart in het geding zijn.

Op de Afdeling Civiele Techniek van de Technische Hogeschool Delft bestaat bij de vakgroep Waterbouwkunde behoefte aan een model met bijbehorend computerprogramma, dat naar huidige inzichten berekeningen uitvoert aan zulke bodemveranderingen. Zo'n programma zou zowel geschikt moeten zijn voor produktiedoeleinden, en als zodanig door bijvoorbeeld afstudeerders te gebruiken, als voor verdere studie, bijvoorbeeld gevolgd door uitbreiding of verbetering van het programma.

Het doel van dit project is, ervoor te zorgen, dat een dergelijk programma beschikbaar komt voor de TH. Vermeld dient nog te worden, dat hier grootschalige bodemveranderingen bedoeld worden en dat niet gekeken wordt naar de bodemligging op kleine schaal (ribbel- en duinvorming).
Part II: Definition Study

2. BESTAANDE SITUATIE

2.1 Inventarisatie


2.2 Evaluatie

De bestaande situatie kan nauwelijks bevredigend genoemd worden. De programma's van het WL zijn niet beschikbaar voor studiedoeleinden en er moet veel geregeld worden voordat een student met een afstudeerprobleem van RIVMOR gebruik kan maken. Er moet hier wel opgemerkt worden, voordat een verkeerde indruk gewekt wordt, dat het WL altijd zeer welwillend zijn programma's ter beschikking stelt, het is echter voor de TH natuurlijk nog een stuk prettiger als een programma direct bij de hand is.

En de programma's van de TH zelf zijn niet speciaal gericht op morfologische problemen, wat met zich meebrengt, dat de programma's niet goed aansluiten bij specifiek morfologische aspecten, zoals het verschil in tijdschaal tussen bodembeweging en waterbeweging. Bovendien blijkt het gebruik van deze programma's tijdrovend, mede omdat de beperkingen van de modellen minder duidelijk aangegeven zijn.
## 3. PROBLEEMSTELLING

### 3.1.1 Keuzepunten morfologisch model

Bij het opzetten van een morfologisch zijn veel aspecten te onderkennen, die al dan niet door het model kunnen worden meegenomen. Van deze aspecten is een lijst opgesteld, aan de hand waarvan gekozen wordt.

Deze lijst ziet er als volgt uit:

1. Waterbeweging
   a. stationair
   b. niet stationair
2. Riviersamenvloeiingen
3. Riviersplitsingen
4. Ingewikkelder netwerken
5. Sediment
   a. uniform
   b. uniform doch per punt variabel
   c. niet uniform
6. Sedimenttransport
   a. transportformule
   b. ander model (bijv. Galappatti)
7. Samengesteld dwarsprofiel
   a. met transport in uiterwaarden
   b. zonder transport in uiterwaarden
8. Eovererosie
9. Armouring
10. Correctie voor wash-load
11. Andere dan rechthoekige profielen
12. Mogelijkheid van bochten
13. Chezy waarde
   a. constant
   b. variabel per punt
   c. \( C=181\log_{12}a/k_n \), \( k_n \) constant
   d. \( C=181\log_{12}a/k_n \), \( k_n \) variabel per punt
   e. uit ruwheidsvoorspeller
14. Beginvoorwaarden:
   a. bodemligging overal op te geven
   b. bodemligging berekenen met verhang
15. Interne randvoorwaarden:
   a. wateronttrekking/toevoer
   b. sedimentonttrekking/toevoer
   c. overlaat/stuw
   d. breedteverandering
16. Bovenstroomse randvoorwaarde water
   a. debiet constant
   b. debiet periodiek in de tijd
   c. debiet willekeurige functie van tijd
Part II: Definition Study

17. Bovenstroomse randvoorwaarde sediment
   a. sedimenttoevoer constant
   b. bodemniveau constant
   c. transport uit transportformule
   d. transport gelijk aan transport aan einde goot (gesloten circuit)
   e. sedimenttoevoer functie van tijd
   f. bodemligging functie van tijd
18. Benedenstroomse randvoorwaarde
   a. waterniveau constant
   b. waterniveau functie van tijd
   c. waterniveau functie van debiet
   d. waterniveau functie van debiet en tijd
19. Dalende/stijgende rivierbodem

3.1.2 Keuzepunten hardware/software

Ook met betrekking tot het computergebruik zijn nog enkele punten ter overweging aan te wijzen. Deze punten zijn:

20. Waarop moet het programma kunnen draaien
   a. mainframe computer
   b. minicomputer
   c. microcomputer
21. Welke programmeertaal
   a. Algol
   b. Fortran
   c. Pascal
22. Programma aspecten
   a. plotten en printen van in- en uitvoer
   b. doorstarten met gemaakte berekeningen
   c. programma- en datastructuur duidelijk
3.2.1 Criteria en randvoorwaarden

Bij het kiezen uit bovenstaande aspecten zijn als criteria en randvoorwaarden gehanteerd:

a. de mening van de vakgroep rivierwaterbouwkunde
b. de mening van de vakgroep kustwaterbouwkunde
c. de huidige stand van de wetenschap
d. de beperkte tijdsduur van het project
e. de beschikbare hardware en de karakteristieken ervan (randapparatuur)

3.2.2 Keuze argumentatie

De punten uit de paragrafen 3.1.1 en 3.1.2 zullen stuk voor stuk worden nagegaan en de keuze wordt beargumenteerd.

ad 1
STATIONAIRE OF NIET STATIONAIRE STROMING

Vooralsnog wordt uitgegaan van een model, dat met quasi-stationaire waterbeweging rekent. Dit is enerzijds gedaan om de tijdsbesteding te beperken en anderzijds omdat het mogelijk zal blijven later een andere subroutine voor niet stationaire waterbeweging toe te voegen. Deze keuze betekent, dat het model geschikt zal zijn voor rivierproblemen met grote tijdschalaal en ongeschikt voor getijgebieden en hoogwatergolven op rivieren.

ad 2, 3 en 4
SAMENVLOEIINGEN/SPLITTINGSINGEN

Samenvloeiingen leveren geen bijzondere problemen op, doch splitsingen wel. Ten eerste moet bij splitsingen iteratief de waterstand op het splitsingspunt bepaald worden, en bij toename van het aantal splitsingen moet er steeds meer geltereerd worden; dit doet het rekenwerk ernstig toenemen. En ten tweede is a priori niet duidelijk hoe het sediment zich op de splitsing zal verdelen. Hiervoor moet een soort verdeelsleutel aangenomen worden, die uit metingen, schaalmodellen of twee- of driedimensionale beschouwingen moet komen.

Met het oog hierop is uiteindelijk besloten voor het te ontwikkelen model slechts samenvloeiingen toe te staan. Dus geen splitsingspunten.

ad 5
UNIFORM/NIET UNIFORM SEDIMENT

Er wordt uitgegaan van uniform sediment in de rivier, doch per punt kan de korreldiameter nog variëren. Voor niet uniform sediment bestaat bij de vakgroep Vloeistofmechanica een mathematisch model, dat echter nog niet voldoende uitgewerkt en getest is. Als dit (niet uniforme) model operationeel is, moet het zonder al te grote problemen in het morfologisch model inpasbaar zijn.
Part II : Definition Study

ad 6
TRANSPORTBERKENING
Bij gebruik van een transportformule wordt er van uitgegaan,
dat het transport uitsluitend afhankelijk is van de momentane
plaatselijke condities. Dit is voor bodemtransport over het
algemeen een redelijke veronderstelling. Voor gesuspendeerd
materiaal, waarvan de concentratie vertikaal zich met een
bepaalde traagheid aan veranderingen in de condities aanpast
is deze veronderstelling niet altijd juist. Voor die gevallen
is een model ontwikkeld door Galappatti (1983). Dit model
wordt niet als optie in het te schrijven model meegenomen,
doch moet eventueel later wel in te passen zijn. Er zal dus
voorlopig alleen gewerkt worden met verschillende bekende
transportformules.

ad 7
SAMENGESTELD PROFIEL
Hieronder wordt verstaan een rivier met uiterwaarden, waarbij
de uiterwaarden geschematiseerd worden als een extra stuk
dwarsprofiel. De stuwkromme wordt indien nodig berekent voor
een rivier met uiterwaarden, waarbij wordt verondersteld, dat
tussen de hoofdstrom en de uiterwaarden geen impuls-
uitwisseling plaatsvindt.
Er wordt gerekend zonder sediment transport in de
uiterwaarden. Uiterwaarden zijn meestal begroeid en het
eventuele transport in de uiterwaarden kan dan zeker niet met
een transportformule worden beschreven, omdat het transport
mede afhankelijk is van het sedimentaanbod en sediment
uitwisseling met de hoofdstrom.

ad 8, 9 en 10
OEVEREROSIE, ARMOURING, WASH-LOAD
Deze drie effecten worden niet in het model opgenomen, omdat
de fysische en mathematische modellering van deze fenomenen
nog niet voldoende is uitgezocht. In principe is het wel
mogelijk in een later stadium wat mogelijkheden in te bouwen
deze fenomenen wel in rekening te brengen.

ad 11
ANDERE PROFIELEN
Hieronder wordt verstaan andere dan rechthoekige dwars-
profieren. Dit soort profielen zijn aan de orde bij
berekeningen aan beken en irrigatiekanalen bijvoorbeeld. Het
rekenen met dit soort profielen levert wel enige
moeilijkheden op. Ten eerste kan men niet meer rekenen met de
waterdiepte a, doch moet men de hydraulische straal R
gebruiken. En ten tweede moet via de discutabele hypothese
van Einstein een verdeling van de waterstroom bepaald worden,
om de schuifspanning aan de bodem te vinden, die ten slotte
bepalend is voor het transport. Deze optie wordt niet in het
model opgenomen om de tijdsbesteding te beperken en omdat er
al een programma bestaat (v.d. Kolff), dat speciaal voor dit
soort gevallen is geschreven.
Het wordt wel mogelijk de sediment transportenderende breedte
ongelijk te kiezen aan de breedte van de hoofdstroom, zodat eventueel met deze optie nog iets bereikt kan worden in het geval de rivier niet met rechthoekig dwarsprofiel geschematiseerd kan worden.

ad 12
BOCHTEN
Rivieren zijn helaas niet recht. Integendeel zelfs, ze zijn in het algemeen zeer bochtig. Dit maakt het gewenst ook bochtverschijnselen mee te kunnen nemen. Met name de vakgroep rivierwaterbouwkunde hecht veel waarde aan deze mogelijkheid. Het probleem is, dat de tweedimensionale bochteffecten (spiraalstroming, dwarsverhang bodem en waterspiegel) moeilijk op een verantwoorde manier in een eendimensionaal model te krijgen zijn. Te denken valt aan het meenemen van een parameter r, de plaatselijke bochtstraal, aan de hand waarvan het dwarsverhang ter plaatse en het transport berekend kunnen worden. De optie om een dergelijke optie in het model in te voegen moet aanwezig zijn, echter de enorme traagheid (na-ijling) van de spiraalstroming en dwarsverhang maakt het in feite onverantwoord enige conclusies te verbinden aan de uitkomsten. Vooralsnog berekenen we de rivieren maar alsof ze recht zijn.

ad 13
CHEZY WAARDE

ad 14
BEGINVOORWAARDE
Een belangrijk punt is, dat overal in het beschouwde gebied de bodemligging opgegeven kan worden, en dat waar geen metingen beschikbaar zijn de bodeligging berekend kan worden met behulp van het bodemverhang.

ad 15
INTERNE RANDVOORWAARDEN

ad 16
DEBIET RANDVOORWAARDE
Op de bovenstroomse rand van het probleem moet de waterafvoer opgegeven worden. Deze afvoer moet kunnen worden opgegeven als een constant debiet, een periodiek debiet (bijv. een jaarlijkse cyclus) of een willekeurige functie voor de totale beschouwde periode.
Part II : Definition Study

ad 17
TRANSPORT RANDVOORWAARDE
Behalve het debiet moet bovenstrooms ook voor het sedimenttransport een randvoorwaarde worden opgegeven. De volgende opties zullen in het model mogelijk zijn:

a. sedimenttoevoer constant
b. bodemniveau constant
c. transport uit transportformule
d. transport gelijk aan transport aan einde goot (gesloten circuit)

De overige genoemde mogelijkheden zijn in de praktijk onbelangrijk.

ad 18
BENEDENSTROOMSE RANDVOORWAARDE
Benedenstrooms dient de waterstand opgegeven te worden. Dit kan, zoals aangegeven, direct of via een relatie met het debiet. Buiten getijgebied zijn de belangrijkste mogelijkheden:

a. waterniveau constant
b. waterniveau functie van debiet

Bovendien zal, omdat dat geen problemen oplevert, het waterniveau ook als functie van de tijd opgegeven kunnen worden.

De laatste mogelijkheid, waterniveau als functie van tijd en debiet is nogal lastig te implementeren en wordt dan ook achterwege gelaten. Dit is wel wat jammer, daar dit wel degelijk soms het geval is (zoals bijv. bij stuwen met een bepaald afvoerplan).

ad 19
DALENDE/STIJGENDE RIVIERBODEM
De bodemligging kan ook veranderen door bijvoorbeeld baggeren of mijnstortingen. Dit betekent een bronterm in de continuïteitsvergelijking voor het sediment. Deze mogelijkheid wordt meegenomen. Het wordt dan mogelijk allerlei sedimentbronnen of putten in te voeren, zoals sediment dat, uit de kribvakken, in de hoofdstroom wordt getrokken door de scheepvaart.

ad 20
HARDWARE
Het programma wordt geschreven op een IBM personal computer. Er wordt echter tevens gezorgd voor een versie die op de IBM mainframe computer van de TH kan draaien. Belangrijke beperkingen aan een programma op een microcomputer kunnen zijn:
- onredelijk lange rekentijden
- te omvangrijke datastructuren
- te omvangrijk programma
- benodigde software (beschikbare routines)

Er blijkt, dat vooral rekentijden op micro computers nogal uit de hand kunnen lopen. Met het oog hierop wordt aandacht besteed aan de mogelijkheid op de micro voorbereide invoer
Part II: Definition Study

over te sturen naar een mainframe, alwaar de berekeningen plaats vinden. Voor niet te omvangrijke problemen hoeft de micro echter niet te worden 'verlaten'.

ad 21
PROGRAMMEERTAAL
Er is gekozen voor de programmeertaal Pascal. Pascal heeft de voorkeur gekregen boven Fortran, opdat de programmastructuur wat helderder wordt. Een van de eisen aan het model is, dat er later nog aan gesleuteld moet kunnen worden en dan is een doorzichtig programma erg belangrijk. Bovendien leren alle TH studenten tegenwoordig Pascal. In Fortran bestaan meer standaardtouitines (bijv. voor in- en uitvoer); eventueel zijn deze echter ook vanuit Pascal programma's aan te roepen. Algol valt af, omdat het een te specifieke, uitstervende universiteits taal is.

ad 22
IN- EN UITVOER/DOORSTARTEN
Het is belangrijk, dat de invoer gecontroleerd kan worden op fouten; dit maakt het belangrijk, dat van de invoer een plot gemaakt kan worden, zodat een fout snel opvalt. Ook de uitvoer moet geplot kunnen worden, om een snelle interpretatie van de resultaten duidelijk te maken. Het verder rekenen met verkregen resultaten moet zonder problemen mogelijk zijn. Dit betekent, dat de resultaten op een geschikte wijze moeten worden opgeslagen.
3.3 Probleemdefinitie

Samengevat is het probleem nu het opzetten van een 1D morfologisch model voor rivieren waarin de in de vorige paragraaf gekozen aspecten worden meegenomen.

Dit betekent:
1. Waterbeweging stationair
2. Riviersamenvloeiingen
3. Geen riviersplitsingen
4. Sediment uniform doch per punt variabel
5. Sedimenttransport met transportformule
6. Samengesteld dwarsprofiel zonder transport in uiterwaarden
7. Uitsluitend rechthoekige profielen
8. Chezy waarde
   a. constant
   b. variabel per punt
   c. \( C=18 \log_{10} a/k \), \( k \) constant
   d. \( C=18 \log_{10} a/k \), \( k \) variabel per punt
   e. uit ruwheidsvoorspeller van Van Rijn
9. Beginvoorwaarden:
   a. bodemligging overal op te geven
   b. bodemligging berekenen met verhang
10. Interne randvoorwaarden:
    a. wateronttrekking/toevoer
    b. sedimentonttrekking/toevoer
    c. overlaat/stuw
    d. breedteverandering
11. Bovenstroomse randvoorwaarde water
    a. debiet constant
    b. debiet periodiek in de tijd
    c. debiet willekeurige functie van tijd
12. Bovenstroomse randvoorwaarde sediment
    a. sedimenttoevoer constant
    b. bodemniveau constant
    c. transport uit transportformule
    d. transport gelijk aan transport aan einde goot (gesloten circuit)
13. Benedenstroomse randvoorwaarde
    a. waterniveau constant
    b. waterniveau functie van tijd
    c. waterniveau functie van debiet
14. Bronterm in sedimentcontinuïteit mogelijk
15. Programma moet kunnen draaien op
    a. mainframe computer
    b. microcomputer
16. Programmeertaal Pascal
17. Programma aspecten
    a. plotten en printen van in- en uitvoer
    b. doorstarten met gemaakte berekeningen

Bovendien dient het programma dermate doorzichtig van structuur te zijn, dat uitbreidingen zoals in de beargumentering aangegeven (zoals model voor zwevend
Part II: Definition Study

transport en niet uniform materiaal) zonder onoverkomelijke problemen kunnen worden toegevoegd.

3.4 Systeemschets

In figuur 3.4.1 is een globale indruk van het op te zetten systeem weergegeven.

<doorstarten met eerdere berekeningen>

INVOER

parameters
randvoorwaarden
beginvoorwaarde bodemligging

CALCULATIES

waterbeweging
sedimenttransport
nieuwe bodemligging

UITVOER

bodemligging na gewenst tijdsverloop

fig. 3.4.1 Modelopzet
4. INVOERING

4.1 Acceptatiecriteria

Als het programma geschreven is, moet gecontroleerd worden of het aan de gestelde eisen voldoet. Dat wil zeggen, er moet gecontroleerd worden of het programma inderdaad met de gewenste nauwkeurigheid de oplossing geeft van het stelsel differentiaalvergelijkingen voor sediment en water voor de gevallen aangegeven in paragraaf 3.3. Dit betekent dat uitkomsten van testberekeningen vergeleken moeten kunnen worden met analytische oplossingen ofwel met andere oplossingen waarvan de juistheid op andere wijze vastgesteld is of kan worden (bijvoorbeeld vergelijking met RIVMOR berekeningen).

4.2 Testen in praktijk

Een andere zaak is, of het model ook werkelijke rivierproblemen op kan lossen. Dit hangt af van de juistheid van het mathematisch model en van de gevoeligheid van de oplossing voor de verschillende parameters. Dit kan binnen het kader van dit project voor enige gevallen gecontroleerd worden, wellicht door uitkomsten te vergelijken met bestaande meetresultaten. Hierbij kan ook gekeken worden naar de gevoeligheid van de oplossing voor verandering van parameters (zoals Chezy waarde) of transportformule.
5. WERKPLAN/TIJDSCHEMA


Aan de hand van deze methode is het volgende werkplan opgesteld:

1. Definitiestudie 4 weken
2. Functioneel ontwerp 6 weken
3. Technisch ontwerp/programmeren 15 weken
4. Acceptatie/testen 4 weken
5. Aanpassingen na testen 5 weken
6. Handleiding + programmabeschrijving 4 weken

totaal 38 weken

Er moet bekend worden, dat het werkplan zoals hierboven gegeven, enigszins afwijkt van het oorspronkelijke werkplan en is aangepast aan de werkelijke tijdsbesteding. Het punt "aanpassingen na testen" is genoemd om alle wijzigingen en uitbreidingen, die na de werkelijke programmeerfase zijn geschiedt te verdisconteren.
Contents

1. Introduction
2. Summary Definition Study
3. Calculation of water motion
   3.1 Equations
   3.2 Numerical solution without floodplains
   3.3 Numerical solution with floodplains
   3.4 Calculation of friction term
4. Calculation of bedlevel
   4.1 Equations
   4.2 Numerical solution
      4.2.1 'Inside' scheme
      4.2.2 Boundary schemes
   4.3 Test of numerical scheme
   4.4 Various transport formulae

Appendices

A. Modified equation boundary schemes
B. Test programs
1. Introduction

In this part of the ODIRMO report, the functional and technical design are treated. In chapter 2 the conclusions of the Definition Study are resumed.

In chapter 3 the equations for the water motion are discussed and their numerical solution with a an iterative finite difference method is treated. A difference is made between the calculation with and without floodplains. The equations (continuity and motion) are solved with a quasi steady approach.

In chapter 4 the equations for the bed level are discussed (sediment transport and continuity). In par. 4.2 the numerical solution of the equation for sediment continuity is treated. For this solution a predictor corrector method is chosen. Some conclusions about stability and accuracy of the scheme are reviewed. Special attention is paid to the numerical scheme used on the boundaries. In par. 4.3 the numerical scheme is tested with a test computer program. In par. 4.4 the transport formulae implemented in the model are discussed.

Originally this report has to be concluded with a description of the computer model. As ODIRMO is described separately in Part IV of this report, this is left out.
2. Definition Study

In the definition study the capabilities of the model were discussed. In the end, the following criteria were set for the calculation of water motion, for the boundary conditions and for the morphological computations:

1. Quasi steady flow
2. Confluences possible
3. Bifurcations not possible
4. Sediment uniform, although variable per grid point
5. Composite river cross sections with floodplains
6. Rectangular cross sections
7. Sediment transport depends on local conditions only
8. In a later stage it should be possible to represent the influence of river bends by a parameter
9. a. Chezy roughness constant or variable per grid point
    b. Nikuradse roughness constant or variable per grid point
    \( C = 18 \log_{12} a / \text{Kn} \)
    c. Roughness predictor implemented (v. Rijn)
10. Initial condition: bedlevel given at each point or given at one point plus the slope
11. Internal boundaries:
    a. construction in river
    b. local sediment withdrawal or input
    c. local water withdrawal or input
    d. sudden change of width
12. Upstream boundary:
    a. sediment input constant
    bedlevel constant
    transport calculated with transport formula
    transport equal to transport at end of flume
    b. discharge constant
    discharge function of time
    discharge periodic function of time
13. Downstream boundary:
    waterlevel constant
    waterlevel function of discharge
14. Source in continuity of sediment possible (e.g. geological motion of riverbed)
15. Program written on an (IBM) personal computer, but should also be implemented on the IBM mainframe computer of the Delft University of Technology
16. Pascal programming language
17. Plotting and printing of input and output (plotting facilities are not present yet)
3. Calculation of water motion

3.1 Equations

The water motion in a river section with rectangular cross section (fig. 3.1.1) can be described by the following equations:

equation for water motion:

\[
\frac{Q + \frac{\partial}{\partial x} (\alpha Q^2) + g \cdot w \cdot a \cdot \frac{\partial h}{\partial x} + g \cdot Q^2}{\frac{\partial^2}{\partial x^2} C^2 \cdot w \cdot a \cdot R} = 0
\]  

(3.1.1)

equation for water continuity:

\[
\frac{w \cdot \frac{\partial h}{\partial t} + \frac{\partial w}{\partial x}}{\frac{\partial w}{\partial x}} = 0
\]  

(3.1.2)

equation for alluvial roughness:

\[
C = f(u, a, D_{50}, P_s, P_w, v, g)
\]  

(3.1.3)

In which:

- \(Q\) = discharge (m\(^3\)/s)
- \(u\) = flow velocity (m/s)
- \(w\) = width (m)
- \(h\) = water level (m)
- \(z\) = bed level (m)
- \(g\) = gravitation (m\(^{(1/2)}/s\))
- \(a\) = factor depth (h-z)
- \(\alpha\) = factor
- \(C\) = Chezy roughness
- \(R\) = hydraulic radius (m)
- \(D_{50}\) = grain size (m)
- \(P_s\) = grain density (kg/m\(^3\))
- \(P_w\) = water density (kg/m\(^3\))
- \(v\) = kinematic viscosity (m\(^2/s\))

In the quasi steady approach all time derivatives are zero.
If it is assumed that $\alpha = 1$ equation (3.1.1) can be rewritten as:

$$2Q \frac{\partial Q}{\partial x} - \frac{Q^2}{g} \frac{\partial w}{\partial x} - \frac{Q^2}{g} \frac{\partial a}{\partial x} + \frac{\partial h}{\partial x} + \frac{Q^2}{C^2w^2a^2R} = 0$$

(3.1.4)

and (3.1.2) as:

$$\frac{\partial Q}{\partial x} = 0 \quad (Q = f(t))$$

(3.1.5)

Combining eqs. (3.1.4) and (3.1.5) we get:

$$\frac{\partial h}{\partial x} = \frac{Q^2}{g} \frac{\partial w}{\partial x} + \frac{Q^2}{g} \frac{\partial a}{\partial x} - \frac{Q^2}{C^2w^2a^2R}$$

(3.1.6)

Eq. (3.1.6) is used in this project to calculate the backwater curve. The first two terms on the right hand side of the equation are the convective terms due to change of river width and river depth respectively. The last term is the friction term. In the friction term the (in principle unknown) Chezy value is present (eq. 3.1.3). The main difficulty is, that the flow velocity and the roughness influence each other, so that an iterative calculation of the Chezy value is necessary. The problem can be avoided by giving $C$ a constant value. In fact, this is almost always done. But there are some relations developed between the various parameters and $C$. One of them (the Van Rijn roughness predictor (1982)) is implemented in ODIRMO.

Apart from this, equation (3.1.6) can only be solved iteratively, as the water level $h$ is present on both sides of the equation ($a = h - z$).

The numerical solution of the equation is discussed in the next paragraphs.
3.2 Numerical solution without floodplains

In this paragraph, the numerical solution of eq. (3.1.6) is treated. The friction term is treated separately in par. 3.4 and will in this paragraph be indicated by $F$.

Consider the downstream river section given in fig. 3.2.1.

Rewrite eq. (3.1.6) as:

$$\frac{\partial h}{\partial x} = \frac{Q^2}{2 g a^2 w} (\frac{1}{a^2} \frac{\partial a}{\partial x} + \frac{1}{w^2} \frac{\partial w}{\partial x}) - F \tag{3.2.1}$$

In this equation all derivatives can be approached with a finite difference:

$$\frac{\partial h}{\partial x} = h^n - h^{n-1} + O(\Delta x) \tag{3.2.2}$$

$$\frac{\partial a}{\partial x} = a^n - a^{n-1} + O(\Delta x) \tag{3.2.3}$$

$$\frac{\partial w}{\partial x} = w^n - w^{n-1} + O(\Delta x) \tag{3.2.4}$$

This can be shown with a Taylor expansion:

$$h^{n-1} = h^n - \Delta x \frac{\partial h}{\partial x} + \frac{\Delta x^2}{2} \frac{\partial^2 h}{\partial x^2} + O(\Delta x^3) \tag{3.2.5}$$

Eq. (3.2.5) is exactly equivalent with eq. (3.2.2).

Now we want to calculate $h^{n-1}$ from $h^n$ and the other parameters. The known parameters are: $g$, $w$ (everywhere), $Q$ and the friction term (see par. 3.4 for the friction term). So we can predict $h^{n-1}$ with:

$$h^{n-1} = h^n - \frac{Q^2}{2 g w} (\frac{w^n - w^{n-1}}{g}) + \Delta x \cdot F^n \tag{3.2.6}$$
with \( u = \frac{Q}{(w \cdot a)} \) water flow velocity.

With the estimated \( h^{n-1} \), an estimated waterdepth at \( n-1 \) can be calculated with:

\[
a^{n-1} = h^{n-1} - z^{n-1}
\]  

This value of \( a^{n-1} \) is used for the calculation of the convective term \( \frac{\partial a}{\partial x} \) and the flow velocity at \( n-1 \).

Now \( h^{n-1} \) is corrected with:

\[
h^{n-1} = h^n - u^n \cdot 2 \cdot \frac{(w^n - w^{n-1})}{g \cdot w^n} \cdot \left( \frac{a^n - a^{n-1}}{g \cdot a^n} \right) + \frac{\Delta x \cdot F^n}{2} - \frac{\Delta x \cdot F^{n-1}}{2}
\]  

The last step can be repeated until a certain accuracy is reached. It appears, that in most cases two correction steps are enough for a high accuracy (roughly less than a millimeter). In ODIRMO two corrections are carried out.

With the found value of the waterlevel at \( n-1 \) we can proceed to point \( n-2 \) etc., until the upstream boundary is reached.
3.3 Numerical solution with floodplains

In case of a river with floodplains, the distribution of the total discharge between the mainstream and the floodplain has to be calculated. Consider the downstream river section as in fig. 3.2.1, but now with a composite cross section (fig. 3.3.1).

![Composite cross section](image)

In this case \( h_{e-1} \) can not be predicted without first knowing the value of \( Q_1 \). Instead of \( Q_1 \) and \( Q_2 \) a variable \( r \) is introduced:

\[
Q_1 = r \cdot Q \quad (3.3.1)
\]

\[
Q_2 = (1-r) \cdot Q \quad (3.3.2)
\]

Using compatibility the value of \( r \) can be computed. Assume:

\[
h_1 = h_2 \quad \text{and} \quad \frac{\partial h_1}{\partial x} = \frac{\partial h_2}{\partial x} \quad (3.3.3)
\]

Combining equations (3.2.6) and (3.3.3) we get:

\[
\frac{(r \cdot Q)^2}{w_1 \cdot e^2 \cdot a_1 \cdot e^2} (w_1 \cdot e - w_1 \cdot e^{-1}) - \Delta x \cdot F_1 \cdot e = \frac{\left((1-r) \cdot Q\right)^2}{w_2 \cdot e^2 \cdot a_2 \cdot e^2} (w_2 \cdot e - w_2 \cdot e^{-1}) - \Delta x \cdot F_2 \cdot e \quad (3.3.4)
\]

From eq. (3.3.4) a prediction of \( r^n \) can be calculated, as \( r \) is the only unknown in eq. (3.3.4).

With this predicted \( r^n \), a predictor for \( h_{e-1} \) is calculated with:

\[
h_{e-1} = h_e - \frac{\left(r^n \cdot Q\right)^2}{w_1 \cdot e^2 \cdot a_1 \cdot e^2} (w_1 \cdot e - w_1 \cdot e^{-1}) + \Delta x \cdot F_1 \cdot e \quad (3.3.5)
\]

With the predictor \( h_{e-1} \), the values of \( a_1 \cdot e^{-1} \) and \( a_2 \cdot e^{-1} \) can
be predicted. It is now possible to correct \( r^* \) using eqs. (3.2.8) and (3.3.3):

\[
\frac{(r, Q)^2}{w_1, a_1, 2} \cdot \frac{(w^{1, e} - w^{1, e-1}) - (a^{1, e} - a^{1, e-1})}{g, w_1, e} - \Delta x, F^{1, e} = (3.3.6)
\]

\[
\frac{(1-r, Q)^2}{w_2, a_2, 2} \cdot \frac{(w^{2, e} - w^{2, e-1}) - (a^{2, e} - a^{2, e-1})}{g, a_2, e} - \Delta x, F^{2, e}
\]

A predictor for \( r^{*-1} \) can now be found with:

\[
\frac{(r, Q)}{w_1, e-1, 2} \cdot \frac{(w^{1, e-1} - w^{1, e-2})}{g, w_1, e-1} - \Delta x, F^{1, e-1} = (3.3.7)
\]

With the corrected \( r^* \) and the predicted \( r^{*-1} \) a corrector for \( h^{*-1} \) is found with:

\[
\frac{h^{*-1}}{2} = h^* - \frac{(r^*, Q)^2}{2, w_1, e, 2} \cdot \frac{(w^{1, e} - w^{1, e-1}) + (a^{1, e} - a^{1, e-1})}{g, w_1, e, g, a_1, e} + \Delta x, F^{1, e-1}
\]

(3.3.8)

With the corrected \( h^{*-1} \) this process can be repeated:

- correct \( r^* \) with eq. (3.3.6), \( r^{*-1} \) with eq. (3.3.7) and \( h^{*-1} \) with eq. (3.3.8) again. In ODIRMO the process is repeated one time. So, resumed:

1) predict \( r^* \)
2) predict \( h^{*-1} \)
3) correct \( r^* \)
4) predict \( r^{*-1} \)
5) correct \( h^{*-1} \)
6) correct \( r^* \)
7) correct \( r^{*-1} \)
8) correct \( h^{*-1} \)

After this, we continue to point n-2 etc. until the upstream end of the branch is reached. It is remarked, that the distribution of the discharge between the mainstream and the floodplain does not depend on the upstream situation in this case. This may lead to wrong results, when this distribution is strongly influenced by the upstream situation (e.g. when a large dredged pit is present in the floodplain).

It is remarked, that the friction term, represented by \( F \) in this case, also contains a term with \( (r, Q)^2 \), so the
Part III: Functional and Technical Design

The equations given above are slightly more complicated than they might seem.

### 3.4 Calculation of friction term

In ODIRMO the friction term in the backwater equation can be calculated in several ways according to the wish of the user. The friction term, as represented by $F$ in the previous paragraphs can be written as:

$$F = \frac{u^2}{C^{2.0}}$$  \hspace{1cm} (3.4.1)

In which:

- $u$ = water flow velocity (m/s)
- $C$ = Chezy roughness (m^{1/2}/s)
- $R$ = hydraulic radius (m)

$F$ is dimensionless.

In ODIRMO both $C$ and $R$ can be determined in several ways. For the hydraulic radius $R$:

$$R = a \text{ (waterdepth)}$$  \hspace{1cm} (3.4.2)

or:

$$R_b = \frac{a \cdot w}{2 \cdot a + w}$$  \hspace{1cm} (3.4.3)

or:

$$R = \text{calculated with Einstein formula}$$

for hydraulic radius

and for the Chezy roughness:

$$C = \text{Chezy value given by the user}$$  \hspace{1cm} (3.4.5)

or:

$$C = 18 \log(12 \cdot a / K)$$  \hspace{1cm} (3.4.6)

$K$ = Nikuradse roughness given by the user

or:

$$C = \text{calculated with roughness predictor}$$  \hspace{1cm} (3.4.7)

Some special attention will be given to the implemented roughness predictor. This is the roughness predictor developed by Van Rijn (1982). In ODIRMO the predictor can be used in two ways:

1) with the waterdepth as hydraulic radius
2) with the Einstein predictor for the hydraulic radius
In case 1) the calculation of the Chezy roughness is as follows:

1) a dimensionless grain size is calculated:
\[ D_g = \frac{D_{50}}{50} \times \left( \frac{16.187}{v^2} \right)^{(1/3)} \quad (3.4.8) \]

2) the critical shear velocity according to Shields is computed:
\[ 0 < D_g < 4 \text{ then:} \]
\[ u_{cr}^2 = 16.187 \times D_{50} \times 0.24 / D \quad (3.4.9) \]
\[ 4 < D_g < 10 \text{ then:} \]
\[ u_{cr}^2 = 16.187 \times D_{50} \times 0.14 \times D_g^{-0.64} \quad (3.4.10) \]
\[ 10 < D_g < 20 \text{ then:} \]
\[ u_{cr}^2 = 16.187 \times D_{50} \times 0.04 \times D_g^{-0.10} \quad (3.4.11) \]
\[ 20 < D_g < 150 \text{ then:} \]
\[ u_{cr}^2 = 16.187 \times D_{50} \times 0.013 \times D_g^{0.29} \quad (3.4.12) \]
\[ D_g > 150 \text{ then:} \]
\[ u_{cr}^2 = 0.055 \quad (3.4.13) \]

3) The Chezy value related to the grain size is computed:
\[ C = 18 \log(12 a / 3D_{90}) \quad (3.4.14) \]

4) The shear velocity related with this Chezy value is computed:
\[ u_s = \frac{g \times u_{cr}^2}{C^2} \quad (3.4.15) \]

5) A dimensionless transport parameter is calculated:
\[ T = \left( \frac{u_s^2 - u_{cr}^2}{u_{cr}^2} \right) \quad (3.4.16) \]

6) Compute bed form height:
\[ H = 0.11 \times (D_{50} / a)^{(0.3)} \times (1 - e^{(-0.5T)}) \times (25 - T) \quad (3.4.17) \]

7) Compute bed form length:
\[ L = 7.3 \times a \quad (3.4.18) \]

8) Compute roughness:
Part III: Functional and Technical Design

\[ k_s = 3D_{w0} + 1.1 H (1 - e^{(-25 H/L)}) \]  (3.4.19)

9) Compute Chezy value:

\[ C = 18 \log \left( 12 \frac{a}{K_s} \right) \]  (3.4.20)

This Chezy value is used to calculate \( F \).

In the second case (with Einstein predictor for the hydraulic radius) there are some difficulties. The Chezy roughness related to the grain size (eq. (3.4.14)) can not be computed, as \( a \) has to be replaced with \( R_b \). This \( R_b \) is iteratively calculated with Einstein's method. A first guess of \( R_w \) is taken:

\[ R_w = \frac{a}{8} \]  (3.4.21)

then:

\[ R_b = a \times (1 - 2 \times \frac{R_w}{w}) \]  (3.4.22)

\[ i = u^2 / (C^2 \times R_b) \]  (3.4.23)

\[ \delta_w = 11.6 \sqrt{v / (g R_w i)} \hat{-} 0.5 \]  (3.4.24)

\[ C_w = 18 \log \left( 12 \frac{R_w}{(K_w + 0.3 \delta_w)} \right) \]  (3.4.25)

In the last equation \( K_w \) is the wall roughness given by the user. With the calculated \( R_b \) (3.4.22), the calculated \( C_w \) (3.4.25) and the first guess of \( C \), a new hydraulic radius related to the wall calculated:

\[ R_w^{p1} = R_b \times (C / C_w) \times \hat{2} \]  (3.4.26)

With \( R_w^{p1} \) subsequently a new \( R_b \) (with 3.4.22), a new slope (3.4.23), a new \( \delta_w \) (3.4.24) and \( C_w \) (3.4.25) are calculated and with the now found \( R_b \) and \( C_w \) again a new hydraulic radius related to the wall is calculated:

\[ R_w^{p2} = R_b \times (C / C_w) \times \hat{2} \]  (3.4.27)

Now is checked if the original value \( R_w \) minus \( R_w^{p2} \) is smaller than a given number \( (\varepsilon_R) \). If not a new initial guess \( R_w \) is taken:

\[ R_w = \frac{(R_w^{p1} + R_w^{p2})}{2} \]  (3.4.28)

and the procedure from eq. (3.4.22) until eq. (3.4.27) is repeated until the condition is satisfied. If this is the case the wanted \( R_b \) can now be calculated with the last iteration of \( R_w \) with eq. (3.4.22). Now the Chezy value related to the grain size can be computed with:
All problems are not yet eliminated though. In eq. (3.4.23) to compute the slope, already a Chezy value is needed. This means, that the user should give an initial guess for the Chezy value, which can be used in eq. (3.4.23). Resumed the procedure to calculate the Chezy value according to Van Rijn is in this case:

1) initial guess $C$ (given by the user)  
2) initial guess $R_w$ ($a/8$ is taken)  
3) compute $R_b$ iteratively with the mentioned procedure  
4) compute $C$ related to grain size eq. (3.4.29)  
5) compute $C_p$ with the Van Rijn procedure  
6) use $C_p^1$ as new guess  
7) compute $R_b$ again iteratively  
8) compute a new $C$ related to grain size  
9) compute $C_p^2$ with Van Rijn  
10) Check if $C_p^2$ minus the initial guess is smaller than a given number ($\varepsilon$). If this is the case, the Chezy value is found, if not, take the found Chezy value as a new initial guess and go back to step 3.
4. Calculation of bedlevel

4.1 Equations

For the sediment transport and the bed level there are two equations, an equation for the motion of sediment (4.1.1) and one for continuity (4.1.2).

For the first one a transport formula is used, so that the transport depends on the local conditions only:

\[ s = f(u, D_{50}, \rho_s, \rho_w, v, C, g, ...) \]  (4.1.1)

in which:

- \( s \) = transport
- \( u \) = flow velocity
- \( D \) = grain size
- \( \rho_s \) = grain density
- \( \rho_w \) = water density
- \( v \) = kinematic viscosity
- \( C \) = Chezy value
- \( g \) = gravity

There are several known transport formulae developed. Some of them are treated in par. 4.4.

For the equation for continuity consider a river section (fig. 4.1.1)

\[ s \cdot w - (s + \frac{\partial s}{\partial x} \Delta x) (w + \frac{\partial w}{\partial x} \Delta x) = \]  (4.1.2)

\[ \frac{(w + w + (\partial w/\partial x) \Delta x) \Delta x}{2} (z + (\partial z/\partial t) \Delta t - z) \]
Neglecting the terms with $\Delta x^2$ this can be rewritten as:

$$\frac{\partial w_s}{\partial x} + w \frac{\partial z}{\partial t} + wY = 0$$

(4.1.3)

In the last equation $Y$ is introduced as an external sediment source.

Eq. (4.1.3) is used in DDIRMO to compute the new bedlevel for each timestep. The numerical solution is treated in the next paragraph.
4.2 Numerical solution

4.2.1 'Inside' scheme

There are many finite difference methods to obtain a numerical solution for eq. (4.1.3). Some of those methods are compared with respect to accuracy, stability, damping of secondary waves and numerical diffusion by Vreugdenhil (1981) and Olesen (1981). It appears that no method is really superior on every aspect.

A relatively good and straightforward scheme was first used by Olesen (1981): a predictor corrector method with an explicit predictor and an implicit corrector (Crank - Nicholson scheme) (fig. 4.2.1.1).

![fig. 4.2.1.1 Numerical scheme](image)

Predictor:

\[
\frac{w_i (z_i^{''} - z_i^{'}) + (S_{i+1}^{'} - S_{i-1}^{'}) + w_i Y_i}{\Delta t} + \frac{2w_i}{2\Delta x} = 0
\]  

(4.2.1.1)

or, rewritten:

\[
z_i^{'''} = z_i^{'} - \frac{\Delta t}{2w_i \Delta x} \left( S_{i+1}^{'} - S_{i-1}^{'} \right) - \Delta t Y_i
\]  

(4.2.1.2)

And the corrector:

\[
\frac{w_i (z_i^{''} - z_i^{'}) + \theta (S_{i+1}^{'} - S_{i-1}^{'}) + (1-\theta) (S_{i+1}^{'} - S_{i-1}^{'}) + w_i Y_i}{\Delta t} + \frac{2\Delta x}{2\Delta x} = 0
\]  

(4.2.1.3)

or, rewritten:

\[
z_i^{'''} = z_i^{'} - \frac{\Delta t}{w_i 2\Delta x} \left( \theta (S_{i+1}^{'} - S_{i-1}^{'}) + (1-\theta) (S_{i+1}^{'} - S_{i-1}^{'}) \right) - \Delta t Y_i
\]  

(4.2.1.4)

In which \( \theta \) is a numerical weight factor \( \theta > 0.5 \) for
stability).
In the following part, some of the conclusions of Olesen are summarized. He performed a thorough analysis of this predictor corrector method.

There are two methods of analysing the stability of a numerical method. For the PC scheme this results in the following criteria.

When the PC method is applied for a simple wave equation (eq. (4.2.1.5)), the so called modified equation can be determined. This is the equation that is really solved with the PC method (eq. (4.2.1.6)).

simple wave:
\[
\frac{\partial z}{\partial t} + c \frac{\partial z}{\partial x} = 0 \tag{4.2.1.5}
\]
In which \( c \) is the propagation velocity of the sediment wave.

modified equation:
\[
\frac{\partial z}{\partial t} + c \frac{\partial z}{\partial x} - \frac{dx^2}{2 \Delta t} \frac{\partial^2 z}{\partial x^2} - \frac{dx^3}{6 \Delta t} \frac{\partial^3 z}{\partial x^3} + \ldots = 0 \tag{4.2.1.6}
\]
In this equation \( \lambda_2 \) is the numerical diffusion coefficient. Olesen found:
\[
\lambda_2 = (2 \theta - 1) \sigma^2 \tag{4.2.1.7}
\]
in case of one corrector step. In this equation:
\[
\sigma = \frac{c \Delta t}{\Delta x} \tag{4.2.1.8}
\]
the Courant number.
The numerical diffusion coefficient has to be positive, otherwise an exponential growing solution would be the result. So the demand \( \theta > 0.5 \) is found.

The second method is to examine the complex propagation factor. A solution of eq. (4.2.1.5) is assumed with the form:
\[
z_i' = \rho' \exp(\imath j \xi) \tag{4.2.1.9}
\]
In which:
\( \rho \) = the complex propagation factor.
\( \xi = k \Delta x \)
\( k = 2 \pi / L \)
\( L = \text{wavelength} \)
so:
\[ \rho = \frac{z_i^{t+1}}{z_i} \quad (4.2.1.10) \]

For the PC method with three iterations can be shown:
\[ \rho = 1 - \sigma^2 \sin^2 \xi - i(\sigma \sin \xi \cdot \theta^2 \sigma^2 \sin^2 \xi) \quad (4.2.1.11) \]

The demand for stability is:
\[ |\rho| \leq 1 \quad (4.2.1.12) \]

This gives:
\[ \sigma^2 \leq \frac{1 + (1 + 4(2\theta - 1))^2}{2 \theta^2} \quad (4.2.1.13) \]

In ODIRMO is chosen for \( \theta = 0.65 \), this gives for the Courant number:
\[ \sigma \leq 1.71 \quad (4.2.1.14) \]

Because the analysis is (and can be) only performed for the simple wave equation, no definite conclusions can be drawn for a real case.

accuracy

For the simple wave equation some remarks can be made about the accuracy of the PC method.

The damping factor per wave period:
\[ d = |\rho|^{\eta_i} \quad (4.2.1.15) \]

in which \( \eta_i \) is the number of timesteps in one wave period. And the relative propagation velocity:
\[ c_r = \frac{n_i \arg(\rho)}{-2\pi} \quad (4.2.1.16) \]

For the PC method with three iterations can be shown:
\[ d = 1 - (2\theta - 1) \pi \sigma \xi \quad (4.2.1.17) \]

and:
\[ c_r = 1 - \xi^2 \left(1 + 2\sigma^2(1 - 3\theta + 3 \theta^2)\right) / 6 \quad (4.2.1.18) \]

or, with \( \theta = 0.65 \):
\[ d = 1 - 0.3 \pi \sigma \xi \quad (4.2.1.19) \]

\[ c_r = 1 - \xi^2 \left(1 + 0.64 \sigma^2\right) / 6 \quad (4.2.1.20) \]
With these expressions a rough indication of the numerical solution can be obtained. To get a better impression of the accuracy, however, the only way is to perform the same calculations with different time- and placesteps.

4.2.2 Boundary schemes

The numerical scheme treated in par. 4.2.2 is not applicable at boundaries. Therefore an adapted scheme is needed at the boundaries. Olesen used a predictor (upstream) corrector (four point) scheme at the downstream boundary and an extra point at the upstream boundary where the sediment transport is known. For ODIRMO a new upstream and downstream scheme are developed. This is done, because the Olesen solution (mainly the upstream solution) is not very useful at internal boundaries.

The developed schemes are:

\[ z_{n+1}^l = z_n^l - \frac{\Delta t}{w_n \Delta x} (S_n^l - S_{n-1}^l) - \Delta t Y_n \]  \hspace{1cm} (4.2.2.1)

\[ z_{n+1}^l = z_n^l - \frac{\Delta t}{w_n \Delta x} (\theta (S_n^l - S_{n-1}^l) + (1-\theta) (S_n^l - S_{n-1}^l)) - \Delta t Y_n \]  \hspace{1cm} (4.2.1.4)

- 19 -
Part III: Functional and Technical Design

Predictor upstream boundary:

\[ z_0^{1+1} = z_0^1 - \frac{\Delta t}{w_0} \left( S_1^1 - S_0^1 \right) - \Delta t Y_0 \] (4.2.2.1)

And the corrector:

\[ z_0^{1+1} = z_0^1 - \frac{\Delta t}{w_0} \left( \theta \left( S_1^{1+1} - S_0^{1+1} \right) + (1-\theta) \left( S_1^1 - S_0^1 \right) \right) - \Delta t Y_0 \] (4.2.1.4)

In appendix A the modified equation for the upstream and downstream scheme are derived. It appears that both schemes have a negative numerical diffusion coefficient. This seems to cause no problems in this case as they are only used at the boundaries.

4.3 Test of numerical scheme

To examine the numerical scheme in combination with the new boundary schemes, some test calculations are carried out with computer programs written in BBC Basic on a BBC microcomputer. There are two slightly different alternatives tested: one with \( z \) and \( s \) calculated at every point and one with alternately \( s \) or \( z \) computed (staggered grid) (fig. 4.3.1).

At first sight, the staggered grid seems to cost a little less calculation time, but as \( s \) has to be calculated between points of known \( z \), an interpolation has to be carried out to calculate the water flow velocity.

In the test programs a flume is considered with horizontal uniform flow. The flume has a length of 4 metres. The used transport formula is:

\[ s = w^h \] (4.3.1)

with:

- \( m = 5 \times 10^{-5} \)
- \( n = 5 \)

(the unit of width is taken)

In fig. 4.3.2 the situation is given.
It is easy to see that the bed level is in equilibrium when the sediment input at the upstream boundary is $5 \times 10^{-5} \text{ m}^2/\text{s}$. First consider what happens when the sediment input is more than $5 \times 10^{-5}$, say $8 \times 10^{-5} \text{ m}^2/\text{s}$: a sedimentation wave will begin to propagate in the flume. In the following pages the situation after 8000 seconds will be shown with varying $\theta$, $\Delta t$ and Courant number. The Courant number is in this case:

$$\sigma = \frac{c \Delta t \approx n \Delta t}{\Delta x} = \frac{5 \times 5 \times 10^{-5} \Delta t}{\Delta x}$$  \hspace{1cm} (4.3.2)

The number of iterations is three (one predictor and two corrections), as Olesen stated that this gives the greatest region of stability.

In the results the old and new equilibrium are indicated with a dotted line. The theoretical position of the sedimentation wave is also indicated (simple wave approach).
fig. 4.3.3 Sedimentation wave
left non staggered grid
gleich staggered grid
above = 0.7
below = 0.6
Part III: Functional and Technical Design

\[ \Delta X = 0.700 \]

<table>
<thead>
<tr>
<th>Z (m)</th>
<th>( \text{BODEM} )</th>
<th>( \text{COURANT} )</th>
<th>( \text{BODEM} )</th>
<th>( \text{COURANT} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.050</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.100</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.150</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.200</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.250</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.300</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.350</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.400</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.450</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.500</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.550</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.600</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.650</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.700</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.750</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.800</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.850</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.900</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>0.950</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
<tr>
<td>1.000</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
<td>0.879</td>
</tr>
</tbody>
</table>

\[ \Delta T = 0.400 \]

**fig. 4.3.4 Sedimentation wave**
- left non staggered grid
- right staggered grid

- 23 -
Part III: Functional and Technical Design

\[
\begin{align*}
\Delta X &= 0.4 \\
\Delta T &= 400 \\
T_{\text{MAX}} &= 8000 \\
N_A &= 8000 \\
\text{SEC IS DE SITUATIE ALS VOLGT:} \\
\text{BODEM} &\quad \text{COURANT} \\
Z(0.000) &= 1.092 \quad 0.440 \\
Z(0.200) &= 1.084 \quad 0.434 \\
Z(0.400) &= 1.077 \quad 0.430 \\
Z(0.600) &= 1.069 \quad 0.425 \\
Z(0.800) &= 1.061 \quad 0.420 \\
Z(1.000) &= 1.053 \quad 0.415 \\
Z(1.200) &= 1.046 \quad 0.410 \\
Z(1.400) &= 1.039 \quad 0.405 \\
Z(1.600) &= 1.032 \quad 0.400 \\
Z(1.800) &= 1.025 \quad 0.395 \\
Z(2.000) &= 1.018 \quad 0.390 \\
Z(2.200) &= 1.011 \quad 0.385 \\
Z(2.400) &= 1.004 \quad 0.380 \\
Z(2.600) &= 0.997 \quad 0.375 \\
Z(2.800) &= 0.990 \quad 0.370 \\
Z(3.000) &= 0.983 \quad 0.365 \\
Z(3.200) &= 0.976 \quad 0.360 \\
Z(3.400) &= 0.969 \quad 0.355 \\
Z(3.600) &= 0.962 \quad 0.350 \\
Z(3.800) &= 0.955 \quad 0.345 \\
Z(4.000) &= 0.948 \quad 0.340 \\
TETA &= 0.700
\end{align*}
\]

\[
\begin{align*}
\Delta X &= 0.2 \\
\Delta T &= 400 \\
T_{\text{MAX}} &= 8000 \\
N_A &= 8000 \\
\text{SEC IS DE SITUATIE ALS VOLGT:} \\
\text{BODEM} &\quad \text{COURANT} \\
Z(0.000) &= 0.894 \quad 0.771 \\
Z(0.200) &= 0.887 \quad 0.768 \\
Z(0.400) &= 0.880 \quad 0.766 \\
Z(0.600) &= 0.872 \quad 0.764 \\
Z(0.800) &= 0.865 \quad 0.762 \\
Z(1.000) &= 0.858 \quad 0.760 \\
Z(1.200) &= 0.851 \quad 0.758 \\
Z(1.400) &= 0.844 \quad 0.756 \\
Z(1.600) &= 0.837 \quad 0.754 \\
Z(1.800) &= 0.830 \quad 0.752 \\
Z(2.000) &= 0.823 \quad 0.750 \\
Z(2.200) &= 0.816 \quad 0.748 \\
Z(2.400) &= 0.809 \quad 0.746 \\
Z(2.600) &= 0.802 \quad 0.744 \\
Z(2.800) &= 0.795 \quad 0.742 \\
Z(3.000) &= 0.788 \quad 0.740 \\
Z(3.200) &= 0.781 \quad 0.738 \\
Z(3.400) &= 0.774 \quad 0.736 \\
Z(3.600) &= 0.767 \quad 0.734 \\
Z(3.800) &= 0.760 \quad 0.732 \\
Z(4.000) &= 0.753 \quad 0.730 \\
TETA &= 0.700
\end{align*}
\]

Fig. 4.3.5
left non staggered grid
right staggered grid
above sedimentation wave small Courant number
below erosion wave
Examining the results for the staggered and the non-staggered grid is concluded that there is not much difference in the results for both schemes. As there is little to be gained using the staggered grid in this case, the staggered grid is abandoned and from now on, only the non-staggered scheme is discussed.

The results of the test are not surprising. With $\theta = 0.6$ the results tend to have a higher accuracy, as $\theta = 0.7$ smooths the propagating wave more (fig. 4.3.3). Smaller timesteps (but constant Courant number, so also smaller placesteps) increase the accuracy (fig. 4.3.4). Taking the Courant number much smaller than one decreases the accuracy considerably (fig. 4.3.5 together with a erosion wave).

Next is examined, what happens when the sedimentation reaches the downstream boundary (fig. 4.3.6). It appears, that there is a small wave reflected at the downstream boundary. This phenomenon is not alarming, as the reflected waves are really quite small.

As is stated before, the new equilibrium is indicated by a dotted line in the plots. This new equilibrium can be computed as follows:

$$s = m \frac{u^n}{h} = 5 \times 10^{-5} \left( \frac{q}{h-z} \right) = 8 \times 10^{-5}$$

As $q$ and $h$ are constant in this case, the new $z$ equilibrium is the only unknown variable in the equation. It appears: $z_{\text{new}} = 1.090$ m. The last test (last part of fig. 4.3.5) has an erosion wave with sediment input $3 \times 10^{-5}$ m$^2$/s. This gives $z_{\text{new}} = 0.893$ m.

These facts are stated because it can be seen now, that the existing original waterdepth is decreased resp. increased by about 10%. The next step is to test what happens when a much higher sedimentation wave is generated at the upstream boundary (fig. 4.3.7). The results shown are for a sediment input of $20 \times 10^{-5}$ m$^2$/s. This means $z_{\text{new}} = 1.24$ m, so a reduction of the waterdepth by 25%. It is stated, that the secondary waves following the sedimentation front are more pronounced now. This is not surprising, as the wave front itself has an unstable character. This unstable character is caused by the propagation velocity, which is higher just after the front than just before the front, so the sedimentation wave tends to break, just like an ocean wave near the coast. This breaking effect is not represented in this model. However, the results are not desastrous, and it is seen that the wave 'leaves' the flume neatly.

Finally in fig. 4.3.8 the results are shown for calculations with a (too) high Courant number. With eq. (4.2.1.13) the
theoretical maximum for the Courant number is (with \( \theta = 0.7 \)) 1.63. In the test cases it appears that a Courant number of 1.7 still causes no instability (but the accuracy is rather bad). A Courant number of 1.9 causes instability. It would be advisable to avoid instability in the calculation at all times. With the remarks on the accuracy in mind the best advice is to keep the Courant number close to unity.
### Part III: Functional and Technical Design

#### Nl I Versierzangen Schema

**DELTA x = 0.2**

**DELTA T = 400**

**TMAX = 12000**

NA 12000 SEC IS DE SITUATIE ALS VOLGT:

<table>
<thead>
<tr>
<th>BODEM</th>
<th>COURANT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z(0.000) = 1.090</td>
<td>0.879</td>
</tr>
<tr>
<td>Z(0.200) = 1.090</td>
<td>0.879</td>
</tr>
<tr>
<td>Z(0.400) = 1.090</td>
<td>0.879</td>
</tr>
<tr>
<td>Z(0.600) = 1.090</td>
<td>0.879</td>
</tr>
<tr>
<td>Z(0.800) = 1.090</td>
<td>0.879</td>
</tr>
<tr>
<td>Z(1.000) = 1.090</td>
<td>0.879</td>
</tr>
<tr>
<td>Z(1.200) = 1.090</td>
<td>0.879</td>
</tr>
<tr>
<td>Z(1.400) = 1.090</td>
<td>0.879</td>
</tr>
<tr>
<td>Z(1.600) = 1.090</td>
<td>0.879</td>
</tr>
<tr>
<td>Z(1.800) = 1.090</td>
<td>0.879</td>
</tr>
<tr>
<td>Z(2.000) = 1.090</td>
<td>0.879</td>
</tr>
<tr>
<td>Z(2.200) = 1.091</td>
<td>0.878</td>
</tr>
<tr>
<td>Z(2.400) = 1.089</td>
<td>0.876</td>
</tr>
<tr>
<td>Z(2.600) = 1.090</td>
<td>0.879</td>
</tr>
<tr>
<td>Z(2.800) = 1.091</td>
<td>0.887</td>
</tr>
<tr>
<td>Z(3.000) = 1.092</td>
<td>0.887</td>
</tr>
<tr>
<td>Z(3.200) = 1.094</td>
<td>0.896</td>
</tr>
<tr>
<td>Z(3.400) = 1.095</td>
<td>0.914</td>
</tr>
<tr>
<td>Z(3.600) = 1.090</td>
<td>0.896</td>
</tr>
<tr>
<td>Z(3.800) = 1.063</td>
<td>0.746</td>
</tr>
<tr>
<td>Z(4.000) = 1.040</td>
<td>0.640</td>
</tr>
</tbody>
</table>

**TETA = 0.700**

---

**fig. 4.3.6 Sedimentation wave leaving flume**

**fig. 4.3.7 Large overload wave**

---

- 27 -
Part III: Functional and Technical Design

---

**Fig. 4.2.10**

**Fig. 4.3.8 Sedimentation wave**

*left: Courant number 1.7*

*right: Courant number 1.9*
4.4 Various transportformulae

In ODIRMO four transportformulae for the sediment are implemented:

Meyer-Peter & Mueller (1948)

Engelund & Hansen (1967)

Ackers & White (1973)

Power law

They are briefly described in this paragraph. For more information is referred to the original publications.

Meyer-Peter & Mueller

The sediment transport is calculated with:

\[ s = 13.3 \times (g \Delta D_{50}^{3})^{(1/2)} \times (\mu T - 0.047)^{(1.5)} \]  

(4.4.1)

in which:

\[ T = u^2 / (\Delta \times D_{50} \times C^2) \]  

(4.4.2)

\[ \mu = (C / C_{gr})^{(1.5)} \]  

(4.4.3)

with:

\[ C_{gr} = 18 \log(12R / (D_{90} + 0.38)) \]  

(4.4.4)

and:

\[ \delta = 11.6 \times C / (u \times g^{(1/2)}) \]  

(4.4.5)

R in eq. (4.4.4) is calculated when the friction term in the equation for the water motion is determined. The value of R can be equal to the water depth or the hydraulic radius.

In ODIRMO both constants in the formula (13.3 and 0.047) are not resident in the program. They should be given by the user. This is done, because a user may want to adapt the formula for his own situation.
Engelund & Hansen

The sediment transport is calculated with:

\[ s = 0.084 \times m \times (u/n)^2 \times (u'/n)^3 \]  \hspace{1cm} (4.4.6)

in which:

\[ n = (g \Delta D_{50})^{(1/2)} \]  \hspace{1cm} (4.4.7)
\[ m = n \times D_{50} \]  \hspace{1cm} (4.4.8)
\[ u' = g^{(1/2)} \times u / C \]  \hspace{1cm} (4.4.9)

In ODIRMO the user may give a factor by which the transport according to this formula can be multiplied. With this possibility one can adapt the calculations to experimental data.

Ackers & White

The transport according to this formula can be written as:

\[ s = C_{gr} \times u \times D_{35} \times (u/u)^n \times (F_{gr} - 1)^m \times A \]  \hspace{1cm} (4.4.10)

in which:

\[ F_{gr} = u \times (g \Delta D_{35})^{(-1/2)} \times (u / u)^n \]
\[ \times (32^{(1/2)} \times 10 \log(10a/D))^{(n-1)} \]  \hspace{1cm} (4.4.11)
\[ A = 0.23/D_{gr}^{(1/2)} + 0.14 \]  \hspace{1cm} (4.4.12)
\[ m = 9.66/D_{gr} + 1.34 \]  \hspace{1cm} (4.4.13)
\[ n = 1 - 0.56 \log D_{gr} \]  \hspace{1cm} (4.4.14)
\[ \log C_{gr} = 2.86 \log D_{gr} - (10 \log D_{gr})^2 - 3.53 \]  \hspace{1cm} (4.4.15)

In which \( D_{gr} \) is a dimensionless grain diameter given by:

\[ D_{gr} = D_{35} \times (g \Delta/v^2)^{(1/3)} \]  \hspace{1cm} (4.4.16)

In case \( D_{gr} > 60 \) (this is hardly ever the case):

\[ A = 0.17; \ n = 0; \ m = 1.5; \ C_{gr} = 0.025 \]  \hspace{1cm} (4.4.17)

In ODIRMO \( D_{50} \) is used in stead of \( D_{35} \). This is done to save memory space. If \( D_{35} \) and \( D_{50} \) are both needed this presents a problem. An array containing \( D_{35} \) should than be added. For \( \epsilon \) a value of 0.4 is assumed.
As with Engelund & Hansen, the user may give a multiplication factor for the formula.

**Power law**

This is really the most simple transport formula one can imagine:

\[ s = m \cdot u^n \]  \hspace{1cm} (4.4.18)

In which \( m \) and \( n \) are constants given by the user. A big advantage of this formula is, that it is easy to handle. It is easy to calculate the Courant number and to check results of the transport calculation.

It is remarked, that the Engelund & Hansen formula has the same basic form as the power law. When the grain size and the Chezy roughness are constant, both formula are identical. In that case:

\[ n = 5 \]  \hspace{1cm} (4.4.19)

\[ m = 0.084 \cdot g^{(3/2)} \]  \hspace{1cm} (4.4.20)

\[ \frac{(g \Delta)^2 \cdot C^3 \cdot D_{50}}{(g \Delta)^2 \cdot C^3 \cdot D_{50}} \]
Appendix A

Modified equation for the upstream and downstream schemes.

Upstream:

1. Predictor:

\[ \frac{z_1 - z_0}{\Delta t} + c \left( \frac{z_1 - z_0}{\Delta x} \right) = 0 \]  
\hspace{1cm} (A.1)

2. Predictor

\[ \frac{z_1 - z_1}{\Delta t} + a \left( \frac{z_0 - z_0}{2 \Delta x} \right) = 0 \]  
\hspace{1cm} (A.2)

3. Corrector

\[ \frac{z_0 - z_0}{\Delta t} + c \left( \frac{1}{2} \right) \left( z_1 - z_0 \right) + (1 - \frac{1}{2}) \left( z_0 - z_0 \right) = 0 \]  
\hspace{1cm} (A.3)

Taking (A.2) - (A.1) we get:

\[ \frac{z_1 - z_0}{\Delta x} = \frac{z_0 - z_0}{\Delta x} - c \Delta t \left( \frac{1}{2} z_0 - z_1 + \frac{1}{2} z_0 \right) \]  
\hspace{1cm} (A.4)

Combining (A.4) with (A.3):

\[ \frac{z_0 - z_0}{\Delta t} + c \left( \frac{-1}{2} \right) \left( \frac{1}{2} \left( \frac{z_0 - 2 z_0 + z_0}{\Delta x} \right) + \frac{z_0 - z_0}{\Delta x} \right) = 0 \]  
\hspace{1cm} (A.5)

\[ \begin{array}{ccc}
I & II & III \\
\frac{z_0 - z_0}{\Delta t} + c \left( \frac{-1}{2} \right) \left( \frac{z_0 - 2 z_0 + z_0}{\Delta x} \right) + \left( \frac{z_0 - z_0}{\Delta x} \right) & = 0 \\
\end{array} \]  

\[ I := \frac{\partial z}{\partial t} + c^2 \frac{\Delta t}{2} \frac{\partial^2 z}{\partial x^2} \]  
\hspace{1cm} (A.6)  

\[ III := \frac{\partial^2 z}{\partial x^2} + 0 \left( \Delta x^3 \right) \]  
\hspace{1cm} (A.7)

So we find for the modified equation:

\[ \frac{\partial z}{\partial t} + c \frac{\partial z}{\partial x} = \frac{\Delta x^2}{2 \Delta t} \left( \sigma \left( \frac{-1 + (\Delta /\Delta) \sigma}{\Delta x^2} \right) \frac{\partial^2 z}{\partial x^2} \right) \]  
\hspace{1cm} (A.9)

\[ \lambda_2 < 0 \Rightarrow \text{instability} \]

For the downstream boundary scheme an analogous \( \lambda_2 \) is found.
Appendix B
Test programs

On the next pages the test programs for the predictor corrector method are given (see chapter 4.3). They are written in BBC-basic. Some explanation is given below.

Program for staggered grid:

The predictor corrector loop starts at line 290 and ends at line 530. At line 330 to 360 the transport is calculated with the interpolated waterdepth. At line 410 S is prepared for the predictor step. Further:
- line 430: upstream scheme
- line 440-460: inside scheme
- line 480: downstream scheme
- line 500-520: calculation of new waterdepth
- line 540-560: reset z for new timestep

Program for normal grid:

The predictor corrector loop starts at line 260 and ends at line 510. At line 330 to 360 the transport is calculated with the interpolated waterdepth. At line 390 S is prepared for the predictor step. Further:
- line 410: upstream scheme
- line 420-440: inside scheme
- line 460: downstream scheme
- line 480-500: calculation of new waterdepth
- line 520-540: reset z for new timestep

Used variables in this program:

- L: length of flume
- M: waterlevel
- Q: discharge
- ZB: initial bedlevel
- F: factor in S=F.u^n
- TETA: numerical weight factor
- DELTAX: placestep
- DELTAT: timestep
- JDIM: L/DELTAX
- Z(0..1, 0..JDIM): bedlevel
- S(0..1, 0..JDIM): sediment transport
- A(0..JDIM): waterdepth
- u: flow velocity
- ITERNR: counter for pre-co iteration
10 REM NUMERIEK PROBEERPROGRAMMA
20 REM VERSPRONGEN SCHEMA
30 *KEY 0 MODE4:MDU4:MRUN:1
40 *KEY 1 MODE2:MDU4:ML.:M
50 CLS
60 REM inlezen LENGTE, WATERNIVO, DEBIET, BEGINNIVO BODEM, F uit S=F*U*S en T
70 REM cranck nicholson schema
80 INPUTTABO,101 "DELTA X = "DELTAX
90 INPUT "DELTA T = "DELTAT
100 COURANT=DELTAT/DELTAX
110 REM JDIM 1S het aantal stappen DELTAX in de goot
120 JDIM=L/DELTAX
130 REM bereken beg1nwaterdiepte voor crank nicholson schema
140 FOR 1%=0 TO JDIM
150 AI1%1=H-Z10,1%1
160 NEXT
170 REM lees randvoorwaarde transport
180 READ S10,01
190 S10,01=S11,01
200 REPEAT
210 INPUT "TNAX "TMAX
220 @%=10
230 FOR 1%=0 TO JDIM+1: SII,01=S10,01: NEXT
240 REM bereken nieuwe waterdiepte voor corrector stappen
250 FOR 1%=0 TO JDIM
260 AI1%1=H-Z11,1%1
270 NEXT
280 UNTIL ITERNR%=3
290 FOR 1%=0 TO JDIM
300 Z11,1%1=Z10,1%1
310 NEXT
320 TIJD=TIJD+DELTAT
330 UNTIL TIJD+DELTAT>TMAX
340 CLS
350 PRINT"NA ";TIJD;" SEC IS DE SITUATIE ALS VOLGT ":PRINT
360 FOR 1%=0 TO JDIM
370 PRINT Z(1,1%):COURANT*S(K%,1%)*S/A(1%)
380 NEXT
390 PRINT:PRINT "druk op spatiebalk voor plot"
400 REPEAT UNTIL GET=32
410 MOVEO,500+(Z(I,0)-1*20000)
420 CLS
430 FOR 1%=1 TO JDIN
440 DRAW(1200/LI*IX*DELTAX,500+(Z(I,1%)-1*20000)
450 NEXT
460 PRINT "de situatie n;o ";TIJD;" seconden:"
470 REPEAT UNTIL WENS="J" OR WENS="N"
480 END
VOU3

L.L.

10 REM NUMERIEK PROBEERPROMMA
20 REM NIET VERSPRONGEN SCHEMA
30 *KEY 0 MODE4:VDU14:MRUN=I
40 *KEY 1 MODE3:VDU14:PLM=I
50 CLS
60 REM lees inlenen LENGTE, WATERNIVO, DEBIET, BEGINNIVO BODEM, F uit S=F*U^5 en T

ETA voor crank nicholson schema
70 READ L,H,0,1,B,TETA
80 INPUTTABIO,IOI "DELTA X = "DELTAX
90 INPUT "DELTA T = "DELTAT
100 COURANT=DELTAT/DELTAX
110 REM JDIM is het aantal stappen DELTAX in de goot
120 JDIM=L/DELTAX
130 DIM ZII,JDIMI,SII,JDIMI,AIJDIMI
140 REM bereken beginwaterniveau voor alle X
150 FOR 1%=0 TO JDIM
160 ZIO,1Y.I=ZB
170 AI1%=H-ZIO,1Y.I
180 NEXT
190 REM lees randvoorwaarde transport
200 READ SII,OI
210 SII,OI=SII,OI
220 REM bereken beginwaterdiepte voor alle X
230 FOR 1%=0 TO JDIM
240 ZII,1Y.I=ZB
250 NEXT
260 FOR 1%=0 TO JDIM
270 SII,1Y.I=SII,1Y.I
280 NEXT
290 FOR 1%=0 TO JDIM
300 ZII,1Y.I=SII,1Y.I
310 NEXT
320 FOR 1%=0 TO JDIM
330 U=Q/AIIY.I
340 SIK%=F*U%*S
350 NEXT
360 REM bereken nieuwe waterdiepte voor corrector stappen
370 FOR 1%=0 TO JDIM
380 AII%=H-ZII,1Y.I
390 NEXT
400 FOR 1%=0 TO JDIM
410 PRINT"NA ";TIJD;" SEC IS DE SITUATIE ALS VOLGT :"
420 PRINT
430 FOR 1%=0 TO JDIM
440 PRINT"ZI";1%;" = ";ZII,1Y.I*COURANT*SIK%;1%**S/AIIY.I
450 NEXT
460 PRINT:PRINT@d600@"DEBEM
470 FOR 1%=0 TO JDIM
480 FOR 1%=0 TO JDIM
490 PRINT"Wil je verder rekenen? J/N"
500 UNTIL WENSS="J" OR WENSS="N"
510 UNTIL WENSS="N"
520 END
530 REM data: L; H; 0; IB; F; TETA
540 DATA 4,2,1,1,5E-5,.6
550 DATA 8E-5
Part IV : Program Description

Contents

1. General remarks 2
2. CREADATA 3
   2.1 Main Program 4
   2.2 Subroutines
      2.2.1 InitCrea 6
      2.2.2 AskWidth 8
      2.2.3 AskBedlevel 10
      2.2.4 AskDischarge 12
      2.2.5 AskTransport 14
      2.2.6 AskWaterlevel 16
      2.2.7 AskRoughness 18
      2.2.8 AskGrainsize 20
      2.2.9 AskFormula 22
      2.2.10 AskSource 23
      2.2.11 WriteDataToFile 25
      2.2.12 WhatNext 26
3. ODIRMO 28
   3.1 Main Program 30
   3.2 Subroutines
      3.2.1 ReadDataFromFile 32
      3.2.2 DischargeCalc 34
      3.2.3 WaterlevelDownstream 37
      3.2.4.1 BackwaterCalc 39
      3.2.4.1 Function Roughterm 42
      3.2.5 SeditransCalc 44
      3.2.6 SeditransUpstrm 46
      3.2.7 Results 48
      3.2.8 BedlevelCalc 49
      3.2.9 WriteDataToFile 51

Appendices

A. List of most important variables 53
B. Program listing CREADATA 55
C. Program listing ODIRMO 57
1. General Remarks

In this part of the report, the computer programs are treated.
CREADATA is treated in chapter 2. It is a menu-driven program to prepare data input for ODIRMO, which asks the necessary data subsequently from the user.
ODIRMO is treated in chapter 3. It is a program for computations of one dimensional morphological phenomena in rivers. Attention will be paid to the structure of the program and to the use of the system parameters.

Both CREADATA and ODIRMO are written on an IBM Personal Computer in the programming language Pascal (Borland's TURBO Pascal Compiler). Of ODIRMO a version in VS-Pascal has been made. This version is executable on mainframe computers with a VS Pascal Compiler.

It is remarked, that a separate part of this report (Part VI) is written as a User Guide. In that Part attention will also be paid to the possibility to prepare input data on a personal computer, send the data to a mainframe and to perform the calculations there. This option can be important when the calculations on a micro computer take too much time.

In the Appendices the program listings are given. They can give additional information about the use of the system parameters and the structure of the programs.
2. CREADATA

2.1 Main Program

- Purpose:

Call the subroutines subsequently.

- Usage:

The program should be started from the main menu. Choose option 1 from the menu to start CREADATA.

- Remarks:

CREADATA can not be started directly from MS-DOS. This is because CREADATA is resident in a so called Chain-file (CREADATA.CHN) on the run-time diskette. The necessary run-time program is resident in the main menu (file MENU.COM), so this program should be loaded first. See the User Guide (Part VI) for more information and the TURBO Pascal Manual for detailed information on the use of Command-(.COM) and Chain-(.CHN) files.

CREADATA has no special structure. The subroutines are called subsequently in the same order as they are treated in paragraph 2.2.
2.2 Subroutines

2.2.1 InitCrea

Purpose:

Input of some general data of the problem:
- data filename
- number of branches
- maximum time of calculations
- timestep
- length of each branch
- place step in each branch
- number of downstream branch for each branch

Usage:

InitCrea(OutputDataname, nbranch, tOutput, ntstep, tmax,
        deltaT, nstep, dostrm, upstrm, length, deltaX);

Description of parameters:

OutputDataname : name of the datafile to be created
nbranch : number of branches
tOutput : the timesteps after which output is desired
ntstep : number of timesteps
tmax : maximum time of computations
deltaT : timestep
nstep : number of place steps in a branch
dostrm : number of the downstream branch
upstrm : number of the upstream branch(es)
length : length of a branch
deltaX : place step in a branch

All values changed on exit.

Remark:

All data are asked from the user, only ntstep, nstep and upstrm are determined by the computer.
structure InitCrea

enter InputDatename
enter nbranch
enter tmax
enter deltaT
ntstep := round(tmax/deltaT)
enter tOutput

for i := 1 to nbranch
    enter length[i]
    enter deltaX[i]
    nstep[i] := round(length[i]/deltaX[i])
    enter dostrm[i]

for i := 1 to nbranch
    upstrm[dostrm[i],1] := 0
    upstrm[dostrm[i],2] := 0

for i := 1 to nbranch

| T | upstrm[dostrm[i],1] = 0 |
| F |
| i | upstrm[dostrm[i],2] := i |
2.2.2 AskWidth

Purpose:

Input of the width of each riverbranch (mainstream width, floodplain width and sediment transporting width).

Usage:

AskWidth(nstep, deltaX, width1, width2, widthS, sysp);

Description of parameters:

nstep : number of place steps in a branch
unchanged on exit

deltaX : place step in a branch
unchanged on exit

width1 : width of the mainstream
changed on exit

width2 : width of the floodplain
changed on exit

widthS : sediment transporting width
changed on exit

sysp : system parameters
some of them changed on exit

Remark:

The input of the various data is controlled by three systemparameters:

sysp[i, WidthMainstrm] :
"c" : width of the mainstream is constant and is only asked once
"v" : width of the mainstream is variable per grid point and is asked at every grid point

sysp[i, WidthFloodpln] :
"0" : there are no floodplains
"c" : width of the floodplain is constant
"v" : width of floodplain is variable per grid point

sysp[i, WidthTransprt] :
"e" : sediment transporting width is equal to the width of the mainstream at every point
"c" : sediment transporting width is constant
"v" : sediment transporting width is variable per grid point

The value of the systemparameters correspond with the options given in the menu's on the screen.
### Part IV: Program Description

**structure AskWidth**

```plaintext
for i := 1 to nbranch

repeat

create menu for sysp[i, WidthMainstrm]
enter sysp[i, WidthMainstrm]

create menu for sysp[i, WidthFloodpln]
enter sysp[i, WidthFloodpln]

create menu for sysp[i, WidthTransprt]
enter sysp[i, WidthTransprt]

case sysp[i, WidthMainstrm] of

"c" : enter Width1[i, 0]
    for i1 := 1 to nstep[i]
        Width1[i, i1] := Width1[i, 0]

"v" : for i1 := 0 to nstep[i]
        enter Width1[i, i1]

case sysp[i, WidthFloodpln] of

"c" : enter Width2[i, 0]
    for i1 := 1 to nstep[i]
        Width2[i, i1] := Width2[i, i1]

"v" : for i1 := 0 to nstep[i]
        enter Width2[i, i1]

case sysp[i, WidthTransprt] of

"c" : enter WidthS[i, 0]
    for i1 := 1 to nstep[i]
        WidthS[i, i1] := widthS[i, i1]

"v" : for i1 := 0 to nstep[i]
        enter widthS[i, i1]

ask if the input was right for branch(i)

until input right
```
2.2.3 AskBedlevel

Purpose:
Input of the initial bedlevel of the mainstream and the floodplain (if present) of each branch.

Usage:
AskBedlevel(nstep, deltaX, Bedlevel2, Bedlevel1, sysp);

Description of parameters:

nstep : number of place steps in a branch unchanged on exit
deltaX : place step in a branch unchanged on exit
Bedlevel2 : bedlevel of floodplain changed on exit
Bedlevel1 : bedlevel of mainstream changed on exit
sysp : system parameters some of them changed on exit

Remark:
The input of the various data is controlled by three system parameters:

sysp[i,BedlevelMainstrm] :
"e" : bedlevel of the mainstream is asked in every grid point
"o" : bedlevel of the mainstream is asked in one point plus the slope

sysp[i,WidthFloodpln] :
(value already known from AskWidth)
"0" : there are no floodplains (the bedlevel of the floodplain is not asked in that case)
"c",
"v" : bedlevel of the floodplain is asked

sysp[i,BedlevelFloodpln] :
"c" : bedlevel of the floodplain is on a constant distance above the bedlevel of the main stream
"e" : bedlevel of the floodplain is asked in every grid point
"o" : bedlevel of the floodplain is asked in one point plus the slope

The value of the system parameters correspond with the options given in the menu's on the screen.
**structure AskBedlevel**

for $i := 1$ to $n_{\text{branch}}$

repeat

create menu for sysp[$i, \text{BedlevelMainstrm}$]
enter sysp[$i, \text{BedlevelMainstrm}$]

case sysp[$i, \text{WidthFloodpln}$] of

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;c&quot;</td>
<td>create menu for sysp[$i, \text{BedlevelFloodpln}$] enter sysp[$i, \text{BedlevelFloodpln}$]</td>
</tr>
<tr>
<td>&quot;v&quot;</td>
<td>create menu for sysp[$i, \text{BedlevelFloodpln}$] enter sysp[$i, \text{BedlevelFloodpln}$]</td>
</tr>
<tr>
<td>&quot;0&quot;</td>
<td>sysp[$i, \text{BedlevelFloodpln}$] := &quot;0&quot;</td>
</tr>
</tbody>
</table>

case sysp[$i, \text{BedlevelMainstrm}$] of

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
</table>
| "e"   | for $i_1 := 0$ to $n_{\text{step}[i]}$
enter Bedlevel1[$i,i_1,0$] |
| "0"   | enter position enter bedlevel enter slope
for $i_1 := 0$ to $n_{\text{step}[i]}$
Bedlevel1[$i,i_1,0$] := bedlevel+slope*(position-$i_1*\delta X[i]$) |

Ask if the input was right for branch(i)

ask if the input was right for branch(i)

until input right
2.2.4 AskDischarge

Purpose:

Input of the discharge at the upstream boundary and the eventual water withdrawal at internal boundaries.

Usage:

AskDischarge(ntstep, deltaT, upstrm, Qextern, sysp);

Description of parameters:

- ntstep: number of time steps unchanged on exit
- deltaT: time step unchanged on exit
- upstrm: contains the number of the upstream branch(es) unchanged on exit
- Qextern: discharge at the upstream boundary or withdrawal at an internal boundary changed on exit
- sysp: system parameters some of them changed on exit

Remarks:

The input of the various data is controlled by two system parameters:

sysp[i, UpstrmBound]:
- "u": branch has a real upstream boundary
- "i": branch has an internal upstream boundary
- "c": branch has a confluence at the upstream boundary

sysp[i, Discharge]:
- "c": constant discharge at upstream boundary
- "p": periodic discharge curve at upstream boundary
- "f": general discharge curve (for the whole time range)
- "u": discharge equal to the discharge of the upstream branch(es)
- "e": constant water extraction at the upstream boundary

The value of the sysp[i, UpstrmBound] is determined by the program itself with the use of the array upstrm. This array has been filled in the subroutine InitCrea. According to the value of sysp[i, UpstrmBound] the user is asked for an external or an internal boundary condition.
structure AskDischarge

for i := 1 to nbranch
  repeat
  case upstrm[i,1] of
    "O": sysp[i,UpstrmBound] := "u"
      create upstream-menu for sysp[i,Discharge]
    else
      if upstrm[i,2] = 0 then
        sysp[i,UpstrmBound] := "i"
      else
        sysp[i,UpstrmBound] := "c"
      create internal boundary menu for sysp[i,Discharge]
  enter sysp[i,Discharge]
  case sysp[i,UpstrmBound] of
    "u": case sysp[i,Discharge] of
      "c": enter Qextern[i,0]
        for il := 1 to nstep
          Qextern[i,il] := Qextern[i,0]
      "P": enter period
        periodStep := round(period/deltaT)
        for il := 1 to periodStep
          enter Qextern[i,il]
          for il := periodStep to nstep
            Qextern[i,il] := Qextern[i,il-periodStep]
    "f": for il := 0 to nstep
      enter Qextern[i,il]
  "i", "c": case sysp[i,Discharge] of
    "e": enter Qextern[i,0]
      for il := 1 to nstep
        Qextern[i,il] := Qextern[i,0]
    "u": for il := 0 to nstep
      Qextern[i,il] := 0
  ask if the input was right for branch[i]
  until input right
2.2.5 AskTransport

Purpose:

Input of the upstream sediment transport at boundary condition and the eventual sediment withdrawal at internal boundaries.

Usage:

AskTransport(ntstep, deltaT, Sextern, sysp);

Description of parameters:

ntstep : number of time steps unchanged on exit
deltaT : time step unchanged on exit
Sextern : sediment input at the upstream boundary or withdrawal at an internal boundary changed on exit
sysp : system parameters some of them changed on exit

Remarks:

The input of the various data is controlled by two system parameters:

.sysp[i,UpstrmBound] :
"u" : branch has a real upstream boundary
"i" : branch has an internal upstream boundary
"c" : branch has a confluence at the upstream boundary

.sysp[i,Sediment] :
"b" : bedlevel constant at upstream boundary
"c" : constant sediment input at upstream boundary
"f" : sediment input function of time
"t" : transport at upstream boundary calculated with transport formula
"o" : transport at upstream boundary equal to transport at downstream boundary
"u" : sediment input equal to the sediment output of the upstream branch(es)
"e" : constant sediment extraction at the upstream boundary

The value of the sysp[i,UpstrmBound] is already determined in AskDischarge. According to the value of sysp[i,UpstrmBound] the user is asked for an external or an internal boundary condition.
Part IV: Program Description

**structure AskTransport**

```plaintext
for i := 1 to nbranch
    repeat
        case sysp[i, UpstrmBound] of
            "u" : create upstream-menu for sysp[i, Sediment]
            "i" : create internal boundary menu for sysp[i, Sediment]
            "c" : create confluence boundary menu for sysp[i, Sediment]
        enter sysp[i, Sediment]
        case sysp[i, UpstrmBound] of
            "u" : case sysp[i, Sediment] of
                "c" : enter Sextern[i, 0]
                    for i1 := 1 to ntstep
                        Sextern[i, i1] := Sextern[i, 0]
                "f" : for i1 := 0 to ntstep
                    enter Sextern[i, i1]
            "o", "b", "t" : for i1 := 0 to ntstep
                Sextern[i, i1] := 0
            "i", "c" : case sysp[i, Sediment] of
                "e" : enter Sextern[i, 0]
                    for i1 := 1 to ntstep
                        Sextern[i, i1] := Sextern[i, 0]
                "u" : for i1 := 0 to ntstep
                    Sextern[i, i1] := 0
        ask if the input was right for branch[i]
    until input right
```

- 13 -
2.2.6 AskWaterlevel

**Purpose:**

Input of the downstream boundary condition for the waterlevel of each branch.

**Usage:**

`AskWaterlevel(ntstep, deltaT, dostrm, upstrm, hConst, hFact, hExp, hExtern, Qextern, sysp);`

**Description of parameters:**

- `ntstep`: number of time steps unchanged on exit
- `deltaT`: time step unchanged on exit
- `dostrm`: contains number of the downstream branch unchanged on exit
- `upstrm`: contains number of the upstream branch(es) unchanged on exit
- `hConst`: used in Q-h relation \( h = h_{\text{Const}} + h_{\text{Fact}} \cdot Q^{h_{\text{Exp}}} \) changed on exit
- `hFact`: see `hConst`
- `hExp`: see `hConst`
- `hExtern`: downstream waterlevel or sometimes used for Q-h curve changed on exit
- `Qextern`: for use with Q-h curve sometimes changed on exit
- `sysp`: system parameters some of them changed on exit

**Remarks:**

The input of the various data is controlled by two system parameters:

- `sysp[i,DownstrmBound]`:
  - "d": branch has a real downstream boundary
  - "i": branch has an internal downstream boundary
  - "c": branch has a confluence at the downstream boundary

- `sysp[i,Waterlevel]`:
  - "c": constant waterlevel at downstream boundary
  - "q": downstream waterlevel special function of discharge
  - "g": downstream waterlevel general function of discharge
  - "e": downstream waterlevel equal to the upstream waterlevel of the downstream branch

The value of `sysp[i,DownstrmBound]` is determined by the program itself with the use of `upstrm` and `dostrm`. According to the value of `sysp[i,DownstrmBound]` the user is asked for an external or an internal boundary condition.
structure AskWaterlevel

for i := 1 to nbranch
repeat

case dostrm[i] of

"0" : sysp[i,DownstrmBound] := "d"
create downstream-menu for sysp[i,Waterlevel]

else

if upstrm[dostrm[i],2] = 0 then
sysp[i,DownstrmBound] := "i"
else
sysp[i,DownstrmBound] := "c"
create internal boundary menu for sysp[i,Waterlevel]

enter sysp[i,Sediment]
hConst[i] := 0
hFact[i] := 0
hExp[i] := 0
for ii := 1 to ntstep
hExtern[ii,i] := 0

end

case sysp[i,DownstrmBound] of

"d" : case sysp[i,Waterlevel] of

"c" : enter hExtern[ii,i]
for ii := 1 to ntstep
hExtern[ii,i] := hExtern[ii,0]

"g" : enter Q-h relation
(array Qextern and hExtern are used!)

"q" : enter hConst[i]
enter hFact[i]
enter hExp[i]

"i",
"c" : case sysp[i,Waterlevel] of

"g" : enter Q-h relation
(array Qextern and hExtern are used!)

"q" : enter hConst[i]
enter hFact[i]
enter hExp[i]

ask if the input was right for branch[i]
until input right
2.2.7 AskRoughness

Purpose:
Input of the alluvial roughness.

Usage:
AskRoughness(nstep, deltaX, Roughness1, Roughness2, sysp);

Description of parameters:

- **nstep**: number of place steps
  - unchanged on exit
- **deltaX**: place step
  - unchanged on exit
- **Roughness1**: roughness of mainstream
  - changed on exit
- **Roughness2**: roughness of floodplain or wall
  - changed on exit
- **sysp**: system parameters
  - some of them changed on exit

Remarks:
The input of the various data is controlled by two system parameters:

- **sysp[i,WidthFloodpln]**:
  - "0": branch has no floodplain
  - "c": branch has floodplain

- **sysp[i,Roughness]**:
  - friction-term in the flow-profile calculation should be calculated with:
    - "a": constant Chezy value and waterdepth
    - "b": variable Chezy value per grid point and the waterdepth
    - "c": as "a" but with hydraulic radius
    - "d": as "b" but with hydraulic radius
    - "e": constant K value and the waterdepth
    - "f": variable K value per grid point and the waterdepth
    - "g": as "e" but with hydraulic radius
    - "h": as "f" but with hydraulic radius
    - "i": Van Rijn roughness predictor
    - "j": Van Rijn roughness predictor with Einstein hydraulic radius calculation

The value of the **sysp[i,WidthFloodpln]** is known from AskWidth. According to the value of **sysp[i,Roughness]** the user is asked for data.
structure AskRoughness

for i := 1 to nbranch
repeat
  create menu for sysp[i, Roughness]
  enter sysp[i, Roughness]
  case sysp[i, WidthFloodpln] of
    "0": case sysp[i, Roughness] of
      "a", "c", "e", "g": enter Roughness1[i, 0]
      for i1 := 0 to nstep[i]
        Roughness1[i, i1] := Roughness1[i, 0]
        Roughness2[i, i1] := 0
      "b", "d", "f", "h": for i1 := 0 to nstep[i]
        enter Roughness1[i, i1]
        Roughness2[i, i1] := 0
      "i": for i1 := 0 to nstep[i]
        Roughness1[i, i1] := 0
        Roughness2[i, i1] := 0
      "j": enter estimated Roughness1[i, 0]
         enter wall roughness Roughness2[i, 0]
         for i1 := 1 to nstep[i]
           Roughness1[i, i1] := Roughness1[i, 0]
           Roughness2[i, i1] := Roughness2[i, 0]
    "c", "v": case sysp[i, Roughness] of
      "a", "c", "e", "g": enter Roughness2[i, 0]
      for i1 := 0 to nstep[i]
        Roughness1[i, i1] := Roughness1[i, 0]
        Roughness2[i, i1] := Roughness2[i, 0]
      "b", "d", "f", "h": for i1 := 0 to nstep[i]
        enter Roughness1[i, i1]
        enter Roughness2[i, i1]
      "i": for i1 := 0 to nstep[i]
        Roughness1[i, i1] := 0
        Roughness2[i, i1] := 0
      "j": option not possible
      ask if the input was right for branch[i]
  until input right

- 17 -
2.2.8 AskGrainsize

Purpose:

Input of the grain size of the bed material (D90 and D50).

Usage:

AskGrainsize(nstep, deltaX, D50, D90, sysp);

Description of parameters:

nstep : number of place steps unchanged on exit
deltaX : place step unchanged on exit
D50 : grainsize exceeded by 50% of the material changed on exit
D90 : grainsize exceeded by 10% of the material changed on exit
sysp : system parameter changed on exit

Remarks:

The input of the data is controlled by one system parameter:

sysp[i,Grain] :
"c" : constant grain size in this branch
"v" : grain size variable per grid point

Often it is not necessary to give the grain size (e.g. when the power law is used as transport formula). In this case the user can enter 0 for the grain size.
structure AskGrainsize

for i := 1 to nbranch
repeat
  create D50 menu for sysp\[i,Grain\]
  enter sysp\[i,Grain\]
  case sysp\[i,Grain\] of
    "c" : enter D50\[i,0\]
      for i1 := 1 to nstep\[i\]
        D50\[i,i1\] := D50\[i,0\]
    "v" : for i1 := 0 to nstep\[i\]
      enter D50\[i\]
  create D90 menu for sysp\[i,Grain\]
  enter sysp\[i,Grain\]
  case sysp\[i,Grain\] of
    "c" : enter D90\[i,0\]
      for i1 := 1 to nstep\[i\]
        D90\[i,i1\] := D90\[i,0\]
    "v" : for i1 := 0 to nstep\[i\]
      enter D90\[i\]
  ask if the input was right for branch\[i\]
until input right
2.2.9 AskFormula

Purpose:
Input of the wanted transportformula.

Usage:
AskFormula(mtr, ntr, ucr, sysp);

Description of parameters:

mtr : factor in transportformula changed on exit
ntr : factor in transportformula changed on exit
ucr : factor in transportformula changed on exit
sysp : system parameter changed on exit

Remarks:
The input of the data is controlled by one systemparameter:

sysp[i,Formula] :
"a" : Ackers & White is selected
"e" : Engelund & Hansen is selected
"m" : Meyer-Peter & Mueller is selected
"p" : power law is selected

See Part III for more information on the transportformulae.
### Structure AskFormula

```plaintext
for i := 1 to nbranch
  repeat
    create menu for sysp[i,Formula]
    enter sysp[i,Formula]
    case sysp[i,Formula] of
      "a" : enter multiplication factor mtr[i]
            ntr[i] := 0
            ucr[i] := 0
      "e" : enter multiplication factor mtr[i]
            ntr[i] := 0
            ucr[i] := 0
      "m" : enter mtr[i] (13.3)
            enter ucr[i] (0.047)
            ntr[i] := 0
      "p" : enter mtr[i]
            enter ntr[i]
            ucr[i] := 0
    ask if the input was right for branch[i]
  until input right
```
2.2.10 AskSource

Purpose:
Input of an eventual external sediment source in the equation for sediment continuity.

Usage:
AskSource(seds, sysp);

Description of parameters:

seds : sediment source (in m/s)
changed on exit

sysp : system parameter
changed on exit

Remark:
The input of the data is controlled by one system parameter:

sysp[i,Source] :
"n" : no external sediment source
"c" : constant external sediment source

Structure AskSource

for i := 1 to nbranch
repeat
create menu for sysp[i,Source]
enter sysp[i,Source]
case sysp[i,Source] of
"n" : seds[i] := 0
"c" : enter seds[i]
ask if the input was right for branch[i]
until input right
2.2.11 WriteDataToFile

- Purpose:

Write the output data to a file. On a microcomputer the User is asked to type the name of the file on the keyboard, on the mainframe version the necessary job control must contain the name of the file.

- Usage:

```
WriteDataToFile(nbranch, ntstep, deltaT, tmax, nstep, dostrm, upstrm, length, deltaX, ntr, mtr, ucr, seds, hConst, hFact, hExp, width1, width2, widthS, Bedlevel2, Roughness1, Roughness2, d50, d90, Bedlevel1, hExtern, Qextern, Sextern, sysp, tOutput, InputDataname, datafile);
```

InputDataname should be omitted in the mainframe version.

- Remark:

WriteDataToFile does not change the value of any parameter.
Part IV: Program Description

structure WriteDataToFile

open file

for i := 1 to 6
    write(tOutput[i])

for i := 1 to nbranch
    write(length[i], deltaX[i], nstep[i], dostrm[i])
    write(upstrm[i,1], upstrm[i,2], ntr[i], mtr[i], ucr[i])
    write(seds[i], hConst[i], hFact[i], hExp[i])

for i := 1 to nbranch
    for i2 := WidthMainstrm to Source
        write(sysp[i,i2])

for i := 1 to nbranch
    for i1 := 0 to nstep[i]
        write(width1[i,i1], width2[i,i1], widthS[i,i1])
        write(Bedlevel1[i,i1,0], Bedlevel2[i,i1])
        write(Roughness1[i,i1], Roughness2[i,i1])
        write(D50[i,i1], D90[i,i1])

for i := 1 to nbranch
    for i1 := 0 to ntstep
        write(Qextern, Sextern, hExtern)

close datafile
Part IV: Program Description

2.2.12 Whatnext

Purpose:
Bring the user back to the main menu.

Usage:
Whatnext(OutputDataName);

Description of parameters:
OutputDataName: name of the created datafile

structure Whatnext

| print the name of the datafile |
| execute MENU |
3. Computer program ODIRMO

3.1 Main Program

ODIRMO is written in the programming language Pascal on a IBM PC at the Delft University of Technology, department of Civil Engineering, section Fluid Mechanics. An implementation on the IBM mainframe of the University has been made. ODIRMO's aim is to calculate time dependent morphological changes in alluvial streams with uniform sediment. It therefore needs knowledge about boundary and initial conditions. The input should be prepared with the program CREADATA, which is discussed in chapter 2.

ODIRMO consists of a main program and nine or ten subroutines. Nine for the mainframe version and ten for the micro version. The tenth subroutine of the micro version is a very short routine, which only task it is to ask the user the name of the inputfile and the outputfile. In the mainframe version this information must be given in the jobcontrol cards. The main program controls the flow of the calculations and the calling of subroutines.
Structure ODIRMO

<table>
<thead>
<tr>
<th>InitOdirmo</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadDataFromFile</td>
</tr>
<tr>
<td>for ( t := 0 ) to ( ntstep )</td>
</tr>
<tr>
<td><strong>DischargeCalc</strong></td>
</tr>
<tr>
<td>for iter := 1 to 3</td>
</tr>
<tr>
<td>for ( i := nbranch ) downto 1</td>
</tr>
<tr>
<td>WaterlevelDownstream</td>
</tr>
<tr>
<td>BackwaterCalc</td>
</tr>
<tr>
<td>for ( i := 1 ) to ( nbranch )</td>
</tr>
<tr>
<td>SediTransCalc</td>
</tr>
<tr>
<td>SediTransUpstrm</td>
</tr>
<tr>
<td>t in ([tOutput(1,...,6)])</td>
</tr>
<tr>
<td>Results</td>
</tr>
<tr>
<td>t &lt; ( ntstep )</td>
</tr>
<tr>
<td>for ( i := 1 ) to ( nbranch )</td>
</tr>
<tr>
<td>BedlevelCalc</td>
</tr>
<tr>
<td>WriteDataToFile</td>
</tr>
</tbody>
</table>
3.2 Discussion of subroutines

3.2.1 ReadDataFromFile

- Purpose:
Reads the input data from a file. On a microcomputer the User is asked to type the name of the file on the keyboard, on the mainframe version the necessary job control must contain the name of the file.

- Usage:
ReadDataFromFile(nbranch, tOutput, ntstep, deltaT, tmax, nstep, dostrm, upstrm, length, deltaX, ntr, mtr, ucr, seds, hConst, hFact, hExp, width1, width2, width5, Bedlevel12, Roughness1, Roughness2, d50, d90, Bedlevel1, hExtern, Qextern, Sextern, sysp, InputDataname, datafile);

InputDataname should be omitted in the mainframe version.

- Remark:
The main program passes the data to the various other subroutines.
structure ReadDataFromFile

open file

for i := 1 to 6
   read(tOutput[i])

for i := 1 to nbranch
   read(length[i], deltaX[i], nstep[i], dostrm[i])
   read(upstrm[i,1], upstrm[i,2], ntr[i], mtr[i], ucr[i])
   read(seds[i], hConst[i], hFact[i], hExp[i])

for i := 1 to nbranch
   for i2 := WidthMainstrm to Source
      read(sysp[i,i2])

for i := 1 to nbranch
   for i1 := 0 to nstep[i]
      read(width1[i,i1], width2[i,i1], widthS[i,i1])
      read(Bedlevel1[i,i1,0], Bedlevel2[i,i1])
      read(Roughness1[i,i1], Roughness2[i,i1])
      read(D50[i,i1], D90[i,i1])

for i := 1 to nbranch
   for i1 := 0 to ntstep
      read(Qextern, Sextern, hExtern)

close datafile
3.2.2 DischargeCalc

- Purpose:
Calculates in every timestep the discharge in each branch of the problem. This can be done without calculating flow profiles first, as bifurcations are not allowed.

- Usage:
DischargeCalc(q, nbranch, t, upstrm, Qextern, sysp);

- Description of parameters:

  q : discharge
      changed on exit
  nbranch : number of riverbranches
            unchanged on exit
  t : index indicating the timestep
      unchanged on exit
  upstrm : the number of the upstream branch(es)
           unchanged on exit
  Qextern : boundary condition discharge
            unchanged on exit
  sysp : system parameter indicating which boundary condition should be applied
         unchanged on exit

- Remark:
This subroutine uses the system parameters sysp(i, UpstrmBound) and sysp(i, Discharge). These parameters are characters with some mnemonic meaning:

  sysp(i, UpstrmBound) :
"u" : real upstream boundary
"i" : internal upstream boundary
"c" : confluence at upstream boundary

  sysp(i, Discharge) :
"u" : discharge equal to discharge upstream branch(es)
"e" : water extraction at upstream (internal) boundary
**structure DischargeCalc**

<table>
<thead>
<tr>
<th>For i := 1 to nbranch</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>case sysp[i,UpstrmBound] of</strong></td>
</tr>
<tr>
<td>&quot;u&quot; : real upstream boundary</td>
</tr>
<tr>
<td>q is given by the user</td>
</tr>
<tr>
<td>&quot;i&quot; : internal boundary</td>
</tr>
<tr>
<td>q depends on sysp[i,Discharge]</td>
</tr>
<tr>
<td><strong>case sysp[i,Discharge] of</strong></td>
</tr>
<tr>
<td>&quot;u&quot; : discharge is equal to the discharge of the upstream branch</td>
</tr>
<tr>
<td>&quot;e&quot; : discharge is equal to the discharge of the upstream branch minus withdrawal</td>
</tr>
<tr>
<td>&quot;c&quot; : confluence</td>
</tr>
<tr>
<td>discharge depends on sysp[i,Discharge]</td>
</tr>
<tr>
<td><strong>case sysp[i,Discharge] of</strong></td>
</tr>
<tr>
<td>&quot;u&quot; : discharge is equal to the discharge of the upstream branches</td>
</tr>
<tr>
<td>&quot;e&quot; : discharge is equal to the discharge of the upstream branches minus withdrawal</td>
</tr>
</tbody>
</table>
Part IV: Program Description

### 3.2.3 WaterlevelDownstream

**- Purpose:**
Determines the downstream boundary condition for the waterlevel of a river branch.

**- Usage:**
WaterlevelDownstream(i, t, nstep, dostrm, hConst, hFact, hExp, h, hExtern, sysp);

**- Description of parameters:**

- **i** : index indicating the number of the branch unchanged on exit
- **t** : index indicating the timestep unchanged on exit
- **nstep** : number of spatial steps in the branch unchanged on exit
- **dostrm** : the number of the downstream branch unchanged on exit
- **hConst** : used in formula $h = h_{\text{Const}} + h_{\text{Fact}} \times Q^{h_{\text{Exp}}}$ unchanged on exit
- **hFact** : see hConst
- **hExp** : see hConst
- **h** : waterlevel changed on exit
- **hExtern** : boundary condition for the waterlevel unchanged on exit
- **sysp** : system parameter indicating which boundary condition should be applied

**- Remark:**
This subroutine uses the system parameters `sysp(i, DownstrmBound)` and `sysp(i, Waterlevel)`. These parameters are characters with some mnemonic meaning:

**sysp(i, DownstrmBound):**
- "d": real downstream boundary
- "i": internal downstream boundary
- "c": confluence at downstream boundary

**sysp(i, Waterlevel):**
- "c": constant downstream waterlevel
- "g": general Q-h curve at downstream boundary
- "q": $h = h_{\text{Const}} + h_{\text{Fact}} \times Q^{h_{\text{Exp}}}$
- "e": waterlevel downstream equal to upstream waterlevel of downstream branch
structure WaterlevelDownstream

<table>
<thead>
<tr>
<th>Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>symp[i,DownstreamBound] of</td>
<td></td>
</tr>
<tr>
<td>&quot;d&quot;</td>
<td>real downstream boundary waterlevel depends on symp[i,Waterlevel]</td>
</tr>
<tr>
<td></td>
<td>case symp[i,Waterlevel] of</td>
</tr>
<tr>
<td>&quot;c&quot;</td>
<td>constant waterlevel downstream</td>
</tr>
<tr>
<td>&quot;g&quot;</td>
<td>general q-h relation downstream</td>
</tr>
<tr>
<td>&quot;q&quot;</td>
<td>special q-h relation downstream</td>
</tr>
<tr>
<td>&quot;i&quot;, &quot;c&quot;</td>
<td>internal downstream boundary or confluence waterlevel depends on symp[i,Waterlevel]</td>
</tr>
<tr>
<td></td>
<td>case symp[i,Waterlevel] of</td>
</tr>
<tr>
<td>&quot;e&quot;</td>
<td>waterlevel downstream equal to upstream waterlevel of downstream branch(es)</td>
</tr>
<tr>
<td>&quot;g&quot;</td>
<td>general q-h relation downstream</td>
</tr>
<tr>
<td>&quot;q&quot;</td>
<td>special q-h relation downstream</td>
</tr>
</tbody>
</table>
3.2.4.1 BackwaterCalc

- Purpose:
Calculates the backwater curve for a river branch with or without floodplanes.

- Usage:
BackwaterCalc(iter, nstep, deltaX, width1, width2,
Roughness1, Roughness2, Bedlevel2, d50, d90,
Rb, h, r, Bedlevel1, syzp);

- Description of parameters:
iter : index for the iteration step in Predictor Corrector method
unchanged on exit
nstep : number of spatial steps in the branch
unchanged on exit
deltaX : distance between two grid points
unchanged on exit
width1 : width of mainstream
unchanged on exit
width2 : width of floodplain
unchanged on exit
Roughness1: roughness of mainstream
unchanged on exit
Roughness2: roughness of floodplains or wall roughness
unchanged on exit
Bedlevel2 : bedlevel of floodplains
unchanged on exit
d50 : grain size exceeded by 50% of sediment
unchanged on exit
d90 : grain size exceeded by 10% of sediment
unchanged on exit
Rb : hydraulic radius
changed on exit
h : waterlevel
changed on exit
r : division of discharge between mainstream and floodplain
changed on exit
Bedlevel1 : bedlevel of mainstream
unchanged on exit
sysp : system parameters indicating the presence or absence of floodplains and the way the friction term should be calculated
unchanged on exit

- Remark:
This subroutine uses the system parameters
Part IV: Program Description

sysp(i, WidthFloodpln) and sysp(i, Roughness). These parameters are characters with the following meaning:

sysp(i, WidthFloodpln):
"0": no floodplains
"e", "f": floodplains present

sysp(i, Roughness):
passed to the function RoughTerm

The method of the calculations is discussed in Part III.
structure BackwaterCalc

<table>
<thead>
<tr>
<th>T</th>
<th>iter = 1</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>predictor step</td>
<td></td>
<td>corrector step</td>
</tr>
</tbody>
</table>

**case sysp[i, WidthFloodpln] of**

- **"c"**: "s", "v":
  - calculation with floodplain
  - for il := nstep[i] downto 1
    - determine friction term mainstream at il
    - determine friction term floodplain at il
    - determine convective term due to change of mainstream width at il
    - determine convective term due to change of floodplain width at il
    - predict distribution of discharge
    - predict water level at il-1
    - for it := 1 to 2
      - determine friction term mainstream at il-1
      - determine friction term floodplain at il-1
      - determine convective term due to change of mainstream width at il-1
      - determine convective term due to change of floodplain width at il-1
      - determine convective term due to change of mainstream depth at il
      - determine convective term due to change of floodplain depth at il
      - correct distribution of discharge at il
      - predict distribution of discharge at il-1
      - correct water level at il-1

- **"s"**: calculation without floodplain
  - for il := nstep[i] downto 1
    - determine friction term at il
    - determine convective term due to change of width at il
    - predict water level at il-1
    - for it := 1 to 2
      - determine friction term at il-1
      - determine convective term due to change of width at il-1
      - determine convective term due to change of depth at il
      - correct water level at il-1
3.2.4.2 Function Roughterm

- Purpose:

This function is used by the routine BackwaterCalc to calculate the friction term in the equation.

- Usage:

\[
\text{Variable := Roughterm}(i, i1, \text{Rough1}, \text{Rough2}, \text{Rb}, \text{d50}, \text{d90}, \text{width}, a1, \text{sysp});
\]

- Description of parameters:

  \begin{itemize}
  \item \text{i} : index indicating the number of the branch unchanged on exit
  \item \text{i1} : index indicating the place in the branch unchanged on exit
  \item \text{Rough1} : roughness of the bed sometimes changed on exit
  \item \text{Rough2} : roughness of the wall unchanged on exit
  \item \text{Rb} : hydraulic radius changed on exit
  \item \text{d50} : grain size exceeded by 50% of sediment unchanged on exit
  \item \text{d90} : grain size exceeded by 10% of sediment unchanged on exit
  \item \text{width} : the width of the stream unchanged on exit
  \item \text{a1} : the local waterdepth unchanged on exit
  \item \text{sysp} : system parameter indicating in which way the friction term should be calculated
  \end{itemize}

- Remarks:

Rough1 is changed when a roughness predictor with an estimated Chezy value is used. The estimated value will be replaced by the calculated value. This means that in the next step a smaller number of iterations will probably be sufficient.

Rough2 is the wall roughness which is only used in case a roughness predictor or hydraulic radius predictor is used. In other cases "0" may be passed by this parameter.

Rb, the hydraulic radius, is sometimes used by the routine SeditransCalc. Some transport formulae use it.

This function uses the system parameter \text{sysp}(i, \text{Roughness}) to decide in which way the friction term should be calculated. This is discussed in part III. The possible values are:

- "a": calculate with Chezy value and waterdepth
- "b":
Part IV: Program Description

"d": calculate with Chezy value and hydraulic radius
"e":
"f": calculate with Nikuradse roughness and waterdepth
"g":
"h": calculate with Nikuradse roughness and hydraulic radius
"i": roughness predictor (see chapter 3)
"j": predictor for hydraulic radius (Einstein) (chapter 3)
3.2.5 SeditransCalc

- **Purpose:**

Calculate the sediment transport at each point in a branch with the desired transport formula.

- **Usage:**

```
SeditransCalc(iter, i, nstep, deltaX, q, mtr, ntr, ucr, h, width1, widthS, r, d50, d90, Roughness1, Rb, S, Bedlevel1, sysp);
```

- **Description of parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>iter</code></td>
<td>index Predictor Corrector method unchanged on exit</td>
</tr>
<tr>
<td><code>i</code></td>
<td>index branch number unchanged on exit</td>
</tr>
<tr>
<td><code>nstep</code></td>
<td>number of place steps in branch unchanged on exit</td>
</tr>
<tr>
<td><code>deltaX</code></td>
<td>distance between two grid points unchanged on exit</td>
</tr>
<tr>
<td><code>q</code></td>
<td>discharge unchanged on exit</td>
</tr>
<tr>
<td><code>mtr</code></td>
<td>factor in transport formula unchanged on exit</td>
</tr>
<tr>
<td><code>ntr</code></td>
<td>factor in transport formula unchanged on exit</td>
</tr>
<tr>
<td><code>ucr</code></td>
<td>critical flow velocity unchanged on exit</td>
</tr>
<tr>
<td><code>h</code></td>
<td>waterlevel unchanged on exit</td>
</tr>
<tr>
<td><code>width1</code></td>
<td>width of mainstream unchanged on exit</td>
</tr>
<tr>
<td><code>widthS</code></td>
<td>sediment transporting width unchanged on exit</td>
</tr>
<tr>
<td><code>r</code></td>
<td>division of discharge between mainstream and floodplains unchanged on exit</td>
</tr>
<tr>
<td><code>d50</code></td>
<td>grain size exceeded by 50% of sediment unchanged on exit</td>
</tr>
<tr>
<td><code>d90</code></td>
<td>grain size exceeded by 10% of sediment unchanged on exit</td>
</tr>
<tr>
<td><code>Roughness1</code></td>
<td>roughness of mainstream unchanged on exit</td>
</tr>
<tr>
<td><code>Rb</code></td>
<td>hydraulic radius unchanged on exit</td>
</tr>
<tr>
<td><code>S</code></td>
<td>sediment transport changed on exit</td>
</tr>
<tr>
<td><code>Bedlevel1</code></td>
<td>bedlevel of mainstream unchanged on exit</td>
</tr>
<tr>
<td><code>sysp</code></td>
<td>system parameter indicating which transport formula is to be used</td>
</tr>
</tbody>
</table>
- Remarks:

It is obvious that not for all transport formulae all parameters must have a value. For example, when the power law is used \( s = \text{mtr} \times u^n \) the parameters \( u_{cr}, d_{50}, d_{90}, \) Roughness1, \( R_b \) are not used.

The system parameter \( \text{sysp}(i,\text{Formula}) \) can have the following values:

- "a" : Ackers and White is used
- "e" : Engelund and Hansen is used
- "m" : Meyer-Peter and Mueller is used
- "p" : Power law is used
### Structure SediTransCalc

<table>
<thead>
<tr>
<th>T</th>
<th>iter = 1</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>predictor step</td>
<td></td>
<td>corrector step</td>
</tr>
</tbody>
</table>

for \( i1 := 0 \) to \( nstep[i1] \)

- calculate flow velocity
- case \( sysp[i,Formula] \) of

  - "a": calculate transport with Ackers & White formula
  - "m": calculate transport with Meyer-Peter & Mueller formula
  - "e": calculate transport with Engelund & Hansen formula
  - "p": calculate transport with \( s = m \times u^n \)
3.2.6 SeditransUpstrm

- **Purpose:**

Determines the upstream boundary condition for the sediment transport.

- **Usage:**

SeditransUpstrm(iter, i, t, nstep, upstrm, S, Sextern, sysp);

- **Description of parameters:**

  iter : index Predictor Corrector method
         unchanged on exit

t : index branchnumber
   unchanged on exit

i : index indicating the timestep
    unchanged on exit

upstrm : index-number of the upstream branch(es)
         unchanged on exit

S : transport
   changed on exit

Sextern : transport boundary condition
          unchanged on exit

sysp : system parameter indicating the type of boundary condition

- **Remarks:**

It will be clear that only the \( S(0) \) of a branch leaves this routine changed.

The system parameters used by this routine are sysp(i,UpstrmBound) and sysp(i,Sediment). The first one can have the following values:

"u" : real upstream boundary
"i" : internal boundary upstream
"c" : confluence upstream

The second can have the values:

"c" : input of sediment constant
"f" : input of sediment function of time
"b" : bedlevel constant
"o" : input equal to output at end of flume
"u" : input equal to output of upstream branch(es)
"e" : input equal to output of upstream branch(es) minus a sediment extraction
### Structure SediTransUpstrm

<table>
<thead>
<tr>
<th>predictor step</th>
<th>corrector step</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>case sy</strong>p[i,UpstrmBound] of</td>
<td></td>
</tr>
<tr>
<td>&quot;u&quot; : real upstream boundary boundary condition depends on sy<strong>s</strong>p[i,Sediment]</td>
<td></td>
</tr>
<tr>
<td>case sy<strong>s</strong>p[i,Sediment] of</td>
<td></td>
</tr>
<tr>
<td>&quot;c&quot; :</td>
<td></td>
</tr>
<tr>
<td>&quot;f&quot; : constant sediment input or sediment input function of time</td>
<td></td>
</tr>
<tr>
<td>&quot;b&quot; : bedlevel at upstream boundary constant</td>
<td></td>
</tr>
<tr>
<td>&quot;o&quot; : input at upstream boundary equal to sediment output at downstream boundary (closed circuit)</td>
<td></td>
</tr>
<tr>
<td>&quot;i&quot; : internal boundary boundary condition depends on sy<strong>s</strong>p[i,Sediment]</td>
<td></td>
</tr>
<tr>
<td>case sy<strong>s</strong>p[i,Sediment] of</td>
<td></td>
</tr>
<tr>
<td>&quot;u&quot; : transport equal to transport at the end of the upstream branch</td>
<td></td>
</tr>
<tr>
<td>&quot;e&quot; : transport equal to transport at the end of the upstream branch minus a sediment extraction</td>
<td></td>
</tr>
<tr>
<td>&quot;c&quot; : confluence at upstream boundary boundary condition depends on sy<strong>s</strong>p[i,Sediment]</td>
<td></td>
</tr>
<tr>
<td>case sy<strong>s</strong>p[i,Sediment] of</td>
<td></td>
</tr>
<tr>
<td>&quot;u&quot; : transport equal to transport at the end of the upstream branches</td>
<td></td>
</tr>
<tr>
<td>&quot;e&quot; : transport equal to transport at the end of the upstream branches minus a sediment extraction</td>
<td></td>
</tr>
</tbody>
</table>
3.2.7 Results

- Purpose:

Print the results of the calculations after the desired number of timesteps. The micro version types direct on the printer, the mainframe version writes to the standard output file.

- Usage:

Results(t, q, r, h, width1, S, Bedlevel1);

- Description of parameters:

  t : index of timestep
  q : discharge
  r : division of discharge between mainstream and floodplains
  width1 : width of mainstream
  S : sediment transport
  Bedlevel1 : bedlevel of mainstream

All parameters unchanged on exit.
3.2.8 BedlevelCalc

- Purpose:

Calculate the new bedlevel for every timestep. In fact, this routine is used three times in every timestep, as a predictor corrector method is used with one prediction and two corrections.

- Usage:

BedlevelCalc(iter, i, nstep, deltaX, widthl, S, Bedlevel1);

- Description of parameters:

iter : index of predictor corrector method unchanged on exit
i : index of branch number unchanged on exit
nstep : number of place-steps in the branch unchanged on exit
deltaX : distance between two grid points unchanged on exit
widthl : width of mainstream unchanged on exit
S : sediment transport unchanged on exit
Bedlevel1 : bedlevel of mainstream changed on exit

- Remark:

The predictor corrector method is treated in Part III.
structure BedlevelCalc

\[
\begin{array}{c|c|c}
\text{T} & \text{iter} = 1 & \text{F} \\
\hline
\text{prepare transport for predictor step} & & \\
\hline
\text{apply numerical scheme for upstream boundary} & & \\
\hline
\text{for i1 := 1 to nstep[i]-1} & & \\
\text{apply numerical 'inside' scheme} & & \\
\hline
\text{apply numerical scheme for downstream boundary} & & \\
\hline
\text{T} & \text{iter} = 3 & \text{F} \\
\hline
\text{prepare bedlevel for next time-step} & & \\
\end{array}
\]

3.2.9 WriteDataToFile

- Purpose:
Writing the results at the end of the calculations to a file which can be used for further calculations. This file is exactly the same as the original input file, only the bedlevel of the mainstream has changed. The micro version asks for a name of the output file, which is to be given with the keyboard. In the mainframe version the jobcontrol has to indicate this output file. The micro version also sends a copy of the output file to the printer.

- Remarks:
The use is exactly the same as the routine ReadDataFromFile. In fact, it is almost the same routine, only 'read' statements are replaced with 'write' statements.
Appendix A

The most important variables are listed below:

- **branchmax**: maximum number of sections
- **stepmax**: maximum number of spatial steps in a section
- **timestepmax**: maximum number of timesteps

**integer**:
- **iter**: number of iterations predictor-corrector method
- **nbranch**: number of riversections
- **ntstep**: number of timesteps

**real**:
- **deltaT**: timestep used in the calculations
- **tmax**: time of calculations

1 dim integer array(1..6)
- **tOutput**: indicates after how many timesteps output of calculations is wanted

1 dim integer array(1..branchmax):
- **dostrm**: number of the downstream branch of the considered branch
- **nstep**: number of spatial steps in a section

2 dim integer array(1..branchmax, 1..2):
- **upstrm**: index number of the upstream branch(es)

1 dim real array(1..branchmax):
- **deltaX**: spatial step used in the calculations
- **hExp**: exponent in q-h relation: \( Q = h_{\text{Const}} + h_{\text{Fact}} \cdot Q^\ast h_{\text{Exp}} \)
- **hConst**: see hExp
- **hFact**: see hExp
- **length**: length of riverbranch
- **mtr**: used in various transport formulae
- **ntr**: used in various transport formulae
- **q**: discharge
- **seds**: external sediment source over the whole riversection
- **ucr**: critical flow velocity used in transportformula

2 dim real array(1..branchmax, 1..stepmax):
- **Bedlevel2**: bedlevel of floodplains
- **d50**: grain diameter which is exceeded by 50% of the particles
- **d90**: grain diameter which is exceeded by 10% of the particles
- **h**: waterlevel at a certain grid point
Part IV: Program Description

\( r \) : division of discharge between mainstream and floodplain

Roughness1 : roughness of mainstream

Roughness2 : roughness of mainstream or roughness of wall

width1 : width of mainstream

width2 : width of floodplain

widthS : sediment transporting width

3 dim real array \((1..\text{branchmax}, 1..\text{stepmax}, 0..1)\):

Bedlevel1 : bedlevel of mainstream

S : sediment transport

2 dim real array \((1..\text{branchmax}, 1..\text{timestepmax})\):

hExtern : \( Q-h \) relation at downstream boundary: \( h_{\text{Extern}} \) given for some \( Q_{\text{Extern}} \)

QExtern : \( Q \) boundary condition as function of time OR used in \( Q-h \) relation (see \( h_{\text{Extern}} \))

SExtern : transport boundary condition

2 dim char array \((1..\text{branchmax}, \text{WidthMainstrm}..\text{Source})\):

sysp : system parameter indicating which options of the program are activated
This is a completely menu-driven program to create an input datafile for ODIRMO, a program for one dimensional river morphology.

--- DECLARATION SECTION ---

type

component = (WidthMainstrm,WidthFloodpln,WidthTransp, BedlevelMainstrm,BedlevelFloodpln, UpstrmBound,Discharge,Sediment,DownstrmBound, Waterlevel,Roughness,Grain,Formula,Source);

const

branchmax = 6;
stepmax = 90;
timestepmax = 100;

filename = string[8];
ArrayType1 = array[1..branchmax] of Byte;
ArrayType2 = array[0..branchmax,1..2] of Byte;
ArrayType3 = array[1..branchmax] of Real;
ArrayType4 = array[1..branchmax,0..stepmax] of Real;
ArrayType5 = array[1..branchmax,0..stepmax,0..1] of Real;
ArrayType6 = array[1..branchmax,0..timestepmax] of Real;
ArrayType7 = array[1..branchmax,WidthMainstrm..Source] of Char;
ArrayType8 = array[1..6] of Integer;

var

nbranch : Byte;
ntstep : Integer;
deltaT,
tmax : Real;

OutputDataname : filename;
dostrm,
nstep : ArrayType1;

upstrm : ArrayType2;
deltaX,
length,
hConst,
hFact,
hExp,
mtr,
mtr,
ucr,
seds : ArrayType3;
roughness1, roughness2, d50, d90, width1, width2, width3, Bedlevell : ArrayType4;
Bedlevel1 : ArrayType5;

hExterrn, Sextern, Dextern : ArrayType6;
sysp : ArrayType7;
tOutput : ArrayType8;

NextProg : file;
datafile : text;

{"---------------------- END DECLARATION SECTION ----------------------}

{"-------------------------- INCLUDED FILES --------------------------}

{"InputChe.pas}
- "{*******************************************************}
- {-} routine InputCheck (CREADATA)
- "{*******************************************************}
- ( checks the input from keyboard
- 
- procedure InputCheck(var LegalEntry : Boolean);
- begin
- LegalEntry := (IOresult = 0);
- if not LegalEntry then
- begin
- highvideo;
- writeln('ILLEGAL ENTRY, TRY AGAIN : ');
- end; (of condition)
- end; (of procedure InputCheck)

{"EnterRea.pas}
- "{*******************************************************}
- {-} routine EnterReal (CREADATA)
- "{*******************************************************}
- ( to enter a real value
- 
- procedure EnterReal (Var RealNumber : real);
- 
- Var
- LegalEntry : boolean;
- begin
- repeat
- highvideo;
- readln(RealNumber);
- InputCheck(LegalEntry);
- until LegalEntry;
- lowvideo;
- end; (of procedure EnterReal)
{File: InitCrea.pas}
-
-{******************************************************************************}
-{ routine InitCrea (CREADATA) }
-{******************************************************************************}
-{ asks general data }
-
-procedure InitCrea(var OutputDataname :filename;
 var nbranch :Byte;
 var ntstep :Integer;
 var tmax, deltaT :Real;
 var nstep, dostrm :ArrayType1;
 var upstrm :ArrayType2;
 var length, deltaX :ArrayType3;
 var tOutput :ArrayTypeB);
-
-var
 LegalEntry : Boolean;
 i : Byte;
 Sure : Char;
-
-begin
-repeat
-Clrscr;
-lowvideo;
-gotoXY(20,6);
-writeln('Creating a new datafile.');
-gotoXY(12,12);
-writeln('Just answer my questions with your data.');
-highvideo;
-gotoXY(19,20);
-write('press any key to continue');
-repeat until keypressed;
-Clrscr;
-gotoXY(20,8);
-lowvideo;
-writeln('What is the name of your datafile ?');
-gotoXY(30,10);
-highvideo;
-repeat
-readln(OutputDataname);
 InputCheck(LegalEntry);
-until LegalEntry;
-lowvideo;
-Clrscr;
-gotoXY(1,5);
-write('The number of riverbranches is : ');
-highvideo;
-repeat
-readln(nbranch);
 InputCheck(LegalEntry);
-until LegalEntry;
-if nbranch > branchmax then
-begin
-highvideo;
-write;
-write('Too many branches ');;
-write('CHANGE SETTING OF BRANCHMAX FIRST');
-
-end;
-lowvideo;
-write;
-write('What is the time (in hours) ?');
-write('of the calculations? ');
-EnterReal(tmax);
-tmax:=tmax * 60 * 60;
-write;
-write('What is the time step (in hours) ?');
-write('of the calculations? ');
-EnterReal(deltaT);
-deltaT:=deltaT * 60 * 60;
-ntstep:=round(tmax/deltaT);
if ntstep > timestepmax then
begin
  highvideo;
  writeln;
  writeln('Too many timesteps !');
  writeln('CHANGE SETTING OF TIMESTEPMAX FIRST');
end;
writeln;
writeln('After how many timesteps do you want output of results ? ');
writeln('A maximum of 6 output levels is possible.');
for i:=1 to 6 do
begin
  tOutput[i] := ntstep;
  write('tOutput[i,1] = '); repeat
    highvideo;
    readln(tOutput[i]);
    InputCheck(LegalEntry);
    lowvideo;
  until LegalEntry;
end;
clrscr;
for i:=1 to nbranch do
begin
  writeln;
  lowvideo;
  write('What is the length of branch ',i,' : '); EnterReal(length[i]);
  write('What is the spatial step for branch ',i,' : '); EnterReal(deltaX[i]);
  nstep[i] := round(length[i]/deltaX[i]);
  if nstep[i] > stepmax then
begin
  highvideo;
  writeln;
  writeln('Too many spatial steps !');
  writeln('CHANGE SETTING OF STEPMAX FIRST');
  writeln;
end;
writeln('What is the number of the downstream branch, ? ');
write('give "0", when no downstream branch present : '); highvideo;
repeat
  readln(d ostrm[i]);
  InputCheck(LegalEntry);
until LegalEntry;
end;(of do-loop)
for i:=1 to nbranch do
begin
  upstrm[i,1] := 0;
  upstrm[i,2] := 0;
end;(of do-loop)
for i:=1 to nbranch do
begin
  if upstrm[d ostrm[i],1] = 0 then
  upstrm[d ostrm[i],1] := i
  else
  upstrm[d ostrm[i],2] := i;
end;(of do-loop)
writeln;
highvideo;
repeat
  write('ARE YOU SURE ? (Y/N) ');
  read(kbd, sure);
  writeln;
  until sure in ['y','Y','n','N'];
  until Sure in ['y','Y'];
end;(of procedure InitCrea)
{SI AskWidth.pas:)
- (**-----------------------------------------------**
- ( routine AskWidth (CREADATA) )
- (**-----------------------------------------------**
- (- this routine asks the width of the mainstream, floodplain and ----
- - transporting width for each riverbranch -------------------)
- procedure AskWidth ( nstep :ArrayType1;
- deltaX :ArrayType3;
- var width1, width2, widthS :ArrayType4;
- var sysp :ArrayType7);
- var
- LegalEntry : Boolean;
- il, il : Byte;
- Sure : Char;
- begin
- for i:=1 to nbranch do
- begin
- repeat
- {------------------- CREATION OF MENU ----------------------------}
- writeln('The width of the mainstream in branch ',i,' is :');
- write('1. '); highvideo;
- write('C'); lowvideo;
- writeln('Constant');
- write('2. '); highvideo;
- write('V'); lowvideo;
- writeln('variable per grid point');
- writeln;
- highvideo;
- LegalEntry := False;
- repeat
- writeln('Enter your choice and press <RETURN> : '); Read(kbd,syspl,WidthMainstrm);
- writeln;
- if syspl[i,WidthMainstrm] in ['c','C','v','V'] then
- LegalEntry := True;
- if not LegalEntry then
- writeln('ILLEGAL ENTRY, TRY AGAIN : ');
- until LegalEntry;
- writeln;
- writeln('The width of the floodplain in branch ',i,' is : ');
- write('0. Constant = '); highvideo;
- write('0'); lowvideo;
- writeln(' (no floodplains)');
- write('1. '); highvideo;
- write('C'); lowvideo;
- writeln('Constant > 0');
- write('2. '); highvideo;
- write('V'); lowvideo;
- writeln('variable per grid point');
- writeln;
- highvideo;
- LegalEntry := False;
- 53
repeat
  write('Enter your choice and press <RETURN> : ');
  Read(Kbd,sysp[i,WidthFloodpln]);
  writeln;
  if sysp[i,WidthFloodpln] in ['0', 'c', 'C', 'v', 'V'] then
    LegalEntry := True;
  if not LegalEntry then
    writeln('ILLEGAL ENTRY, TRY AGAIN : ');
until LegalEntry;
lowvideo;
writeln;
writeln('The sediment-transporting width in branch ',i,' is : ');
write('0. ');
highbideo;
writeln('E');
lowvideo;
writeln('equal to the width of the mainstream');
write('1. ');
highbideo;
writeln('C');
lowvideo;
writeln('constant (but not equal to mainstream width)');
write('2. ');
highbideo;
writeln('variable per grid point');
writeln;
highvideo;
LegalEntry := False;
repeat
  write('Enter your choice and press <RETURN> : ');
  Read(Kbd,sysp[i,WidthTransprt]);
  writeln;
  if sysp[i,WidthTransprt] in ['e', 'E', 'c', 'C', 'v', 'V'] then
    LegalEntry := True;
  if not LegalEntry then
    writeln('ILLEGAL ENTRY, TRY AGAIN : ');
until LegalEntry;
(----------------------------- END OF MENU-SECTION -----------------------------)
(---------------------------- DATA INPUT -----------------------------------)
clearscr;
write('branch ',i);
writeln;
lowvideo;
case sysp[i,WidthMainstrn] of
  'c', 'C' : begin
    write('Give the constant width of the mainstream : ');
EnterReal(width1[i,0]);
    for i1:=0 to nstep[i] do
      width[i1,i]:=width1[i,0];
    end ;(of case)
  'v', 'V' : begin
    writeln('Give the mainstream-width at each point : ');
    for i1:=0 to nstep[i] do
      begin
        write('point at ',deltaX[i]*i1:6:0,' meters : ');
EnterReal(width1[i,i1]);
      end ;(of do-loop)
  end ;(of case)
end ;(of WidthMainstrn case statement)
procedure AskBedlevel (nstep : ArrayType1; deltaX : ArrayType3; var Bedlevel2 : ArrayType4; var Bedlevel1 : ArrayType5; var sysp : ArrayType7) ;

var
LegalEntry : Boolean;

i, il : Byte;
Position, Bedlev, Slope : Real;
Sure : Char;

begin
for i:=l to nbranch do
begin
repeat
(-- CREATION OF MENU --)
highvideo;
c1rscr;
writeln('branch ',i); writeln;
lowvideo;
writeln('Do you know the original bedlevel?');
highvideo;
writeln('E'); lowvideo;
writeln('very grid point?');
highvideo;
writeln('2. at '); lowvideo;
writeln('point only plus the slope?');
write('Enter your choice and press <RETURN> '); ReadIKbd,sysp[i,Bedlevel1Mainstrm)l;
if sysp[i,Bedlevel1Mainstrm) in ['e','E','o','O'] then
LegalEntry := True;
if not LegalEntry then writeln('ILLEGAL ENTRY, TRY AGAIN ');
until LegalEntry;
case sysp[i,WidthFloodpln] of
'c','C',
'v',
'V' : begin
writeln;
lowvideo;
writeln('The level of the floodplains is');
write('1. at '); highvideo;
write('C'); lowvideo;
writeln('constant level above the mainstream bedlevel');
write('2. known at '); highvideo;
write('E'); lowvideo;

writeln('very point.');
writeln('3. known at ');
if sysp[i,BedlevelFloodpln] in ['c', 'C', 'e', 'E', 'o', 'O'] then
  LegalEntry := True;
if not LegalEntry then
  writeln('ILLEGAL ENTRY, TRY AGAIN');
until LegalEntry;
end; {of case}
'0' : sysp[i,BedlevelFloodpln] := '0';
end; {of WidthFloodpln case statement}
----------- END OF MENU SECTION -----------

{------------------- DATA INPUT -------------------------
case sysp[i,BedlevelMainstrm] of
  'E',
    'E' : begin
      writeln;
      writeln('Give the bedlevel of the mainstream ');
      writeln('at each point : ');
      for il:=0 to nstep[i] do
        begin
        writeln('point at ',deltaX[i]*il:6:0,' meters ');
        EnterReal(Bedlevel[i,i1,OJ);
        end; {of do-loop}
    end; {of case}
  '0',
    'O' : begin
      writeln;
      writeln('where do you know the mainstream-bedlevel : ');
      EnterReal(Position);
      write('what is the bedlevel at this place ');
      EnterReal(Bedlev);
      write('what is the slope in this branch ');
      EnterReal(Slope);
      for il:=0 to nstep[i] do
        Bedlevel1[i,i1,0] := Bedlev + Slope * (Position - il*deltaX[i]);
    end; {of case}
end; {of BedlevelMainstrm case statement}
case sysp[i,BedlevelFloodpln] of
  'C',
    'C' : begin
      writeln;
      writeln('Give the difference bedlevel floodplain ');
      writeln(' - bedlevel mainstream : ');
      EnterReal(Bedlevel2[i,nstep[i]]); 
      for il:=0 to nstep[i] do
        Bedlevel2[i,i1] := Bedlevel1[i,i1,0] + Bedlevel2[i,nstep[i]]:
    end; {of case}
  'E',
    'E' : begin
      writeln;
      writeln('Give the level of the floodplain at');
      for il:=0 to nstep[i] do
        begin
        writeln(deltaX[i]*il:6:0,' meters ');
        EnterReal(Bedlevel2[i,i1]);
        end; {of do-loop}
    end; {of case}
end; {of case}
0': begin
  writeln;
  lowvideo;
  writeln('where do you know the floodplain-bedlevel: ');
  EnterReal(Position);
  writeln('what is the floodplain-level here: ');
  EnterReal(Bedlev);
  writeln('what is the slope in this branch: ');
  EnterReal(Slope);
  for i:=0 to nstep[i] do
  Bedlevel2[i,i1] := Bedlev + Slope * (Position - i1*deltaX[i1]);
  end; (of case)
end; (of for)

for i:=0 to nstep[i] do
  Bedlevel2[i,i1] := 0;
end; (of BedlevelFloodpln case statement)
(------------------- END OF DATA INPUT ---------------------)

writeln;
highvideo;
repeat
  writeln('ARE YOU SURE ') (YIN: '):
  Read(Kbd,sure);
  writeln;
  until Sure in ['y','Y','n','N'];
until Sure in ['y','Y'];
end; (of do-loop)
end; (of procedure AskBedlevel)

{AskDisch.pas}
{******************************************************************************}
{routine AskDischarge (CREADATA) **********************************************}
{******************************************************************************}

{ asks boundary conditions for water discharge }

procedure AskDischarge( nstep :Integer;
                        deltaT :Real;
                        upstrm :ArrayType2;
                        Dxtern :ArrayType6;
                        sysp :ArrayType7);
var
  LegalEntry : Boolean;
  i,
  periodStep : Byte;
  il : Integer;
  Sure : Char;
  period : Real;
begin
  for i:=1 to nbranch do
  begin
    repeat
      writeln('branch ',i);
      writeln;
      lowvideo;
      writeln('The discharge is: ');
      writeln;
    until
  end; (of procedure AskDischarge)
case upstrm[1,1] of
  0 : begin
    sysp[1,UpstrmBound] := 'u';
    writeln('1. known and ');
    highvideo;
    write('C');
    lowvideo;
    writeln('constant');
    write('2. known as ');
    highvideo;
    write('P');
    lowvideo;
    writeln('periodic function (e.g. yearly discharge curve)');
    highvideo;
    write('3. known as an other ');
    lowvideo;
    writeln('function of time');
  end; {of case}
else
  begin
    if upstrm[1,2]=0 then
      sysp[1,UpstrmBound] := 'i'
    else
      sysp[1,UpstrmBound] := 'c';
    writeln('1. equal to the discharge of the ');
    highvideo;
    write('U');
    lowvideo;
    writeln('stream branch(es)');
    highvideo;
    writeln('2. equal to the discharge of the upstream ');
    highvideo;
    writeln('branch(es)');
    highvideo;
    writeln('minus DeltaQ (water ');
    lowvideo;
    writeln('traction');
  end; {of case}
end; {of upstrm case statement}
writeln;
LegalEntry := False;
repeat
  write('Enter your choice and press <RETURN> :');
  read(Kbd,sysp[1,Discharge]);
  writeln;
  if sysp[1,Discharge] in ['c','C','e','E','f','F',
    'p','P','u','U'] then
    LegalEntry := True;
  if not LegalEntry then
    writeln('ILLEGAL ENTRY, TRY AGAIN : ');
until LegalEntry;
------------------- END OF MENU-SECTION -------------------

------------------- DATA INPUT ---------------------

lowvideo;
writeln;
case sysp[1,UpstrmBound] of
  'u': (upstream boundary of problem )
    'U': begin
      case sysp[1,Discharge] of
        'c': (constant discharge )
        'C': begin
          write('Give the constant discharge : ');
          EnterReal(Qextern[1,0]);
          for ii:=1 to nttstep do
            Qextern[1,ii]:=Qextern[1,0];
          end; {of case}
'p', (periodic discharge curve)

'F' : begin
  write('Give the period (in hours) : '); EnterReal(period);
  period := period * 60 * 60;
  periodStep := round(period/deltaT);
  writeln('Give the discharge at ');
  for i1:=0 to periodStep do begin
    writeln('at ',i1*(deltaT/(60.*60)):10:0);
    writeln(' hours : ');
    EnterReal(Qextern[i,i1]);
  end; {of do-loop}
  for i1:=periodStep to ntstep do begin
    Qextern[i,i1] := Qextern[i,i1-periodStep];
  end; {of do-loop}
end; {of case}

'f', (general discharge curve)

'F' : begin
  write('Give the discharge at ');
  writeln('every timestep : ');
  for i1:=0 to ntstep do begin
    writeln('at ',i1*(deltaT/(60.*60)):10:0);
    writeln(' hours : ');
    EnterReal(Qextern[i,i1]);
  end; {of do-loop}
end; {of case}

'C', (internal boundary or confluence upstream)

'C' : begin
  case sysp[i,Discharge] of
    'E', (water extraction) begin
      write('Give constant delta Q : ');
      EnterReal(Qextern[i,0]);
      for i1:=1 to ntstep do begin
        Qextern[i,i1] := Qextern[i,0];
      end; {of case}
    end; {of case}
  end; {of UpstrmBound case statement}
end; {of case}

{------------------- END OF DATA INPUT -----------------------------}
readvideo;
repeat
  writeln;
  read(Kbd,sure);
  writeln;
  until Sure in ['y','Y','n','N'];
until Sure in ['y','Y'];
end; {of do-loop}
end; {of procedure AskDischarge}
procedure AskTransport( intstep, deltaT : Integer; var Sxtern, sysp : ArrayType6; var LegalEntry, Num: Boolean; Byte; i : Integer; Sure : Char; begin
  for i:=1 to nbranch do begin
    repeat
      clrscr;
      highvideo;
      writeln('branch ',i);
      lowvideo;
      writeln;
      writeln('The upstream boundary condition for the sediment transport: ');
      writeln('transport: ');
      case sysp[i,UpstrmBound] of
        'u', 'U': begin
          writeln('1. Sediment input constant');
          highvideo;
          writeln('2. Sediment input function of time');
          highvideo;
          writeln('3. Sediment input calculated with transport formula');
          highvideo;
          writeln('4. Sediment input equal to sediment input');
          highvideo;
          writeln('0' ) ;
          lowvideo;
          writeln('output at');
          writeln('end of flume (special flume facility)');
        end;
      end;
    end;
  end;
end;
1: begin
   writeln('1. Sediment input equal to sediment output of');
   write('');
   highvideo;
   write('U');
   lowvideo;
   writeln('pstream branch');
   write('2. Constant sediment ');
   highvideo;
   write('E');
   lowvideo;
   writeln('traction at this point');
   write('3. Sediment input equal to sediment ');
   highvideo;
   write('O');
   lowvideo;
   writeln('output at');
   writeln(' end of flume (special flume facility)');
   end;(of case)

'C': begin
   writeln('1. Sediment input equal to sediment output of');
   write('');
   highvideo;
   write('U');
   lowvideo;
   writeln('pstream branches');
   write('2. Constant sediment ');
   highvideo;
   write('E');
   lowvideo;
   writeln('traction at this point');
   end;(of case)
end;(of UpstreamBound case statement)
write;
highvideo;
LegalEntry := False;
repeat
   write('Enter your choice and press <RETURN> : '); read(Kbd,sysp[i,Sediment]); writeln;
   if sysp[i,Sediment] in ['b','B','c','c','e','E',
      'f','F','o','O','t','T','u','U'] then
      LegalEntry := True;
   if not LegalEntry then
      writeln('ILLEGAL ENTRY, TRY AGAIN : ');
   until LegalEntry;

------------------- END OF MENU SECTION -------------------
------------------- DATA INPUT -------------------
case sysp[i,UpstreamBound] of
   'u', ( upstream boundary of problem )
   'U': begin
      case sysp[i,Sediment] of
         'c', ( constant sediment input )
            'C': begin
               writeln('Give the constant sediment input : ');
               EnterReal(Sextern[i,0]);
               for ii:=0 to nstep do
                  Sextern[i,ii] := Sextern[i,0];
               end;(of case)
         'F', ( sediment input function of time )
            'F': begin
               writeln('Give the sediment input at every');
               writeln(' time step : ');
               for ii:=0 to nstep do
                  writeln('at ',deltaT*ii/(60.*60):10:0);
                  write(' hours : ');
                  EnterReal(Sextern[i,ii]);
               end;(of do-loop)
            end; (of case)
"b", "B", "t", (bedlevel constant / transport formula)
'T': for i1 := 0 to nstep do
  Sextern[i1, i1] := 0;
end; (of Sediment case statement)
end; (of case)

'i', "I", 'c', (internal boundary or confluence upstream)
'C': begin
  case sysp[i, Sediment] of
    'u', (no sediment extraction)
      'U': for i1 := 0 to nstep do
        Sextern[i1, i1] := 0;
      end; (of case)
    'e', (sediment extraction)
      'E': begin
        write('Give the constant sediment extraction:');
        EnterReal(Sextern[i1, 0]);
        for i1 := 1 to nstep do
          Sextern[i1, i1] := Sextern[i1, 0];
        end; (of case)
      end; (of Sediment case statement)
    end; (of case)
end; (of UpstrBound case statement)
}

------------------- END OF DATA INPUT -----------------------------

writeln;

repeat
write( 'ARE YOU SURE? (Y/N) ');
readln(Kbd, sure);
writeln;
until Sure in ['y', 'Y', 'n', 'N'];
until Sure in ['y', 'Y'];
end; (of do-loop)
end; (of procedure AskTransport)

{II: Waterv.pas}

(*--------------------------------------------------------------*)
(* routine AskWaterlevel (CREADATA)                           *)
(* asks the downstream boundary condition for the waterlevel  *)

procedure AskWaterlevel(var nstep: Integer;
                        var deltaT: Real;
                        var dostrm: ArrayType1;
                        var upstrm: ArrayType2;
                        var hConst, hFact, hExp: ArrayType3;
                        var hExtern, Dextern: ArrayType6;
                        var sysp: ArrayType7)

var
  LegalEntry: Boolean;
  i1, i2: Byte;
  i1: Integer;
  Sure: Char;

begin
  for i := 1 to nbranch do
    begin
      repeat
        clrscr;
        highvideo;
        writeln('branch ', i);
        writeln;
        lowvideo;
        writeln('The downstream waterlevel is :');
        writeln;
        repeat
          write('ARE YOU SURE? (Y/N) ');
          readln(Kbd, sure);
          writeln;
        until Sure in ['y', 'Y', 'n', 'N'];
        until Sure in ['y', 'Y'];
      end; (of do-loop)
    end; (of procedure AskTransport)
case dostrm[i] of
  0 : begin
    sysp[i,DownstrmBound] := 'd';
    write1. known and ');
    highvideo;
    write('C');
    lowvideo;
    writeln('onstant');
    write2. known via ');
    highvideo;
    write('G');
    lowvideo;
    writeln('eneral Q-h curve');
    write3. known via h = hConst + hFact * ');
    highvideo;
    write('D');
    lowvideo;
    writeln(' in hE:p');
  end; (of case)
else begin
  if upstrm(dostrm[i],2)=0 then
    sysp[i,DownstrmBound] := 'i'
  else
    sysp[i,DownstrmBound] := 'c';
  highvideo;
  write('1. unknown');
  lowvideo;
  writeln('qual to the upstream waterlevel of the ');
  writeln('downstream branch');
  write2. known via ');
  highvideo;
  write('G');
  lowvideo;
  writeln('eneral Q-h curve');
  write3. known via h = hConst + hFact * ');
  highvideo;
  write('Q');
  lowvideo;
  writeln(' in hE:p');
  end; (of case)
end; (of dostrm case-statement)
writeln;
highvideo;
LegalEntry := False;
repeat
  write('Enter your choice and press <RETURN>: ');
  read(Kbd,sysp[i,Waterlevel]);
  writeln;
  if sysp[i,Waterlevel] in ['c','C','e','E','g','G',
    'q','Q'] then
    LegalEntry := True;
  if not LegalEntry then
    writeln('ILLEGAL ENTRY, TRY AGAIN : ');
  until LegalEntry;
{------------------- END OF MENU SECTION -----------------------------}
case sysp[i, Waterlevel] of
  'C' : ( constant waterlevel downstream )
    begin
      write('Give the constant waterlevel : '); 
      EnterReal(hExtern[i,0]);
      for il:=1 to ntstep do
        hExtern[i,il] := hExtern[i,0];
    end; (of case)
  'G' : ( Q-h curve downstream )
    begin
      for il:=0 to ntstep do
        Qextern[i,il] := 0;
        write('For how many values of Q do you want to give the waterlevel '); 
        highvideo;
        repeat
          readln(i2);
          InputCheck(LegalEntry);
        until LegalEntry;
        writeln;
        for il:=1 to i2 do
          begin
            write('Q ',il:3,' I; EnterReal (Qe:i1); 
            write(' h ',il:3,' I; EnterReal (hE>:tern[i,i I];
          end; (of do-loop)
        end; (of case)
  'Q' : ( Q-h function downstream )
    begin
      write('Give hConst : ');
      EnterReal(hConst[i]);
      write('Give hFact 'I;
      EnterReal(hFact[i]);
      write('Give hExp 'I;
      EnterReal(hExp[i]);
    end; (of case)
end; (of case)
  'i', 'I', 'c', ( internal boundary or confluence downstream )
  'C' : ( constant waterlevel downstream )
    begin
      case sysp[i, Waterlevel] of
        'G' : ( Q-h curve downstream )
          begin
            for il:=0 to ntstep do
              Qextern[i,il] := 0;
            write('For how many values of Q do you want to give the waterlevel ');
            highvideo;
            repeat
              readln(i2);
              InputCheck(LegalEntry);
            until LegalEntry;
            writeln;
            for il:=1 to i2 do
              begin
                write('Q ',il:3,' I; EnterReal (Qe:i1); 
                write(' h ',il:3,' I; EnterReal (hE>:tern[i,i I];
              end; (of do-loop)
            end; (of case)
  'Q' : ( Q-h function downstream )
    begin
      write('Give hConst : ');
    end; (of case)
EnterReal(hConst[i]);
write('Give hFact : ');
EnterReal(hFact[i]);
write('Give hExp : ');
EnterReal(hExp[i]);
end; (of case)
end; (of Waterlevel case statement)
end; (of case)
end; (of DownstrBound case statement)
------------------- END OF DATA INPUT ---------------------------
writeln;
highvideo;
repeat
  write('ARE YOU SURE ? (Y/N) ');
  read(kbd,sure);
  writeln;
  until Sure in ['y','Y','n','N'];
  until Sure in ['y','Y'];
end; (of do-loop)
end; (of procedure AskWaterlevel)

(* AskRough.pas)
(*--------------------------------------------------------------------------)
(* routine AskRoughness (CREADATA) *)
(* asks roughness of each riverbranch *)
(*--------------------------------------------------------------------------*)
procedure AskRoughness( nstep :ArrayType1;
  deltaX :ArrayType3;
  var Roughness1, Roughness2 :ArrayType4;
  var sykp :ArrayType7) :

  var
    LegalEntry : Boolean;
    i, i1 : Byte;
    Sure : Char;

  begin
    for i:=1 to nbranch do
      begin
        repeat
          ------------------- CREATION OF MENU -------------------------------
          highvideo;
          clrscr;
          writeln('branch ',i);
          writeln;
          lowvideo;
          writeln('The friction-term in the backwater calculation should');
          writeln('be calculated with : ');
          highvideo;
          write('a. ');
          lowvideo;
          writeln('constant Chezy value and the waterdepth');
          highvideo;
          write('b. ');
          lowvideo;
          writeln('variable Chezy value per grid point and the waterdepth');
          highvideo;
          write('c. ');
          lowvideo;
          writeln('constant Chezy value and R = a.w / (2.a + w)');
          highvideo;
          write('d. ');
          lowvideo;
          writeln('variable Chezy value and R = a.w / (2.a + w)');
          highvideo;
          write('e. ');
          lowvideo;
          writeln;
        until
          writeln:
          highvideo;
          writeln:
          highvideo;
          writeln:
          highvideo;
          writeln:
          highvideo;
          writeln:
          highvideo;
          writeln:
          highvideo;
          writeln:
          highvideo;
          writeln:
          highvideo;
          writeln:
          highvideo;
          writeln:
          highvideo;
          writeln:
          highvideo;
          writeln:
          highvideo;
          writeln:
          highvideo;
          writeln:
          highvideo;
          writeln:
        until
writeln('constant Kn value and the waterdepth');
highvideo;
write('f.
');
lowvideo;
writeln('variable Kn value per grid point and the waterdepth');
highvideo;
write('g.
');
lowvideo;

writeln('constant Kn value and $R = a_w / (2.a + w)$');
highvideo;
write('h.
');
lowvideo;
writeln('variable Kn value and $R = a_w / (2.a + w)$');
highvideo;
writeln('e, f, g and h option : $C = 18\log_{10}(2a/Kn)$');
write('i.
');
lowvideo;
writeln('Van Rijn roughness predictor (see documentation)');
highvideo;
writeln('j.
');
lowvideo;
writeln('Van Rijn roughness predictor (see documentation)');
highvideo;
writeln('Hydraulic radius : waterdepth');
write('k.
');
lowvideo;
writeln('Hydraulic radius : Einstein');
write('l.
');
低video;
writeln('j option not compatible with floodplains!');
write('m.
');
lowvideo;
LegalEntry := False;
repeat
  write('Enter your choice and press <RETURN> : '); read(kbd,sysp[i,Roughness]);
  writeln;
  if sysp[i,Roughness] in ['a'..'A','b'..'B','c'..'C','d'..'D','e'..'E','f'..'F','g'..'G','h'..'H','i'..'I','j'..'J'] then
    LegalEntry := True;
  if not LegalEntry then
    writeln('ILLEGAL ENTRY, TRY AGAIN : ');
until LegalEntry;

------------------- END OF MENU SECTION --------------------------

------------------- DATA INPUT ----------------------------------

writeln;
lowvideo;
case sysp[i,WidthFloodpln] of
  '0': begin { no floodplains }
    case sysp[i,Roughness] of
      'a'..'A','c'..'C','e'..'E','g'..'G', (roughness constant)
        'G': begin
          write('Give the constant roughness of the mainstream : ');
          EnterReal(Roughness1[i,0]);
          Roughness2[i,0] := 0;
          for i1:=1 to nstep[i] do
            begin
              Roughness1[i,i1] := Roughness1[i,0];
              Roughness2[i,i1] := 0;
            end;
        end;
      'b'..'B','d'..'D','f'..'F','h'..'H', (roughness variable)
        'H': begin
          write('Give the mainstream roughness at each point : ');
          writeln('each point : ');
          for i1:=0 to nstep[i] do
            begin
              end;
    end;
end;
}
write('point at ', deltaX[i]*il:6:0, ' meters');
EnterReal(Roughness1[i, il]);
Roughness2[i, il] := 0;
end;(of do-loop)
end;(of case)

'1':
begin
for il:=0 to nstep[i] do
Roughness1[i, il] := 0;
for il:=0 to nstep[i] do
Roughness2[i, il] := 0;
end;(of case)

'2':
begin
writeln('Roughness predictor uses : ');
writeln('a. estimated Chezy value');
writeln('b. wall roughness');
writeln('c. grain diameter D50');
writeln('d. grain diameter D90');
writeln;
write('Give the estimated Chezy value ');
EnterReal(Roughness1[i, il]);
for il:=1 to nstep[i] do
Roughness1[i, il] := Roughness1[i, 0];
writeln('Give the wall roughness ');
EnterReal(Roughness2[i, il]);
for il:=1 to nstep[i] do
Roughness2[i, il] := Roughness2[i, 0];
writeln;
writeln('The grain diameter will be asked later');
end;(of case)
end;(of case)

'3', 'C', 'V', (with floodplains )
'4':
begin
  case sysp[i, Roughness] of
    'a', 'A', 'c', 'C', 'e', 'E', 'g', (roughness constant )
    'G':
      begin
write('Give the constant roughness of the ');
write('mainstream : ');
EnterReal(Roughness1[i, 0]);
write('Give the constant roughness of the ');
write('floodplain : ');
EnterReal(Roughness2[i, 0]);
for il:=1 to nstep[i] do
begin
  Roughness1[i, il] := Roughness1[i, 0];
  Roughness2[i, il] := Roughness2[i, 0];
end;(of do-loop)
end;(of case)

    'b', 'B', 'd', 'D', 'f', 'F', 'h', (roughness variable)
    'H':
      begin
write('Give the roughness at each point : ');
for il:=0 to nstep[i] do
begin
write('point at ', deltaX[i]*il:6:0);
write(' meters ');
write('(mainstream) : ');
EnterReal(Roughness1[i, il]);
write('
');
write('(floodplain) : ');
EnterReal(Roughness2[i, il]);
end;(of do-loop)
end;(of case)
I': begin
  for i1:=0 to nstep[i] do
    Roughness1[i,i1] := 0;
end; {of case}
J:
begin
  highvideo;
  writeln('OPTION NOT POSSIBLE');
  write('roughness predictor not compatible ');
  writeln('with floodplains');
end; {of case}
end; {of case}
end; {of case}
end; {of WidthFloodpln case statement}
------------------- END OF DATA INPUT -----------------------
writeln;
highvideo;
repeat
  write('ARE YOU SURE ?> (Y/N) ');
  read(Kbd,sure);
  writeln;
  until Sure in ['V', 'Y', 'N'];
end; {of do-loop}
- end; {of procedure AskRoughness}

---

Asking the grain size in each branch

procedure AskGrainsize(nstep : Arraytype1;
deltaX : Arraytype3;
var d50, d90 : Arraytype4;
var sysp : Arraytype7);

var
  LegalEntry : Boolean;
  i, i1 : Byte;
  Sure : Char;

begin
  for i:=1 to nbranch do
    begin
      repeat
        clrscr;
        highvideo;
        writeln('branch ',i);
        lowvideo;
        writeln;
        writeln('The D50 grainsize is ');
        write('1. ');
        highvideo;
        write('C');
        lowvideo;
        writeln('constant');
        write('2. '); highvideo;
        write("'\"V'"); lowvideo;
        writeln('variable per grid point');
        writeln;
        highvideo;
        LegalEntry := False;}
repeat
    write('Enter your choice and press <RETURN> : ');
    read(Kbd,sysp[i,Grain]);
    writeln;
    if sysp[i,Grain] in ['c', 'C', 'v', 'V'] then
        LegalEntry := True;
    if not LegalEntry then
        writeln('ILLEGAL ENTRY, TRY AGAIN : ');
    until LegalEntry;

{------------------- END OF MENU SECTION -----------------------}

{------------------- DATA INPUT -------------------------------}

{------------------- CREATION OF MENU ----------------------------}

{------------------- DATA INPUT ----------------------------------}


repeat
    write('Enter your choice and press <RETURN> : ');
    read(Kbd,sysp[i,Grain]);
    writeln;
    if sysp[i,Grain] in ['c', 'C', 'v', 'V'] then
        LegalEntry := True;
    if not LegalEntry then
        writeln('ILLEGAL ENTRY, TRY AGAIN : ');
    until LegalEntry;

{------------------- END OF MENU SECTION -------------------------}

{------------------- DATA INPUT ----------------------------------}

{------------------- CREATION OF MENU ----------------------------}

{------------------- DATA INPUT ----------------------------------}


for il:=1 to nstep[i] do
  d90[i,il] := d90[i,0];
end; (of case)

'V', (grainsize variable per grid point)
'V' begin
  writeln('Give the grain diameter at each point (D90):');
  for il:=0 to nstep[i] do
    begin
      write('point at ',deltaX[i]*il:6:0, 'meters.');
      EnterReal(d90[i,il]);
    end; (of do-loop)
end; (of case)
end; (of case)
end; (of Grain case statement)
(-------------------- END OF DATA INPUT ----------------------)

writeln;
highvideo;
repeat
  write('ARE YOU SURE? (Y/N) ');
  read(kbd,sure);
  writeln;
  until Sure in ['y', 'Y', 'n', 'N'];
  until Sure in ['y', 'Y'];
end; (of do-loop)
end; (of procedure AskGrainsize)

(* AskFormul.pas)
- { ****************************************** }
- (routine AskFormula (CREADATA) )
- { ****************************************** }
- ( asks which transport formula is to be used )

procedure AskFormula(var mtr, ntr, ucr
  var sysp
  :ArrayType3;
  :ArrayType7);

var
  LegalEntry : Boolean;
  i : Byte;
  Sure : Char;

begin
  for i:=1 to nbranch do
    begin
      repeat
        ------------------- CREATION OF MENU -------------------------
        clrscr;
        highvideo;
        writeln('branch ',i);
        writeln;
        lowvideo;
        writeln('The used transport formula is:');
        writeln('1. ');
        highvideo;
        write('P');
        lowvideo;
        writeln('power law s = m . u^n');
        writeln('2. ');
        highvideo;
        write('A');
        lowvideo;
        writeln('cikers and White formula');
        writeln('3. ');
        highvideo;
        write('M');
        lowvideo;
        writeln('eyer-Feter and Mueller formula');
        read(kbd,sure);
        writeln('Are you sure? (Y/N)');
        writeln;
        until Sure in ['y', 'Y'];
      end; (of do-loop)
  end; (of procedure AskGrainsize)
write('Enter your choice and press <RETURN> :');
read(Kbd,symp[i,Formula]);
write;
if symp[i,Formula] in ['a','A','e','E','m','M','p','P'] then
  LegalEntry := True;
if not LegalEntry then
  writeln('ILLEGAL ENTRY, TRY AGAIN :');
until LegalEntry;
--------------------------------------------------------- END OF MENU SECTION ---------------------------------------------------------
--------------------------------------------------------- DATA INPUT ---------------------------------------------------------
write;
lowvideo;
case symp[i,Formula] of
  'p': ( power law )
  begin
    write('Give m of formula :');
    EnterReal(mtr[i]);
    write('Give n of formula :');
    EnterReal(ntr[i]);
    ucr[i] := 0;
  end; (of case)
  'm': ( meyer-peter mueller formula )
  begin
    write('Give MPMcl (see documentation) of formula :');
    EnterReal(mtr[i]);
    write('Give MPMc2 (see documentation) of formula :');
    EnterReal(ucr[i]);
    ntr[i] := 0;
  end; (of case)
  'A': begin
    writeln('All information is resident in the program');
    writeln('Check routine 'SeditrnsalCalc' if necessary.');
    writeln('Transport will be calculated as: ');
    writeln('S = S(A&W) * multiplication factor');
    write('The multiplication factor is: ');
    EnterReal(mtr[i]);
    ntr[i] := 0;
    ucr[i] := 0;
  end; (of case)
  'E': begin
    writeln('All information is resident in the program');
    writeln('Check routine 'SeditrnsalCalc' if necessary.');
    writeln('Transport will be calculated as: ');
    writeln('S = S(E&H) * multiplication factor');
    write('The multiplication factor is: ');
    EnterReal(mtr[i]);
    ntr[i] := 0;
    ucr[i] := 0;
  end; (of case)
end; (of Formula case statement)
--------------------------------------------------------- END OF DATA INPUT ---------------------------------------------------------
write;
highvideo;
repeat
  write('ARE YOU SURE (Y/N) ');
  read(Kbd,sure);
  writeln;
  until sure in ['Y','y','N','n'];
until sure in ['Y','y','N','n'];
end; (of do-loop)
procedure AskSource(var seds : ArrayType3;
               var sysp : ArrayType7);

var
  LegalEntry : Boolean;
  i : Byte;
  Sure : Char;

begin
  for i:=1 to nbranch do begin
    repeat
      clrscr;
      highvideo;
      writeln('branch ',i);
      lowvideo;
      writeln;
      writeln('This branch has');
      writeln('1. '); highvideo;
      writeln('2. A');
      highvideo;
      writeln('o external sediment source');
      writeln('2. A');
      lowvideo;
      writeln('constant external sediment source');
      writeln;
      highvideo;
      LegalEntry := False;
      repeat
        write('Enter your choice and press <RETURN> : ');
        read(kbd,sysp[i,Source]);
        writeln;
        if sysp[i,Source] in ['c', 'C', 'n', 'N'] then begin
          LegalEntry := True;
          if not LegalEntry then writeln('ILLEGAL ENTRY, TRY AGAIN : ');
        until LegalEntry;
    end;{of do-loop}
  end;{of for loop}
end;{of procedure AskSource}
program WhatNext; 
var 
  NextFrog : file;

begin 
  clrscr; 
  lowvideo; 
  gotoXY(21,10); 
  writeln('The input of data is now completed.'); 
  gotoXY(21,12); 
  writeln('The name of the created datafile is :'); 
  gotoXY(34,14); 
  highvideo; 
  writeln(OutputDataname); 
  gotoXY(21,24); 
  writeln('press any key to return to main menu'); 
  repeat until KeyPressed; 
  assign(INextFrog, 'MENU.COM'); 
  execute(NextFrog); 
end; {of procedure Whatnext} 

{---------------------- END OF INCLUDED FILES ----------------------} 

{********************************************************} 
{ MAIN BLOCK CREADATA } 
{********************************************************} 

begin 
  InitCrea(OutputDataname, nbranch, ntstep, tmax, deltaT, 
  nstep, dostrm, upstrm, length, deltaX, tOutput); 
  AskWidth(ntstep, deltaX, width1, width2, widthS, sysp); 
  AskBedlevel(ntstep, deltaX, Bedlevel1, Bedlevel2, sysp); 
  AskDischarge(ntstep, deltaT, upstrm, Qextern, sysp); 
  AskTransport(ntstep, deltaT, Sextern, sysp); 
  AskWaterlevel(ntstep, deltaT, dostrm, upstrm, hConst, hFact, hExp, 
  hExtern, Sextern, sysp); 
  AskRoughness(ntstep, deltaX, Roughness1, Roughness2, sysp); 
  AskGrainsize(ntstep, deltaX, d50, d90, sysp); 
  AskFormula(mtr, ntr, ucr, sysp); 
  AskSource(seds,sysp); 
  WriteDataToFile(nbranch, ntstep, deltaT, tmax, 
  nstep, dostrm, upstrm, length, deltaX, 
  ntr, mtr, ucr, seds, hConst, hFact, hExp, width1, 
  width2, widthS, Bedlevel1, Roughness1, Roughness2, 
  d50, d90, Bedlevel1, hExtern, Sextern, Sextern, 
  sysp, tOutput, OutputDataname, datafile); 
WhatNext(OutputDataname); 
end. {of program}
Listing of ODIRMO.PAS, 00:32am 11/08/85

{*******************************************************************
.....*
.....*
,.,.Jó'
**
*******
""*Jó'****
*******
****
*******....***
*******************************************************************>

Ro D RRM. Ciermeer
august 1985
v. 1.01

Delft University of Technology
Department of Civil Engineering
Section Fluid Mechanics

This is a program for one dimensional river morphology.
The most important limitations are:
1. No bifurcations
2. Quasi steady flow
3. Uniform grain size
4. Transport depends on local conditions only
transportformula

before running this program, a datafile must be created
with the program CREADATA.

-------------------------------------------------------------------

This is a program for one dimensional river morphology.
The most important limitations are:
1. No bifurcations
2. Quasi steady flow
3. Uniform grain size
4. Transport depends on local conditions only
transportformula

before running this program, a datafile must be created
with the program CREADATA.

-------------------------------------------------------------------

-- DECLARATION SECTION --

type

component = (WidthMainstrm,WidthFloodpln,WidthTransport,
BedlevelMainstrm,BedlevelFloodpln,
UpstrmBound,Discharge,Sediment,DownstrmBound,
Waterlevel,Roughness,Grain,Formula,Source);

const

branchmax = 6;
stepmax = 90;
timestepmax = 100;

type

filename = string[8];
ArrayType1 = array[1 ..branchmax] of Byte;
ArrayType2 = array[1 ..branchmax,1 ..2] of Byte;
ArrayType3 = array[1 ..branchmax] of Real;
ArrayType4 = array[1 ..branchmax,0 ..stepmax] of Real;
ArrayType5 = array[1 ..branchmax,0 ..stepmax,0 ..1] of Real;
ArrayType6 = array[1 ..branchmax,0 ..timestepmax] of Real;
ArrayType7 = array[1 ..branchmax,WidthMainstrm..Source] of Char;
ArrayType8 = array[1 ..6] of integer;

var

i,
iter,
nbranch : Byte;
t,
ntstep : Integer;
deltaT,
tmax : Real;
dostrm,
ntep : ArrayType1;
upstrm : ArrayType2;
deltaX, length, hConst, hFact, hExp, q, mtr, ntr, ucr,
seds : ArrayType3;
Roughness1, Roughness2, d50, d90, h, r, Rd, width1, width2, widthS,
Bedlevel2 : ArrayType4;
S, Bedlevel1 : ArrayType5;
hExternal, Sextern, Dextern : ArrayType6;
sysp : ArrayType7;
tOutput : ArrayType8;
datafile : text;
NextProg : file;
InputDatanname, OutputDatanname : filename;

procedure Results(iter, t :Integer;
q, h, width1 :ArrayType3;
S, Bedlevel1 :ArrayType4);

i, l, k : Byte;
a : Real;

begin
if iter=1 then k:=0 else k:=1;
write(lst,'The results after ',t,' timesteps (');
writeln(lst,t * deltaT / 3600:8:2,' hours) :');
writeln(lst);
write(lst, 'place DISCHARGE WATERLEVEL TRANSPORT');
writeln(lst,'BEDLEVEL DEPTH VELOCITY FRAC');
write(lst,' M M-3/s M M-3/day');
writeln(lst,' M M/s -');
writeln(lst);
writeln(lst);
for i:=1 to nbranch do
begin
  for j:=0 to nstep[i] do
  begin
    write(lst,i:2,i+deltaX[i]:7:0,q[i]:10:2,h[i,i+1]:12:4);
    write(lst,S[i,i+1]:24*60*60:11:4,Bedlevel1[i,i+1,0]:
      a:=h[i,i+1]-Bedlevel1[i,i+1,0];
    write(lst,a:8:4);
    write(lst,r[i,i+1]*q[i]/(a*width1[i,i+1,10:3]);
    writeln(lst,s[i,i+1,11]*24*60*60:11:4,Bedlevel^{i,i+1,0});
    a:=h[i,i+1]-Bedlevel1[i,i+1,0];
    write(lst,a:8:4);
    write(lst,r[i,i+1]*q[i]/(a*...
var
  i   : Byte;
  i1  : Integer;
  i2  : Component;
begin
  assign(datafile, InputDatename);
  reset(datafile);
  readln(datafile, nbranch, tmax, ntstep, deltaT);
  for i := 1 to 6 do
  begin
    for i := 1 to nbranch do
    begin
      readln(datafile, tOutput[i]);
    end;
  end;
  for i := 1 to nbranch do
  begin
    for i2 := WidthMainstrm to Source do
    begin
      read(datafile, sysp[i, i2]);
      readln(datafile);
    end;
  end;
  for i := 1 to nbranch do
  begin
    for il := 0 to ntstep do
    begin
      read(datafile, Extern[i, il], Sextern[i, il], hExtern[i, il]);
      close(datafile);
    end;
  end;
end;
procedure DischargeCalc(var q: ArrayType3;
                        nbranch: Byte;
                        t: Integer;
                        upstrm: ArrayType2;
                        Oextern: ArrayType6;
                        sysp: ArrayType7);

var
   i: byte;

begin
   for i:=1 to nbranch do
     begin
       case sysp[i,UpstreamBound] of
         'U',
         'u': q[i] := Oextern[i,t];
         'I',
         'i':
           begin
             case sysp[i,Discharge] of
               'U',
               'u': q[i] := q[upstrm[i,1]];
               'E',
               'e': q[i] := q[upstrm[i,1]] - Oextern[i,t];
             end;
           end;
         end;
       case sysp[i,Discharge] of
         'C',
         'c': begin
           case sysp[i,Discharge] of
             'U',
             'u': q[i] := q[upstrm[i,1]] + q[upstrm[i,2]];
             'E',
             'e': q[i] := q[upstrm[i,1]] + q[upstrm[i,2]] - Oextern[i,t];
           end;
         end;
       end;
     end;
end;
procedure WaterlevelDownstream(input, nstep, dostrm, hConst, hFact, hExp: ArrayType1);
    var h: ArrayType2;
    hExtern: ArrayType3;
    sysp: ArrayType4;
    h: ArrayType5;
    i: Byte;

    begin
        case sysp[i, DownstreamBound] of
            'd', (downstream boundary of problem)
                'D': begin
                    case sysp[i, Waterlevel] of
                        'c', (Waterlevel constant)
                            'C': hi, nstep[i] := hExtern[i, t];
                        'g', (Waterlevel from Q-h curve)
                            'G': begin
                                i1 := 0;
                                repeat
                                    i1 := i1 + 1;
                                    hi, nstep[i1] := hExtern[i, i1] +
                                    (hExtern[i, i1] - hExtern[i, i1 - 1]) *%
                                    (hExtern[i, i1] - hExtern[i, i1 - 1]) *%
                                    (hExtern[i, i1] - hExtern[i, i1 - 1]);
                                end; {of case}
                        'q', (Waterlevel function of Q)
                            'Q': hi, nstep[i] := hConst[i] + hFact[i]
                                * exp( hExp[i] * ln(q[i]));
                    end; {of Waterlevel case statement}
                end; {of case}

            'c', (internal boundary or confluence in problem)
                'C': begin
                    case sysp[i, Waterlevel] of
                        'e', (Waterlevel equal waterlevel of downstream branch)
                            'E': hi, nstep[i] := h[dostrm[i], 0];
                        'g', (Waterlevel from Q-h curve)
                            'G': begin
                                i1 := 0;
                                repeat
                                    i1 := i1 + 1;
                                    hi, nstep[i1] := hExtern[i, i1] +
                                    (hExtern[i, i1] - hExtern[i, i1 - 1]) *%
                                    (hExtern[i, i1] - hExtern[i, i1 - 1]) *%
                                    (hExtern[i, i1] - hExtern[i, i1 - 1]);
                                end; {of case}
                        'q', (Waterlevel function of Q)
                            'Q': hi, nstep[i] := hConst[i] + hFact[i]
                                * exp( hExp[i] * q[i]);
                    end; {of Waterlevel case statement}
                end; {of case}
        end; {of DownstreamBound case statement}
    end; {of procedure WaterlevelDownstream}
function RoughCalc (ODIRMO)

{ calculates Chezy roughness at a point in a branch }

function RoughTerm( i, i1 )

{ Rough1, Rough2, Rb

D50, D90, width

al

sysp

:Byte;

:ArrayType4;

:ArrayType4;

:Real;

:ArrayType7) : Real;

var

Rough1, Rough2, Rb

D50, D90, width

al

sysp

:8yte;

:ArrayType4;

:ArrayType4;

:Real;

:ArrayType7)

begin

case sysp[i,Roughness] of

'a', 'A', 'b',

B : begin

Rb[i,i1] := al;

RoughTerm := 1/(SOR(Rough1[i,i1]) * Rb[i,i1]);

end; (of case)

'c', 'C', 'd',

D : begin

Rb[i,i1] := (al * width[i,i1]) / (2*al + width[i,i1]);

RoughTerm := 1 / (SOR(Rough1[i,i1]) * Rb[i,i1]);

end; (of case)

'e', 'E', 'f',

F : begin

C := 18*ln(12*al/Rough1[i,i1])/log10;

Rb[i,i1] := al;

RoughTerm := 1 / (SOR(C) * Rb[i,i1]);

end; (of case)

'g', 'G', 'h',

H : begin

C := 18*ln(12*al/Rough1[i,i1])/log10;

Rb[i,i1] := (al * width[i,i1]) / (2*al + width[i,i1]);

RoughTerm := 1 / (SOR(C) * Rb[i,i1]);

end; (of case)
\begin{verbatim}
I : begin
  ( roughness predictor : van Rijn (1982)
    hydraulic radius : waterdepth )

  (dimensionless grainsize )
  Dster := DS0(i,i1) * exp((1/3) * ln(16.187 / SQR(nu)) );
  (calculation of Shield's critical shear velocity )
  if (Dster>0) and (Dster<=4) then
    UstCr2 := 16.187 * DS0(i,i1) * 0.24 / Dster;
  if (Dster>4) and (Dster<=10) then
    UstCr2 := 16.187 * DS0(i,i1) * 0.14 * exp(-0.64*ln(Dster));
  if (Dster>10) and (Dster<=20) then
    UstCr2 := 16.187 * DS0(i,i1) * 0.04 * exp(-0.10*ln(Dster));
  if (Dster>20) and (Dster<=150) then
    UstCr2 := 16.187 * DS0(i,i1) * 0.013 * exp(0.29*ln(Dster));
  if (Dster>150) then
    UstCr2 := 0.055;

  (Chezy value related to grainsize )
  C := 18 * ln(12 * RSb(i,i1) / (3 * D90(i,i1))) / log10;
  (calculation of transport stage parameter )
  Tc := (UstAc2 - UstCr2) / UstCr2;
  (calculation of dune height )
  H := 0.11 * a1 * exp(0.3 * ln(DS0(i,i1) / a1)) * (1 - exp(-0.5 * Tc)) * (25 - Tc);
  (calculation of dune length )
  L := 7.3 * a1;
  (effective roughness )
  Ks := 3 * D90(i,i1) + 1.1 * H * (1 - exp(-25 * H / L));
  (Chezy value )
  C := 18 * ln(12 * RSb(i,i1) / Ks) / log10;
  Rough[i,i1] := C;
  Roughterm := 1 / (sqr(C) * Rsb[i,i1]);
end;(of case)

J : begin
  (roughness predictor : van Rijn (1982)
    hydraulic radius : Einstein (1948) )

  (dimensionless grainsize )
  Dster := DS0(i,i1) * exp((1/3) * ln(16.187 / SQR(nu)) );
  (calculation of Shield's critical shear velocity )
  if (Dster>0) and (Dster<=4) then
    UstCr2 := 16.187 * DS0(i,i1) * 0.24 / Dster;
  if (Dster>4) and (Dster<=10) then
    UstCr2 := 16.187 * DS0(i,i1) * 0.14 * exp(-0.64*ln(Dster));
  if (Dster>10) and (Dster<=20) then
    UstCr2 := 16.187 * DS0(i,i1) * 0.04 * exp(-0.10*ln(Dster));
  if (Dster>20) and (Dster<=150) then
    UstCr2 := 16.187 * DS0(i,i1) * 0.013 * exp(0.29*ln(Dster));
  if (Dster>150) then
    UstCr2 := 0.055;

  (Chezy value )
  C := 18 * ln(12 * Rb(i,i1)) / (3 * D90(i,i1)) / log10;
  (calculation of transport stage parameter )
  Tc := (UstAc2 - UstCr2) / UstCr2;
  (calculation of dune height )
  H := 0.11 * a1 * exp(0.3 * ln(DS0(i,i1) / a1)) * (1 - exp(-0.5 * Tc)) * (25 - Tc);
  (calculation of dune length )
  L := 7.3 * a1;
  (effective roughness )
  Ks := 3 * D90(i,i1) + 1.1 * H * (1 - exp(-25 * H / L));
  (Chezy value )
  C := 18 * ln(12 * Rb(i,i1) / Ks) / log10;
  Rough[i,i1] := C;
  Roughterm := 1 / (sqr(C) * Rb(i,i1));
end;(of case)
\end{verbatim}
repeat
  hardstop2 := hardstop2 + 1;
  Rw := (Rwp1 + Rwp2)/2;
  Rbi[i, i1] := al * (1 - 2 * Rw / width[i, i1]);
  slope := sq2U / (SOR(Cpl) * Rbi[i, i1]);
  delW := 11.6 * nu / SQR(g * Rw * slope);
  Cw := 18 * ln(12*Rw/(Rough2[i, i1]+0.5*delW))/log10;
  Rwp1 := Rbi[i, i1] * SQR(Cpl / Cw);
  Rbi[i, i1] := al * (1 - 2 * Rwp1 / width[i, i1]);
  slope := sq2U / (SOR(Cpl) * Rbi[i, i1]);
  delW := 11.6 * nu / SQR(g * Rwp1 * slope);
  Cw := 18 * ln(12*Rwp1/(Rough2[i, i1]+0.5*delW))/log10;
  Rwp2 := Rbi[i, i1] * SQR(Cpl / Cw);
  until (ABS(Rw - Rwp2) < epsR) or (hardstop2 > 10);
Rbi[i, i1] := al * (1 - (Rwp1 + Rwp2) / width[i, i1]);
{Chezy value related to grainsize :}
Cpl := 18 * ln(Rwp1 / (D90[i, i1] / al));
{calculation of shear velocity :}
UstAc2 := (UstAc2 - UstCr2) / UstCr2;
{calculation of dimensionless transport stage parameter :}
Tc := (UstAc2 - UstCr2) / UstCr2;
{calculation of dune height :}
H := 0.11 * al * exp(0.3 * ln(D50[i, i1] / al)) *
(1 - exp(-0.5 * Tc)) * (25 - Tc);
{calculation of dune length :}
L := 7.3 * al;
{effective roughness :}
Ks := 3 * D90[i, i1] + 1.1 * H * (1 - exp(-25 * H / L));
{Chezy value :}
Cp2 := 18 * ln(Rwp1 / (Rough2[i, i1]+0.5*delW))/log10;
{calculation of dimensionless transport stage parameter :}
Tc := (UstAc2 - UstCr2) / UstCr2;
{calculation of dune height :}
H := 0.11 * al * exp(0.3 * ln(D50[i, i1] / al)) *
(1 - exp(-0.5 * Tc)) * (25 - Tc);
{calculation of dune length :}
L := 7.3 * al;
{effective roughness :}
Ks := 3 * D90[i, i1] + 1.1 * H * (1 - exp(-25 * H / L));
{Chezy value :}
Cp2 := 18 * ln(Rwp1 / (Rough2[i, i1]+0.5*delW))/log10;
{calculation of dimensionless transport stage parameter :}
Tc := (UstAc2 - UstCr2) / UstCr2;
{calculation of dune height :}
H := 0.11 * al * exp(0.3 * ln(D50[i, i1] / al)) *
(1 - exp(-0.5 * Tc)) * (25 - Tc);
{calculation of dune length :}
L := 7.3 * al;
{effective roughness :}
Ks := 3 * D90[i, i1] + 1.1 * H * (1 - exp(-25 * H / L));
{Chezy value :}
Cp2 := 18 * ln(Rwp1 / (Rough2[i, i1]+0.5*delW))/log10;
until (ABS(Cp2-Cpl) < epsC) or (hardstop1 > 10);
C := (Cpl + Cp2) / 2;
Rough[i, i1] := C;
Roughterm := 1 / (SQR(C) * Rbi[i, i1]);
end; {of case}
end; {of case-statement}
end; {of procedure RoughCalc}
procedure BackwaterCalc( iter : Byte; 
nstex : ArrayType1; 
DeltaX : ArrayType2; 
var Roughness1, Roughness2 : ArrayType3; 
Bedlevel1, D50, D90 : ArrayType4; 
var Rb, h, r : ArrayType5; 
Endlevel1, sysp : ArrayType6; 
)

begin
if iter=1 then k:=0 else k:=1;
case sysp[i,WidthFloodpln] of

c, 'C', 'e', 'E', 'v', 
'V' : begin
for i:=nsstep[i] downto 1 do
begin

(** waterdepth mainstream at i : **)
a1 := h[i,i] - Bedlevel3[i,i,k];

(** waterdepth floodplain at i : **)
a2 := h[i,i] - Bedlevel2[i,i];

(** if waterlevel below floodplain-bedlevel : **)
if a2<0 then a2:=0;

(** friction term floodplain at i : **)
Frict2 := RoughTerm(i,i,Roughness2,Roughness1,Rb,D50, D90, width2,a2,sysp);

(** friction term mainstream at i : **)
Frict1 := RoughTerm(i,i,Roughness1,Roughness2,Rb,D50, D90, width1,a1,sysp);

(** term due to change of mainstream-width at i : **)
dwdx1 := ((1/deltaX[i]) * g * width1[i,i]) * 
(width1[i,i] - width1[i,i-1]);

(** term due to change of floodplain-width at i : **)
dwdx2 := ((1/deltaX[i]) * g * width2[i,i]) * 
(width2[i,i] - width2[i,i-1]);
if i=1 then
begin
(** terms due to change of width at place 0 : **)
dwdx3 := dwdx1;
dwdx4 := dwdx2;

end

end
end
end
end
end
end
begin

end; (of case)

end; (of do-loop)
end; (of do-loop)

end; (of else statement)

end; (of do-loop)

end; (of case)

else

begin

end; (of else statement)

end; (of do-loop)

end; (of case)
0: begin
  r[1,0] := 1;
  for i:=nstep[1] downto 1 do
    begin
      r[i,1] := 1;
      (waterdepth at i := #)
      a1 := h[i,1] - Bedlevel[i,1,k];
      (friction term place i := #)
      Frict1 := RoughTerm[i,1,Roughness1,Roughness2,Rb,D50,
        D90,width1,a1,symp);
      (term due to change of riverwidth place i := #)
      dwdx1 := (1/(deltaX[i]) * g * width1[i,1]} * 
        (width1[i,1] - width1[i,1-1]);
      if i=1 then
        begin
          dwdx3 := dwdx1;
        end
      else
        begin
          dwdx3 := (1/(deltaX[i] * g * width1[i,1-1]} * 
            (width1[i,1-1] - width1[i,1-2]);
          Fact1 := SQRT(q[i]/(width1[i,1] * a1));
          dummy1 := Fact1 * (dwdx1 - Frict1);
          h[i,1-1] := h[i,1] - deltaX[i] * dummy1;
          for i:=1 to 2 do
            begin
              (waterdepth at i-1 := #)
              a3 := h[i,1-1] - Bedlevel[i,1-1,k];
              (friction term at i-1 := #)
              Frict3 := RoughTerm[i,1-1,Roughness1,Roughness2,Rb,
                D50,D90,width1,a3,symp);
              Fact3 := SQRT(q[i]/(width1[i,1-1] * a3));
              (term due to change of waterdepth := #)
              dadx1 := (1/(g * a1 * deltaX[i]) * (a1 - a3));
              dummy1 := Fact1 * (dadx1 + dwdx1 - Frict1);
              dummy3 := Fact3 * (dwdx3 - Frict3);
              h[i,1-1] := h[i,1] - (deltaX[i]/2) * 
                (dummy1 + dummy3);
            end; (of do-loop)
          end; (of do-loop)
        end; (of case)
    end; (of WidthFloodpln case-statement)
  end; (of procedure BackwaterCalc)
procedure SediTransCalc(iter, iter : Byte;
    nstep : ArrayType1;
    delta, q, mtr, ntr,
    ucr : ArrayType3;
    h, width1, widthS, r,
    D50, D90, Roughness1,
    Rb : ArrayType4
    var S
    Bedlevel1 : ArrayType5;
    sysp : ArrayType7;
    begin
    if iter = 1 then k := 0 else k := 1;
    for il := 0 to nstep[i] do
    begin
    begin
    u := q[i]*r[i, i]/(width1[i, i, il]*h[i, i, il]-Bedlevel1[i, i, il, k]);
    end
    {select transport formula :}
    case sysp[i, il, Formula] of
    'a': begin
    Dgr := D50[i, i, il] * exp(1/3 * ln(g * delta / (nu * nu)));
    if Dgr > 60 then
    begin
    n := 0;
    A := 0.17;
    m := 1.5;
    Cgr := 0.025;
    end
    else
    begin
    n := 1 - 0.56 * ln(Dgr) / log10;
    A := 0.14 + 0.23 / SORT(Dgr);
    m := 1.34 + 9.66 / Dgr;
    Dgr := ln(Dgr) / log10;
    Dgr := 2.86 + Dgr - SORT(Dgr) - 3.53;
    Cgr := exp(Dgr * log10);
    end;
    Uster := SORT(g) * u / Roughness1[i, il];
    Fgr := u * (1 / SORT(g * delta * D50[i, i, il]) * exp(n*ln(Uster/u)) * exp(n-1) * ln(SORT(32) *
    ln(10 + (h[i, i, il] - Bedlevel1[i, i, il, k]) / D50[i, i, il]) / log10));
    S[i, i, il, k] := mtr[i] * widthS[i, i, il] * Cgr / 0.6 * u *
    D50[i, i, il] * exp(n * ln(u/Uster)) *
    exp(m * ln(1 + Fgr / A));
    end; {of case}
begin
  delA := 11.6 * nu * Roughness1[i,i] * 
    (h[i,i] - Bedlevel1[i,i,k]) / (SORT(g) * u);
  Cgr := 18 * ln(12 * Rb[i,i] / (D50[i,i] + 0.3 * delA)) / log10;
  Mu := exp(1.5 * ln(Roughness1[i,i] / Cgr));
  T := u * u / (delta * D50[i,i] * SOR(Roughness1[i,i]));
  if Mu*T > ucr[i] then
    S[i,i,k] := SQRT(delta * g * exp(3 * ln(D50[i,i]))) * 
      mtr[i] * exp(1.5 * ln(Mu * T - ucr[i])) * 
      widthS[i,i];
  else
    S[i,i,k] := 0;
  end; {of case}
  'E': begin
    Uster := SQRT(g) * u / Roughness1[i,i];
    n := SQRT(g * delta * D50[i,i]);
    m := D50[i,i] / n;
    S[i,i,k] := mtr[i] * widthS[i,i] * 0.084 * m * 
      SOR(u / n) * SOR(Uster / n) * (Uster / n);
  end; {of case}
end; {of procedure Formula case-statement}
end; {of do-loop}
end; {of procedure SediTransCalc}

procedure SediTransUpstrm(Iter, t, nstep upstrfTi var S:te-nte-nsysp:Byte; :Integer: :A.-rayTypel: ArrayTypel2: ArrayType5: ArrayType6: ArrayType7: 
var 
begin
  if iter=1 then k:=0 else k:=1;
  case sysp[i,Upstr-bound] of 
    'u', (upstream boundary of problem : )
    'U': begin
      case sysp[i,Sediment] of
        'c', 'C', 'f', (known sediment input )
        'F': S[i,0,k] := Sextern[i,t];
        'b', (bedlevel constant )
        'B': S[i,1,k] := S[i,1,k];
        'o',
        'O': S[i,0,k] := S[i,nstep[i],k];
      end; {of Sediment case statement}
    end; {of case of SediTransUpstrm}
'i', (internal boundary :)
'I': begin
  case sysp[i,Sediment] of
    'u', (transport equal to transport upstream branch)
      U': S[i,0,k] := S[upstrm[i,1],nstep[upstrm[i,1]],k];
    'e', (sediment extraction)
      E': S[i,0,k] := S[upstrm[i,1],nstep[upstrm[i,1]],k]
        - Sextern[i,t];
    'o',
      'O': S[i,0,k] := S[i,nstep[i],k];
  end; (of Sediment case statement)
end; (of case)
end; (of procedure SediTransUpstrm)

'c', (confluence :)
'C': begin
  case sysp[i,Sediment] of
    'u', (transport equal to transport upstream branches)
      U': S[i,0,k] := S[upstrm[i,1],nstep[upstrm[i,1]],k]
        + S[upstrm[i,2],nstep[upstrm[i,2]],k];
    'e', (sediment extraction)
      E': S[i,0,k] := S[upstrm[i,1],nstep[upstrm[i,1]],k]
        + S[upstrm[i,2],nstep[upstrm[i,2]],k]
        - Sextern[i,t];
  end; (of Sediment case statement)
end; (of case)
end; (of procedure SediTransUpstrm)
procedure BedlevelCalc( iter, i :Byte; 
nstep :ArrayType1; 
deltaX :ArrayType3; 
widthl :ArrayType4; 
S :ArrayType5; 
var Bedlevel1 :ArrayType5);

const
teta = 0.65;

var
il : byte;

begin
  { prepare transport for predictor step : }
  if iter=1 then
  begin
    for nil:=0 to nstep[i] do 
      S[i,nil,1]:=S[i,nil,0];
  end;

  { upstream scheme : }
  Bedlevel1[i,0,1] := Bedlevel1[i,0,0] - (deltat/deltaX[i]) * 
  (teta * (S[i,1,1] - S[i,0,1]) + (1 - teta) * (S[i,1,0] - 
  S[i,0,0])) / widthl[i,1];

  { inside scheme : }
  for nil:=1 to nstep[i]-1 do
    Bedlevel1[i,nil,1] := Bedlevel1[i,nil,0] - (deltat / 
    (2 * deltaX[i])) * (teta * (S[i,nil+1,1] - S[i,nil-1,1]) + 
    (1-teta) * (S[i,nil+1,0] - S[i,nil-1,0]))/widthl[i,nil];

  { downstream scheme : }
  Bedlevel1[i,nstep[i],1] := Bedlevel1[i,nstep[i],0] - 
  (deltat/deltaX[i]) * (teta * (S[i,nstep[i],1] - 
  S[i,nstep[i]-1,1]) + (1-teta) * (S[i,nstep[i],0] - 
  S[i,nstep[i]-1,0])) / widthl[i,nstep[i]]; 

  { prepare bedlevel for next timestep : }
  if iter=3 then
  begin
    for nil:=0 to nstep[i] do 
      Bedlevel1[i,nil,0] := Bedlevel1[i,nil,1] + seds[i] * deltaT;
  end; (of condition)
end; (of procedure BedlevelCalc)
(* Writ~Dat.pas *)

***************

routine WriteDataToFile (UDIRMO, CREADATA)

***************

procedure WriteDataToFile(nbranch, ntstep, deltaT, tmax, nstep, dostrm,
upstrm, length, deltaX, ntr, mtr, ucr, sedS, hConst,
hFact, hExp, width1, width2, widthS,
Bedlevel2, Roughness1, Roughness2, d50, d90,
Bedlevel1, hExtern, Dextern, Sextern,
sySp, tOutput, OutputDataName)

var datafile

var i, i1, i2 : Byte;

var i1 : Integer;

var i2 : Component;

var print : Char;

begin

clrscr;

------------------- START WRITING TO PRINTER ------

writeln(lst);

writeln(lst,'CONTENTS OF DATAFILE :');

writeln(lst);

writeln(lst,'nbranch : Byte;');

writeln(lst,'ntstep : Integer;');

writeln(lst,'deltaT, tmax : Real;');

writeln(lst,'nstep, dostrm : ArrayType1;');

writeln(lst,'upstrm, length, deltaX, ntr, mtr, ucr, sedS, hConst,');

writeln(lst,'hFact, hExp : ArrayType2;');

writeln(lst,'width1, width2, widthS,');

writeln(lst,'Bedlevel2, Roughness1, Roughness2, d50, d90,');

writeln(lst,'Bedlevel1, hExtern, Dextern, Sextern, sySp, tOutput,');

writeln(lst,'OutputDataName : filename);');

for i := 1 to 6 do

writeln(lst,'Output[i]:10, 'timesteps');

writeln(lst);

for i := 1 to nbranch do

writeln(lst,'length[i]:8 :0);';

writeln(lst,'deltaX[i]:9:1, nstep[i]:8:0);');

writeln(lst,'dostrm[i]:8:0);');

end; {of do-loop}

writeln(lst,'hFact, hExp');

writeln(lst,'upstrm2, upstrm2, ntr, mtr, ucr, sedS, hConst');

writeln(lst,'Width1, Width2, WidthS,');

writeln(lst,'Bedlevel1, BedlevelF, UpstrmBnd, Discharge');

for i := 1 to nbranch do

writeln(lst,'The system parameters are : ');

writeln(lst);

writeln(lst,'WidthM, WidthF, WidthT');

writeln(lst,'Bedlevel1, BedlevelF, UpstrmBnd, Discharge');

writeln(lst);
for i:=1 to nbranch do
begin
  write(lst, i:2);
  for i2:=WidthMainstrm to Discharge do
  begin
    writeln(lst, sysp[i, i2]:10);
  end; (of do-loop)
  writeln(lst);
  write(lst, 'nr Sediment DownstrmBd Waterlevl ');
  writeln(lst, 'Roughness Grain Formula Source' );
  writeln(lst);
end; {af do'-loop}
writeln(lst); 
write(lst, 'nr place WIDTH1 WIDTH2 WIDTHS BEDL1 BEDL2 ');
writeln(lst, 'ROUGH1 ROUGH2 D50 D90');
writeln(lst);
for i:=1 to nbranch do
begin
  for i2:=Infil to Source do
  begin
    write(lst, i:2, i2:2, sysp[i, i2]:10);
  end; (of do-loop)
  writeln(lst);
end; (of do-loop)
write(lst);
writeln(lst, 'branch time Q-extern S-extern h-extern');
writeln(lst, 'hours m'3/s m'3/s m');
writeln(lst);
if ntstep > 50 then
begin
  hi=highvideo;
  writeln('There are ', ntstep:4, ' timesteps. ');
  repeat
    writeln('Do you want a print of every timestep ?(Y/N) : ');
    read(Kbd, print);
    writeln;
    until print in ['y', 'Y', 'n', 'N'];
end;
if (print in ['y', 'Y']) or (ntstep <= 50) then
begin
for i:=1 to nbranch do
begin
  for i1:=0 to ntstep do
  begin
    write(lst, i:2, i1:2, deltaX[i1]:6:0, width1[i, i1]:8:1);
    write(lst, width2[i, i1]:8:1, widthS[i, i1]:8:1);
    write(lst, Bedlevel1[i, i1]:7:3, Bedlevel2[i, i1]:7:3);
    write(lst, roughness1[i, i1]:8:3, roughness2[i, i1]:8:3);
    writeln(lst, d50[i, i1]:7:4, d90[i, i1]:7:4);
  end; (of do-loop)
  writeln(lst);
end; (of do-loop)
write(lst);
writeln(lst, 'END OF DATAFILE');
writeln(lst);
writeln(lst);
(------------------- WRITING TO DATAFILE -------------------------)
assign(datafile, OutputDatename);
rewriteln(datafile);
writeln(datafile, nbranch:4, tmax, ntstep:4, deltaT);
for i:=1 to 6 do
  writeln(datafile, tOutput[i]);
for i:=1 to nbranch do
  begin
    writeln(datafile, length[i], deltaX[i], nstep[i]:4, dostrm[i]:4);
    writeln(datafile, upstrm[i], upstrm[i, 2:4];
    writeln(datafile, ntr[i], mtr[i], ucr[i]);
    writeln(datafile, seds[i], hConst[i], hFact[i], hExp[i]);
  end; (of do-loop)
for i:=1 to nbranch do
  begin
    for i2:=WidthMainstrm to Source do
      writeln(datafile, sysp[i, i2]);
  end; (of do-loop)
for i:=1 to nbranch do
  begin
    for il:=0 to ntstep do
      writeln(datafile, Bedlevel1[i, i1], Bedlevel2[i, i1]);
    writeln(datafile, Roughness1[i, i1], Roughness2[i, i1]);
    writeln(datafile, d50[i, i1], d90[i, i1]);
  end; (of do-loop)
for i:=1 to nbranch do
  begin
    for i:=0 to ntstep do
      writeln(datafile, Oextern[i], Sextern[i], hExtern[i, i1]);
  end; (of do-loop)
close(datafile);
end; (of procedure WriteDataToFile)

(------------------- END WRITING TO DATAFILE -------------------------)

(---------------------------- END OF INCLUDED FILES -------------------------)

(begin)
InitDirim(inputDatename, outputDatename);
ReadDataFromFile(nbranch, ntstep, deltaT, tmax, nstep, dostrm, upstrm, length, deltaX, ntr, mtr, ucr, seds, hConst, hFact, hExp, width1, width2, widthS, Bedlevel1, Roughness1, Roughness2, d50, d90, Bedlevel1, hExtern, Oextern, Sextern, sysp, tOutput, InputDatename, datafile);
crlscr;
for t:=0 to ntstep do
begin
  ---------------- INFORMATION ON SCREEN -----------------
  gotoXY(20, 10);
  write('I am working on timestep number ');
  gotoXY(35, 12);
  writeln(t);
  gotoXY(20, 14);
  DischargeCalc(q, nbranch, t, upstrm, Oextern, sysp);

for iter:=1 to 3 do begin

------------- BACKWATER CURVE CALCULATION ---------------
for i:=nbranch downto 1 do begin
  WaterlevelDownstream(i, t, nstep, dostrm, hConst, hFact, hExp, h, hExtern, sysp);
  BackwaterCalc(iter, nstep, deltaX, width1, width2, Roughness1, Roughness2, Bedlevel1, d50, d90, Rb, h, r, Bedlevel1, sysp);
end; {of do-loop}

------------- TRANSPORT CALCULATION ---------------------
for i:=1 to nbranch do begin
  SediTransCalc(iter, i, nstep, deltaX, q, mtr, ntr, ucr, h, width1, widthS, r, D50, D90, Roughness1, Rb, S, Bedlevel1, sysp);
  SediTransUpstrm(iter, i, t, nstep, upstrm, S, Sextern, sysp);
end; {of do-loop}

---------- OUTPUT TO PRINTER -----------------------------
if ((t in [tOutput[1], tOutput[2], tOutput[3], tOutput[4], tOutput[5], tOutput[6]] AND (iter=1)) then
begin
  results(iter, t, q, h, width1, S, Bedlevel1);
end; {of condition}

---------- NEW BEDLEVEL CALCULATION ----------------------
if t<nstep then begin
for i:=1 to nbranch do
  BedlevelCalc(iter, i, nstep, deltaX, width1, S, Bedlevel1);
end; {of condition}

end; {of do-loop (iteration)}
end; {of do-loop (timesteps)}

---------- RESULTS TO PRINTER AND NEW DATAFILE ----------
WriteDataToFile(nbranch, nstep, deltaT, tmax, ntr, mtr, ucr, seds, hConst, hFact, hExp, width1, width2, widthS, Bedlevel1, Roughness1, Roughness2, d50, d90, Bedlevel1, hExtern, Qexternal, Sexternal, sysp, tOutput, OutputDataName, datafile);

assign(NextProg, 'MENU.COM');
execute(NextProg);

end. {of program ODIRMO}
Part V : Test Cases

Contents:

1. Introduction  
2. Test cases
   2.1 T1  : overloading  
   2.2 T2  : overloading  
   2.3 T3  : overloading  
   2.4 T4  : overloading  
   2.5 T5  : overloading  
   2.6 T6  : underloading  
   2.7 T7  : floodplain  
   2.8 T8  : floodplain  
   2.9 T9  : discontinuity in bedlevel  
   2.10 T10 : river bend cut-off  
   2.11 T11 : confluence + weir  
part V: Test Cases

1. Introduction

To verify the validity of the mathematical- and computer-model some test cases are calculated. They are numbered T1 to T10.

T1:
Small sediment overload in a riversection (small timescale). The results are compared with RIVMOR calculations. This case is taken from the report "Aggradation in rivers due to overloading" by Ribberink and v.d. Sande, in which the RIVMOR calculations are compared with analytical approaches. In this case the major difference between RIVMOR and ODIRMO becomes very obvious. RIVMOR produces a much steeper wave-front than ODIRMO does. And ODIRMO gives some secondary waves behind the front.

T2:
As T1, only this time a large sediment overloading. The conclusions are the same.

T3:
As T1, only this time a large time scale. In this case there is almost no difference between ODIRMO and RIVMOR results.

T4:
As T2, only this time a large time scale. No difference between RIVMOR and ODIRMO calculations.

T5:
Sediment overloading. Calculations with ODIRMO were performed until the new equilibrium was reached.

T6:
As T5, but now with a sediment underloading.

T7:
Calculations with a floodplain. A dredged pit in the floodplain. ODIRMO crashed on this problem.

T8:
Calculations with a floodplain. A dredged trench in the mainstream. ODIRMO crashed on this problem too.

T9:
Calculations with a discontinuity in the initial bedlevel. Three timesteps are carried out to compare the calculation of the flow profile and the development of the sedimentation wave front with RIVMOR.

T10:
River bend cut-off. The results are given to indicate what kind of results a user might expect for his own case. The calculations are performed with two different place-steps. This may give an expression of the accuracy and the numerical diffusion.

T11:
Confluence of two rivers with a weir in the downstream section. As case T9, the calculations are not compared with other calculations.

2. Test Cases

2.1 T1: overloading

\[
\begin{align*}
i &= 10^{-4} \\
a &= 3.0 \text{ m} \\
u &= 1.0 \text{ m/s} \\
s &= 2.2 \times 10^3 \text{ m}^2/\text{s} \\
C &= 57.74 \text{ m}^{(1/2)}/\text{s}
\end{align*}
\]

fig. 2.1.1 Initial situation
for T1, T2, T3, T4, T5 and T6

T1 initial situation (see fig. 2.1.1 also):

length : 2200 m.
transport formula : \( s = 2.2 \times 10^{-3} u^5 \text{ m}^2/\text{s} \)
timestep : 1 hour (and 3 hour)
placestep : 50 m.
sediment input : \( 2.42 \times 10^{-3} \text{ m}^2/\text{s} \)

The results of the calculations are given in fig. 2.1.2. It appears that the sedimentation wave-front is much steeper with RIVMOR calculations. For \( t = 1 \) hour and \( x = 50 \) meters the Courant number is approximately:

\[
\sigma = \frac{n \times s \times \Delta t}{a \Delta x} = \frac{5 \times 2.42 \times 10^{-3} \times 3600}{2.94 \times 50} = 0.30
\]

This is a rather small Courant number, as it is expected that ODINRMO gives the best results for Courant numbers.
part V: Test Cases

about 1. For this reason the calculations with ODIRMO are repeated with $t = 3$ hours, so the Courant number about 1.2. It is seen that the secondary waves behind the wave-front tend to disappear for increasing Courant number. It is also seen though, that the steepness of the wave-front becomes even less pronounced. This is due to the increased numerical diffusion.
fig. 2.1.2 T1 results

CASE T1

ΔZ_b(cm)

ΔT=1 h

ΔT=3 h

ODIRM0

RIVMOR

ΔS/S_0 = 0.1

43 h

12.9 h

original bedlevel

X (m)
part V: Test Cases

**fig. 2.2.1 T2 results**
2.2 T2: overloading

T2 initial situation (see also fig. 2.1.1):

length : 2200 m.
transport formula : \( s = 2.2 \times 10^{-3} \text{ m}^2/\text{s} \)
timestep : 1 hour
place step : 50 m.
sediment input : 6.6 \( \times 10^{-3} \text{ m}^2/\text{s} \)

The results of the calculations are given in fig. 2.2.1. The conclusion are the same as for T1. The Courant number in this case is approximately:

\[
\sigma = \frac{n \times s \times \Delta t}{\Delta x} = \frac{5 \times 6.6 \times 10^{-3} \times 3600}{2.41 \times 50} = 0.99
\]

(2.2.1)

This Courant number is already close to 1, so the calculations are not repeated with a bigger timestep. It is also seen in this case, that the propagation velocity of the RIVMOR wave-front is slightly higher than the velocity of the ODIRMO wave-front.
2.3 T3: overloading

T3 initial situation (see also fig. 2.1.1):

- **length**: 150 km.
- **transport formula**: \( s = 2.2 \times 10^{-3} \text{ m}^2 \text{/s} \)
- **timestep**: 30 hours
- **place step**: 1500 m.
- **sediment input**: \( 2.42 \times 10^{-3} \text{ m}^3 \text{/s} \)

The results of the calculations are given in fig. 2.3.1. The results of the ODIRMO and RIVMOR calculations are almost identical. The Courant number in this case is approximately:

\[
\sigma = \frac{n \times s \times \Delta t}{a \Delta x} = \frac{5 \times 2.42 \times 10^{-3} \times 108000}{2.94 \times 1500} = 0.30 \quad (2.3.1)
\]

It appears, that the relative small Courant number is not of much influence in this case.

2.4 T4: overloading

T4 initial situation (see also fig. 2.1.1):

- **length**: 150 km.
- **transport formula**: \( s = 2.2 \times 10^{-3} \text{ m}^2 \text{/s} \)
- **timestep**: 30 hours
- **place step**: 1500 m.
- **sediment input**: \( 6.6 \times 10^{-3} \text{ m}^3 \text{/s} \)

The results of the calculations are given in fig. 2.4.1. The results of the ODIRMO and RIVMOR calculations are almost identical. The Courant number in this case is approximately:

\[
\sigma = \frac{n \times s \times \Delta t}{a \Delta x} = \frac{5 \times 6.60 \times 10^{-3} \times 108000}{2.41 \times 1500} = 0.99 \quad (2.4.1)
\]

Case T3 and T4 show, that for large timescales ODIRMO gives (practically) the same results as RIVMOR.
part V: Test Cases

fig. 2.3.1 T3 results
CASE T4

fig. 2.4.1 T4 results
part V : Test Cases

2.5 T5 : overloading

T5 initial situation (see also fig. 2.1.1):

- length : 1500 m.
- transport formula : \( s = 2.2 \times 10^{-3} u^5 m^2/s \)
- timestep : 1 hour
- place step : 50 m.
- sediment input : \( 2.42 \times 10^{-3} m^2/s \)

The calculations are performed for a situation identical to T1. Only the considered length has been diminished in order to reach the new equilibrium sooner. The results are given in fig. 2.5.1. It can be concluded, that ODIRMO approaches the new equilibrium in a neat way. The new equilibrium can be determined in the following way:

1) new equilibrium transport
   \[ s = 2.42 \times 10^{-3} m^2/s \]
2) equilibrium flow velocity from transport formula
   \[ u = \left(\frac{2.42}{2.2}\right)^{1/5} = 1.02 m/s \]
3) new water depth
   \[ a = \frac{q}{u} = 2.94 m. \]
4) new bed level slope with Chezy equation
   \[ i = \frac{u}{A^{2/3} a} = 1.05 \times 10^{-4} \]

2.6 T6 : underloading

T6 initial situation (see also fig. 2.1.1):

- length : 1500 m.
- transport formula : \( s = 2.2 \times 10^{-3} u^5 m^2/s \)
- timestep : 1 hour
- place step : 50 m.
- sediment input : \( 1.80 \times 10^{-3} m^2/s \)

The results are given in fig. 2.6.1. Due to the underload at the upstream boundary, an erosion wave propagates in the considered section. In case of an erosion wave, the steepness of the wave front is much less than for a sedimentation wave. This is due to the fact, that the propagation velocity just before the front is higher than just after the front in this case. It can be concluded in this case too, that ODIRMO approaches the new equilibrium in a neat way.
part V: Test Cases

**CASE T5**

*Original bedlevel*

*New equilibrium*

fig. 2.4.1 T5 results

**CASE T6**

*Original bedlevel*

*New equilibrium*

fig. 2.5.1 T6 results

- 12 -
part V: Test Cases

2.7 T7: floodplain

In fig. 2.7.1 the initial situation is given of an experiment of the Delft Hydraulics Laboratory in a small flume with composite cross section. There are no results of the calculations, as ODIRMO crashed in the calculation of the flow profile. In fig. 2.7.1 the position of the crash is indicated. The reason for this failure can be found in the way the distribution of the discharge between mainstream and floodplain is calculated. As is stated in Part III, this is done with a quadratic equation. For the considered case the solution of this equation becomes complex. This is due to the fact that, at the indicated place, for one of the two channels the convective term in the equation exceeds the friction term. Also the basic idea behind the calculation of the discharge distribution \((\frac{h}{x}) = (\frac{h}{x})\) is not valid here because of the exchange of momentum between the two channels.

The user should keep this well in mind. Calculation with floodplains generally cause no trouble, but in case of steep gradients in the bedlevel in one of the two channels, a program crash can occur.

2.8 T8: floodplain

For this case the same remarks as for case T7 can be made. In fig. 2.8.1 the position of the program crash is indicated.
fig. 2.7.1 Initial situation T7
fig. 2.8.1 Initial situation T8
2.9 T9: discontinuity in bedlevel

In fig. 2.9.1 the situation is given for a case in a flume of the Delft Hydraulics Laboratory. Three timesteps are carried out to compare the ODIRMO and RIVMOR results.

data:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>11.5 m.</td>
</tr>
<tr>
<td>transportformula</td>
<td>$s = 2.9 \times 10^{-4} u^5 \ m^2/s$</td>
</tr>
<tr>
<td>timestep</td>
<td>2 min.</td>
</tr>
<tr>
<td>placestep</td>
<td>0.1 m.</td>
</tr>
<tr>
<td>sediment input</td>
<td>closed circuit</td>
</tr>
<tr>
<td>Chezy</td>
<td>32 m$^{(1/2)}/s$</td>
</tr>
<tr>
<td>discharge</td>
<td>0.0088 m$^{3}$/s</td>
</tr>
<tr>
<td>width</td>
<td>0.2 m.</td>
</tr>
</tbody>
</table>

It appears, that there is a difference in the calculation of the flow profile. If it is assumed that the RIVMOR results are accurate the ODIRMO results seem unacceptable. In this case too, the calculation of the flow profile appears to be the weak-spot in ODIRMO.
CASE T9

flow profile \( t = 0 \)

\( t = 6 \text{ min (3 steps)} \)

original bedlevel

RIVMOR

ODIRMO

\( Z_b (\text{cm}) \)

\( X (\text{m}) \)
In fig. 2.10.1 the situation is given of a river bend cut-off.

![River bend cut-off](image)

**fig. 2.10.1 River bend cut-off**

The data of the three branches are:

**branch 1:**
- length: 10 km.
- $\Delta x$: 500 m.
- width: 100 m.
- discharge: $500 \text{ m}^3/\text{s}$

**branch 2:**
- length: 4 km.
- $\Delta x$: 500 m.
- width: 75 m.

**branch 3:**
- length: 10 km.
- $\Delta x$: 500 m. (250 m.)
- width: 100 m.
- waterlevel: 5 m.

And for all branches:

- Chezy: $44.72 \text{ m}^{(1/2)}/\text{s}$
- $\Delta T$: 192 hours
- transport: $s = 1 \times 10^{-4} u^5$

In fig. 2.10.2 the results after 80 and 240 days are given. The sedimentation wave after the cut-off does not look too good. When the calculations are repeated with $x = 250 \text{ m.}$ for branch 3 the results improve considerably as the Courant number is now closer to one.
fig. 2.10.2 T10 results
2.11 Til: confluence

In fig. 2.11.1 the situation of a confluence is given. On t=0 a weir is positioned in the downstream branch at the indicated point. Before t=0 the river was in equilibrium.

![Confluence with weir](image)

The data of the four branches are:

branch 1:
- length: 10 km.
- deltaX: 500 m.
- width: 80 m.
- discharge: 418.26 m³/s
- Chezy: 27 m^(1/2)/s
- transport: 8.39 10^-5 u^5 m^2/s

branch 2:
- length: 10 km.
- deltaX: 500 m.
- width: 40 m.
- discharge: 177.82 m³/s
- Chezy: 28.12 m^(1/2)/s
- transport: 8 10^-5 u^5 m^2/s

branch 3:
- length: 5 km.
- deltaX: 500 m.
- width: 119.21 m.
- Chezy: 44.72 m^(1/2)/s
- waterlevel: function of discharge
- transport: 10^-4 u^5 m^2/s

branch 4:
- length: 5 km.
- deltaX: 500 m.
- width: 119.21 m.
- Chezy: 44.72 m^(1/2)/s
- waterlevel: 5 m.
- transport: 10^-4 u^5 m^2/s
At the position of the weir a Q-h relation is given. As the discharge is constant in this case, this means a constant waterlevel. In the results is seen, that a degradation of the bed occurs downstream of the weir. This is a result from the fact, that the transport upstream of the weir is strongly diminished.

Upstream of the weir an aggradation should occur. It seems that in this case aggradation almost only occurs in branch 1 and branch 2.
Fig 2.11.2  T11 results
Part VI: User Guide

Contents

1. Input of data
   1.1 General remarks
   1.2 General data
   1.3 Width
   1.4 Bedlevel
   1.5 Discharge
   1.6 Sediment Transport
   1.7 Waterlevel
   1.8 Roughness
   1.9 Grainsize
   1.10 Transport Formula
   1.11 Source
   1.12 Whatnext

2. Software Remarks
   2.1 Contents of software package
   2.2 Change the source program
   2.3 Transmit data to mainframe
1. Input of data

1.1 General remarks

A session with ODIRMO is always preceded by a session with CREADATA. This program produces menu options and questions on the screen. In this chapter all questions and menu's will be discussed. The user should get sufficient information from this chapter to schematize his problem.

Starting a terminal session on a personal computer which can run under MS-DOS or PC-DOS:

1) Insert the diskette marked Odirmo-Code files in drive A.
2) Power-on.
3) A menu will appear on the screen.
4) Press the number of your choice.

If your choice is:

1) the program CREADATA will be loaded into memory and executed. See Part IV 'Discussion of the program CREADATA'.
2) the program EDITDATA should be loaded into memory and executed, but this program does not exist yet.
3) the program ODIRMO will be loaded into memory and executed. See Part IV 'Discussion of the program ODIRMO'.
4) the program will be stopped and the DOS-prompt appears. You can restart the program by typing MENU.

The program CREADATA creates a data file for the program ODIRMO. The type of the file is the standard Pascal file type TEXT. This means that the file is in ASCII format and can be transmitted to another computer if necessary. This will be discussed in chapter 2.
1.2 General data

This routine asks from the user some general information about his river-problem. The following questions appear subsequently on the screen:

Question:
What is the name of your datafile.
Answer:
Any legal file name followed by <RETURN>. In MS-DOS or PC-DOS this means a word of no more than 8 characters and an optional extension of no more than 3 characters. The two parts separated by a period.
Legal filenames are for instance:
  datafile.dat
  File.ex
  FileData
  File
Illegal are filenames such as:
  datafile1.dat (more than 8 char.)
  FILE.data (extension too long)

Question:
The number of riverbranches is:
Answer:
An integer number in the range 1-6. If you try to enter more than 6 branches the message 'Too many branches!' will appear on the screen. You will have to change the constant BranchMax in the source-program. Due to the fact that TURBO-PASCAL can only handle 64 kilobyte of data, the number of branches, the number of spatial-steps in a branch and the number of timesteps are limited to respectively 6, 90 and 100. It is possible to change these values, only if the total amount of memory used does not increase. When you change these values, you should remember to change them in the ODIRMO-source too! ODIRMO uses more memory to store data than CREADATA does, so you should compile this program first to see if your combination fits into memory. For the procedure to change these values, consult the chapter 2.

Question:
What is the time (in hours) of the calculations:
Answer:
The time of calculations in hours. In the program this value will be recalculated in seconds.
  one day = 24 hours
  one year (365 days) = 8760 hours
  one month (1/12 year) = 730 hours

Question:
Part VI: User Guide

What is the timestep (in hours) of the calculations:
Answer:
The timestep (deltaT) of the calculations. The program will now calculate the number of timesteps (ntstep) and when this number is higher than 100 the message 'Too many timesteps' 'CHANGE SETTING OF TIMESTEPMAX FIRST' will appear on the screen. (See question 'What is the number of riverbranches')

Question:
After how many timesteps do you want output of results (tOutput)?
Answer:
a maximum of 6 output stages is possible. Enter integer numbers to indicate after how many timesteps you want to see results of the calculations.

Question:
What is the length of branch i:
Answer:
The length of branch i in meters.

Question:
What is the spatial-step for branch i:
Answer:
The spatial-step of branch i (deltaX(i)) in meters. The computer calculates the number of spatial steps in this branch now. If this number is higher than 90 the message 'Too many spatial steps!' 'CHANGE SETTING OF STEPMAX FIRST' will appear on the screen. (See question 'What is the number of riverbranches')

Question:
What is the number of the downstream branch, give 0 when no downstream branch present:
Answer:
The number of the branch that is situated at the downstream boundary of the considered branch. It is emphasized, that the schematization of the problem has to be so, that no branch has a downstream branch with a lower number than it has itself. This is due to flow of calculations of the program ODIRMO. The only exception is the branch at the downstream boundary which gets the downstream branch number 0.

-- The program will now ask for the length of the next riverbranch until all river branches are asked.

-- The program calculates the number of the upstream branch(es) of each branch

Question:
ARE YOU SURE? (Y/N)
Answer:
If you think that you have made a mistake in the input so far, press 'N' and the program will ask the questions so far again else press 'Y' and the program will go to the next subroutine, in which the width of all branches is asked.
1.3 Width

In this routine the program asks for the width of each river branch. It starts with the following menu:

The width of the mainstream in branch 1 is:
1. Constant
2. Variable per grid point

You will notice that C(onstant) and V(ariable) are highlighted on the screen, indicating the keys you can press to choose an option. Pressing C means that the program will ask the width of the mainstream of this branch only once, while pressing V means that the computer will ask the width at each point separately. After you have chosen your option, the next menu appears:

The width of the floodplain in branch 1 is:
0. Constant = 0 (no floodplains)
1. Constant > 0
2. Variable per grid point

Pressing the highlighted key will activate the chosen option. Now a third menu appears on the screen:

The sediment-transporting width in branch 1 is:
0. Equal to the width of the mainstream
1. Constant (but not equal to the mainstream width
2. Variable per grid point

As it is not always the case that the sediment transporting width is equal to the width of the mainstream, it is possible to give this width an other value than the mainstream width. Pressing the highlighted key of the chosen option will activate this option. The screen clears and questions will appear to enter the width according to the options just given. The width must be given in meters. After entering the last requested width of branch 1 the question:

ARE YOU SURE?

appears on the screen again. Pressing N will cause the program to return to the menu about the width of the mainstream of branch 1. When you press Y the program continues with the same three menus for branch 2 and so on, until all branches have had their turn. Then the program will proceed with the next routine, in which the initial bedlevel is asked.
1.4 Bedlevel

In this routine the original bedlevel for each branch will be asked. It starts with a menu:

branch 1

Do you know the original bedlevel
1. at Every grid point
2. at One point only plus the slope?

Pressing E or Q activates the desired option for branch 1. If this branch has floodplains (you have already told the program in the routine ASKWIDTH if this branch has floodplains and it won’t ask you again). an other menu appears on the screen:

The level of the floodplains is
1. at a Constant level above the mainstream bedlevel
2. known at Every point
3. known at One point plus the slope

Press C, E or Q for the desired option.
If you asked for the Every point option the program will now ask successively the original bedlevel at every grid point.
If you asked for the One point option, the program will ask where in the branch you know the bedlevel (in meters from the upstream boundary of this branch), what the bedlevel is at that point (in meters) and the slope in this branch. The program calculates the bedlevel at every grid point immediately.
Now the question
ARE YOU SURE?
appears on the screen again (and it will return from time to time in the rest of the program but won’t be mentioned again, as the purpose is obvious and pressing Y or N has the same effect every time). Press Y to continue to the next branch or, if this was the last branch, to the next routine, in which the discharge will be asked. Press N if you made a mistake and the program will ask the original bedlevel in this branch again.

1.5 Discharge

In this routine the you will be requested to enter the discharge in each branch. The menu that appears depends on the fact whether the upstream boundary of this branch is situated an an upstream boundary of the considered problem, or at an intern boundary or a confluence. Upstream boundary menu:

branch 1
The discharge is:
1. known and Constant
2. known as Periodic function (e.g. yearly discharge curve)
3. known as an other Function of time

Pressing C causes the program to ask for the constant discharge ($m^3/s$). If you press P the program asks for the period in hours. It calculates how many timesteps there are in one period and asks for the discharge at every timestep during the first period. If you press F the program will simply ask for the discharge at every timestep.

Internal boundary or confluence menu:

The discharge is:
1. equal to the discharge of the Upstream branch(es)
2. equal to the discharge of the upstream branch(es) minus deltaQ (water Extraction)

If you press U then there is no further input needed, if you press E then the program asks for the water extraction ($m^3/s$).

The program now continues with the next branch or with the next routine, in which the upstream boundary for the sediment transport is asked.

1.6 Sediment Transport

In this routine the program asks for the sediment input at the upstream boundary of each branch. Just as in the preceding routine a difference is made between a real upstream boundary and a internal boundary or a confluence. For a upstream boundary the menu is:

The upstream boundary for the sediment transport:

1. Sediment input Constant
2. Sediment input Function of time
3. Bedlevel constant
4. Transport calculated with Transportformula
5. Sediment input equal to sediment output at end of flume (special flume facility)

Choosing option 1 will cause the program to ask for the constant input, option 2 to ask for the input at every timestep. The input must be given in $m^3/s$. If you chose option 3, 4 or 5 no further input is needed; the program handles accordingly to the desired option. Option 3: during calculations the transport at the upstream boundary will be made equal to the transport at the first grid point in the branch, so that no aggradation or degradation at the upstream boundary can take place. Option 4: the transport at the upstream boundary will be calculated with the used transport...
formula. This seems a peculiar option, as this is in fact not a real boundary condition. This option is, however, very practical and is used very often. It causes no particular numerical difficulties, as the calculations of backwater curve and bedlevel are carried out separately. Option 5 is implemented as a special flume facility, where the sediment that leaves the flume at the end is immediately transported to the begin of the flume and there supplied.

For an internal boundary the menu is:

1. Sediment input equal to sediment output of Upstream branch
2. Constant sediment Extraction at this point
3. Sediment input equal to sediment output at end of flume

Option 1 and 3 need no further input, and option 2 will generate the question: Give the constant sediment extraction. Enter the answer in \( \text{m}^3/\text{s} \).

The confluence menu:

1. Sediment input equal to sediment output of Upstream branches.
2. Constant sediment Extraction at this point.

In case option 1 is chosen, the upstream boundary condition for the considered branch will be calculated as the sum of the both transports that 'leave' the upstream branches. In case of option 2, this sum is diminished by a constant extraction at the confluence.

After all this the conditions for the next branch are asked or the next routine is called.

1.7 Waterlevel

This routine takes care of the input of the necessary downstream conditions for the backwater curve calculation. Analogous to the options given for the upstream boundary conditions, a difference is made between real downstream condition and internal boundaries and confluences. In case of a real downstream boundary the following menu appears:

branch i

The downstream waterlevel is:

1. known and Constant
2. known via General Q-h curve
3. known via \( h = h_{\text{Const}} + h_{\text{Fact}} \times Q^{h_{\text{Exp}}} \)

Entering C means you will be prompted to enter that constant waterlevel. After option 2 the program asks for how many values of the discharge you want to enter the waterlevel. After this you are alternately prompted for a Q and the h belonging to it. During calculations with ODIRMO the downstream boundary condition will be linear interpolated for
the present Q. Option 3 means the waterlevel is a fixed function of Q. You will be prompted for hConst, hFact and hExp.

In case of an internal boundary or a confluence downstream the appearing menu is:

branch i

The downstream waterlevel is:
1. Equal to the upstream waterlevel of the downstream branch
2. known via General Q-h curve
3. known via \( h = h\text{Const} + h\text{Fact} \times Q \times h\text{Exp} \)

Options 2 and 3 are similar to the option for a real downstream boundary and option E will produce no more questions.

1.8 Roughness

This routine asks how the friction term in the backwater equation should be calculated. For each branch the following menu appears on the screen:

branch i

The friction term in the backwater calculation should be calculated with:

a. constant Chezy value and the waterdepth
b. variable Chezy value per grid point and the waterdepth
c. constant Chezy value and \( R = a.w / (2.a + w) \)
d. variable Chezy value and \( R = a.w / (2.a + w) \)
e. constant Kn value and the waterdepth
f. variable Kn value per grid point and the waterdepth
g. constant Kn value and \( R = a.w / (2.a + w) \)
h. variable Kn value and \( R = a.w / (2.a + w) \)

j. roughness predictor 1 (see documentation)
i. roughness predictor 2 (see documentation)
j option not compatible with floodplains!

Options a, c, e and g will cause the program to prompt for the constant Chezy or Nikuradse roughness. In case floodplains are present, that value will be asked for the floodplain too. In case of option b, d, f and h, the user is requested to enter the roughness at each grid point in that branch. It will be clear that with the present options it is not possible to choose different options for the mainstream and the floodplain. This possibility is not difficult to add to the program. Choosing option i or j refers the user to the
Option i gives the Van Rijn Roughness predictor (1982) in combination with the water depth. Option j gives the Van Rijn roughness predictor combined with calculation of the hydraulic radius according to Einstein (1948). The use of the Einstein hydraulic radius calculation indicates that a wall roughness should be given. This wall roughness is asked for and stored in the array where usually the roughness of the floodplain is stored. Also an initial guess of the Chezy value is asked.

1.9 Grain size

Input of diameter of sediment particles. Subsequently d50 and d90 are asked. d50 is the diameter that is exceeded by 50% of the particles and d90 is the diameter that is exceeded by 10% of the particles. These grain sizes are used in the van Rijn roughness predictor and in some of the transport formulae. The Ackers and White uses yet another grain size, d35. In this program the array of d50 is used to store the grain size for Ackers and White! The following menu appears:

branch i

The d50 grain size is
1. Constant
2. Variable per grid point

Press C for option 1 and V for option 2. If option 1 is chosen the constant diameter will be asked, otherwise the diameter at each grid point will be asked. Next the menu for d90 appears:

branch i

The d90 grain size is
1. Constant
2. Variable per grid point

The same options as for d50.

1.10 Transport Formula
In this routine the program asks for each branch which transport formula should be used. Menu:

branch i

The used transport formula is:
1. Power law: \( s = mtr \cdot u^{ntr} \)
2. Ackers & White formula
3. Meyer-Peter & Mueller formula
4. Engelund & Hansen formula

For the power law \( mtr \) and \( ntr \) are asked. For the Ackers and White formula only an amplification factor will be asked, stored in \( mtr \). The transport will be calculated with: \( s = s(A&W) \cdot mtr \).

For the Meyer-Peter and Mueller formula the two constants in this formula will be asked. The usual values are 13.3 for the first and 0.047 for the second. The formula of Engelund and Hansen requires no further input, but, as with the Ackers and White formula, an amplification factor may be given. See chapter 4.3 for a discussion of the used transport formulae.

1.11 Source

In this routine a external sediment source can be given. Such a source means an extra term in the equation for the bedlevel. This term can for example be caused by sediment which is drawn into the stream from the riverbenches by ship movement or geological rise or fall of the riverbed. The source should be given in m/s. A menu appears:

branch i

This branch has
1. No external sediment source
2. A Constant sediment source

Press N for option 1 and the sediment source will automatically be set to zero. Press C for option 2. The source is asked.

1.12 WhatNext

This is a very short routine, which gives information about the just created datafile. It then asks:

press any key to return to main menu

The main menu will surely reappear after pressing a key. After this, it will be up to the user what to do next. He can choose to start calculations with the created datafile. This is very easy. ODIRMO will load into memory and execution
starts. An important facility can be, that the data can be transferred to the IBM mainframe of the University. This can be done by means of an interface program. In chapter 2 of this Part is described how to perform a transmission of data with the IRMA terminal emulator on the IBM personal computer.
2.2 Change the program

Changes in the programs and their subroutines can only be made in the source files. In the following part a short description is given how to make those changes. For a full description of the possibilities of TURBO-Pascal the user is referred to the TURBO-Pascal manual by Borland.

Insert source files disk in the diskdrive of the computer (should be a computer running under MS-DOS or PC-DOS). Switch the computer on. After a short while the computer asks the current date and time. After this the MS-DOS prompt appears. Now type TURBO and press <RETURN>. The TURBO-Pascal compiler is now loaded into memory. The compiler asks 'include error messages?'. Type Y, and the error message list is added to the compiler. Now the TURBO-menu appears on the screen. If you want to edit a file press E. The computer prompts: 'Work filename :'; enter the name of the file you want to edit. For example, if you want to edit the main program part of ODIRMO type : ODIRMO and press <RETURN>. There is no need to type the extension .PAS of the file, as this is the default extension assumed by this compiler. The first page of the file appears on the screen now. In the usual way this file can now be edited with TURBO's full screen editor. This means that you can freely move the cursor over the screen, scroll forward and backward with the cursor control keys and type the desired changes on the desired places. When the changes are made, it is time to save the edited version of the program. Leave the editor mode by pressing subsequently <CONTROL>K and <CONTROL>D. Now press the space-bar and the TURBO menu reappears on the screen. Press S to save the work file. It may be important to know that the old version of the edited program is not lost, but is renamed and save with the extension .BAK (e.g. ODIRMO.BAK). After any changes are made, these changes have to be copied to the files on the runtime disk (code files). If changes are made in any of the ODIRMO subroutines, a new file ODIRMO.CHN has to be created and if the changes are made in any of the CREADATA subroutines a new CREADATA.CHN file.

We still are in the TURBO main menu. Press O. Another menu
appears with compiler options. Press H. The chain file (.CHN) option is now activated. Press Q to leave the option menu and return to main menu. Press M. The program asks for the main file name. Type ODIRMO (or CREADATA) and press <RETURN>. Now press C and the compilation of the ODIRMO program with all the included files with subroutines will start. The created file ODIRMO.CHN is stored on the disk. After compilation is finished quit TURBO by pressing Q. The MS-DOS prompt reappears. Insert the disk with code files in drive B and type :copy odirmo.chn b:. The file will be copied to the runtime disk.

The files on the code-disk all contain executable code. They can be activated automatically: when the computer is started up with the run time disk in drive A of the computer, the MENU.COM program will automatically be started. This is done by the special file AUTOEXEC.BAT on the disk. When the computer is already switched on, the menu can be activated by typing MENU and pressing <RETURN>.

I think that the information above should be sufficient to make minor changes in the programs. When problems occur the MS-DOS manual and the TURBO-Pascal manual can be consulted. It is also possible to consult the writer of ODIRMO, CREADATA and this report.

2.3 Transmit data to mainframe

It is possible to send a datafile created by CREADATA to another computer by means of some kind of interface. It is then possible to perform the calculations with ODIRMO on that other computer. The most relevant option in this case is to send the datafile to a mainframe computer to save calculation time. It can be expected, that the IBM microcomputer takes approximately 500 times as much time for the same calculations in comparison with a fast mainframe. The IBM micro can be emulated as a IBM terminal for the IBM mainframe. This is done with the package IRMA. Consult the IRMA user's guide in case the following explanation is not sufficient for you. IRMA consists of software on disk and some interface hardware in the computer. When the IRMA hardware is present and the disk is available the following steps should be taken to transfer a file from the micro to the mainframe. Insert IRMA disk in drive A and type E78 and press <RETURN>. The terminal emulator program is loaded into memory and executed. If the IRMA hardware is resident and the connection with the mainframe is present a terminal screen lay-out will appear on the micro. Now you can logon under TSO in the usual way. After this, press <CONTROL><HOME>. This command makes the terminal screen lay-out resident in the computer. This means that it is now possible to return to the micro stand-alone mode and return to the terminal mode by pressing both the <SHIFT> keys at the same time. Return to micro mode. Now type FT78T and press <RETURN>. This is a program that takes care of receiving and transmitting files.
When it is loaded it will subsequently ask some questions about the file to be send or received. Most questions are self evident and need no further explanation. The question: 'send/receive binary file' can be answered with N. When the question: 'host operands' is asked one can answer DATA NONUM. In case the program ODIRMOMF (mainframe version in VS-Pascal on the source files disk) is transmitted, the last question can be answered by only pressing <RETURN>. When both ODIRMOMF and the datafile are copied to your TSO dataset. The usual jobcontrol statements have to be supplied. This is not the place for a discussion of jobcontrol, but one remark is made: one must not forget to give jobcontrol to identify the input datafile with the datafile that is read from by ODIRMOMF. And the same remark about the output datafile. When the inputfile is named PROBLEM.DATA a jobcontrol with the form of:

```
//GO. INDATA DD DSN=****.PROBLEM.DATA, DISP=SHR
```

should be added. (***) stands for a datasetname like WWVMK:VE).

When the output datafile is named RESULT.DATA a jobcontrol with the form of:

```
//GO. OUTDATA DD VOL=SER=DISKn, UNIT=DISK, DISP=(OLD),
// SPACE=(TRK,(3,2)), DCB=(RECFM=FB, LRECL=80, BLKSIZE=4560),
// DSN=****.RESULT.DATA
```

should be added.

It is advisable to consult a specialist to help you when you want to perform calculations with ODIRMO on the mainframe, because the user friendliness of this possibility is not considered optimal. However, the reward is: much shorter calculation times.