CLOSIM v 1.0

A Simulation Program on the Closure of Tidal Basins

Mark N.K. Middag
August 1994

PART I

Description of Hydraulic Aspects

The program Closim v 1.0 has been written
in fulfillment of the requirements for a Master's Degree from the
University of Technology of Delft, The Netherlands
in cooperation with the Dutch Ministry of Public Works

Professor: Prof. Ir. K. d'Angremond

Supervisors:
Ing. G.A. Beaufort
Dr. Ir. J.A. Cser
Ir. F.C. van Roode
Ir. G.J. Schiereck

Preface

This is volume *I* of the program manual which describes the hydraulic aspects of the simulation program on the closure of tidal basins *CLOSIM*. The program was written in fulfilment of the requirement for a Master's degree in Hydraulic Engineering at the University of Technology in Delft.

For the description of the computer program source code, the reader is referred to volume *II* of this paper. Volume *III* contains a user's manual for *CLOSIM*.

Mark N.K. Middag,

August 1994.

Contents

Introduction

The program *CLOSIM* has been written to simulate the practice of the closure of tidal basins. It has not been the aim of the program to provide a new design application or to have any scientific value with respect to the results of the calculations. The simulations should rather give insight in the way closures take place, and in the problems that might be encountered when closing the connection between a tidal basin and the sea.

In chapter *1*, the user is introduced in the practice of the closure of tidal basins; the main phenomena of the subject are discussed. In the chapters *2* to *6*, the calculations which have been done for the simulation program have been explained in detail. It should be kept in mind that most of the assumptions and calculations are based on rough estimations and do not give more than just an impression of the parameters and variables they deal with.

The boundary conditions of the calculations and the way of schematizing them for the simulation program, have been considered in chapter *3*. A lot of the geotechnical information presented has not been included in the program yet. Chapter *3* also covers the calculations of levels, flows and current velocities, of which the analytical derivation is presented in *Appendix A*.

The calculation of the construction activities (decreasing the flow area) is dealt with in chapter *4*. The complex geometric calculations which translate a dumped volume of material into a dam advancement, have been described in *Appendix B*.

The theory which has been applied to determine the loss of dam material (increasing the flow area), is explained in chapter *5* and *Appendix C*.

Chapter *6* accounts for the scour at the end of the bottom protection, which can cause the bottom protection and eventually the structure itself to collapse.

In chapter *7* general recommendations have been given for extension of the calculation procedures in future versions of the program.

1. The Closure of Tidal Basins

1.1 Introduction

In this chapter, a brief introduction on the closure of tidal basins is given. The main aim of this introduction is to accompany the simulation program *CLOSIM* and provide sufficient information on the various aspects of closures which the program covers.

The information in this chapter is just a general outline of the closure practice. A more detailed description of the applied calculation techniques can be found in the following chapters, where the calculation loop of the program is described.

The simulation program which has been described in these volumes, focuses on the variety in technical phenomena with respect to the closure practice. Non-technical aspects of the closure of tidal basins should be considered as beyond the scope of the program, and therefore irrelevant to this description. As a *technical* description of the closure practice, this volume describes only those phenomena which are dealt with in the program. For a detailed technical overview the reader is referred to the existing, scarce literature with respect to the subject, which has been listed in *appendix F*.

1.2 Tidal Basins

A *tidal basin* can simply be defined as any basin in connection with the sea. As the tide can freely enter and leave the basin, a stable situation will exist in which the tidal variations, the basin area and the bottom material determine the flow area of the branch between basin and sea. It is possible that the estuary has a river inflow. A schematisation of such a tidal basin can be seen in figure *1.1*:

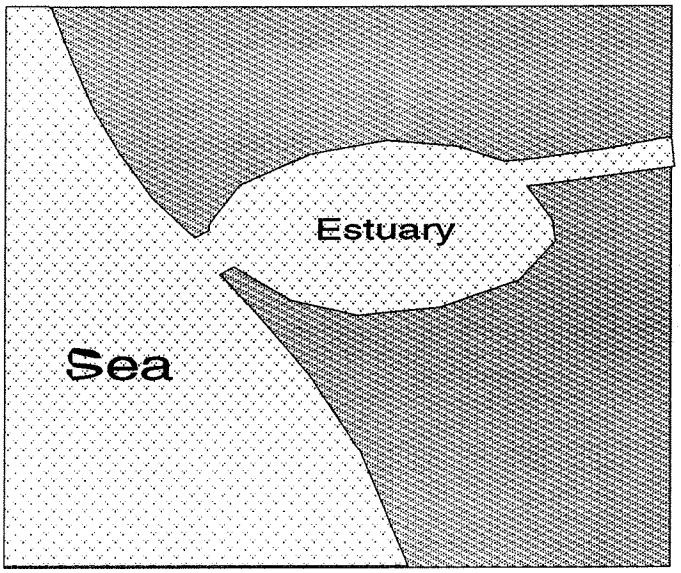


Figure 1.1: A simple tidal estuary

1.3 Closure Works

For various reasons a decision can be made to intervene in the process of inflowing and outflowing water, by closing the gap between sea and estuary: one could think of reasons such as the reclamation of land, the creation of a fresh water reservoir, the generation of tidal energy, and so on.

The closure of a tidal basin is technically difficult, and has numerous side effects which have to be kept in mind when deciding whether an estuary should be closed or not. Due to the multitude of aspects to be considered, the closure practice has had a long history full of struggle, through which experience has been gained by trial and error.

In reality, before a decision is made for tidal basin closure, a great number of considerations must be made with respect to the reason for closure, the location, and the impact which the closure will make on the environment and surrounding area. Various investigations and feasibility studies have to be made. As said, for the simulation program, these aspects have not been considered. The user of the program is not in any way confronted with these questions: his task can be described as purely constructional.

With respect to the structural part of the design, in practice a specific strategy is recognized. At first, in combination with the possible closure methods, the local hydraulic conditions are considered. The closure method that is finally chosen, also depends on the materials that will be used. The occurring velocities during construction lead to a force which these materials have to withstand.

1.4 Geometrical distinction of closure works

In the various methods for closing a tidal basin, two distinctions can be made:

* A distinction by *geometrical characteristics* of the closure;
* A distinction by *structural characteristics* of the closure.

In this and in the next chapter, both distinctions will be specified.

Based on *geometrical characteristics*, three types of closures can be considered:

* *Horizontal closures*, when the profile is restricted horizontally;
* *Vertical closures*, when the profile is restricted vertically;
* *Combined closures*, when a vertically restricted closure is finished horizontally.

1.4.1 Horizontal closures

When a gap is closed horizontally, the closure material is dumped from the channel banks. While this is done over the full height of the dam, the dam head moves progressively forward and is used as a base for further construction activities.

The dam head as well as the bed in front of the dam head, are seriously under threat from the currents. The construction should not be undermined; therefore precautions should be made. This can be done by constructing a bottom protection before the cross sectional area is decreased. Such a protection of the bed exists of a resistant layer, which can be spread out over several hundreds of meters of the initial channel profile.

The horizontal closure method can be applied quite easily from the banks of the gap. This clarifies why this method has been used so often. In figure *1.2*, a schematized view on the horizontal closure method is given:

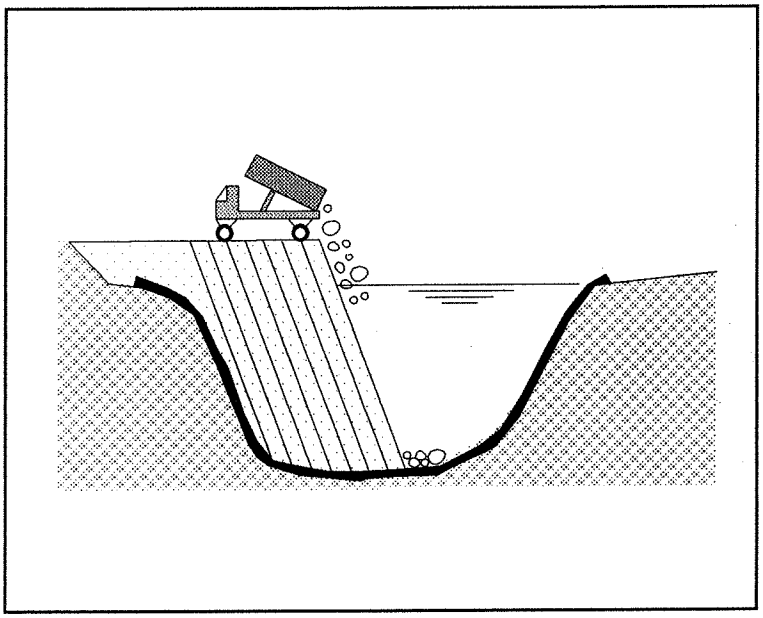


Figure 1.2: The horizontal closure method

1.4.2 Vertical closures

The *vertical closure method* distinguishes itself in dumping layers of material onto the bottom of the gap. Gradually, the gap area then decreases in vertical direction. As the construction advances, the water flow velocities increase above the dam crest, until the flow over the weir becomes critical; from then on, the flow velocities will only decrease.

In comparison with the horizontal closure method, the current attack on the construction and bottom protection is less severe; but the area undergoing current attack is much larger.

For the execution of vertical closures there are various possibilities. Temporal bridges or cable ways can be used, from which the material is dropped; dumping can also be done from dump vessels. These vessels though can only be applied as long as the water above the dam crest is deep enough.

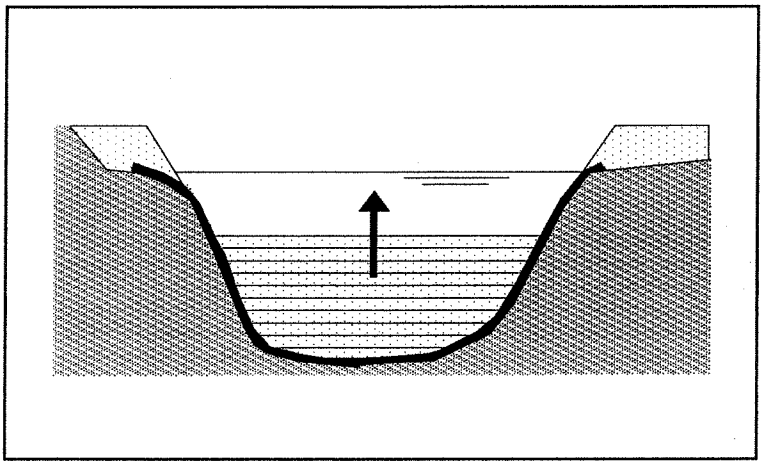


Figure 1.3: The vertical closure method

1.4.3 Combined closures

The third geometrical possibility exists in a *combined closure*. First a sill is realised over the whole width of the gap that is to be closed; thereafter, the construction is finished by dumping material from the banks of the gaps.

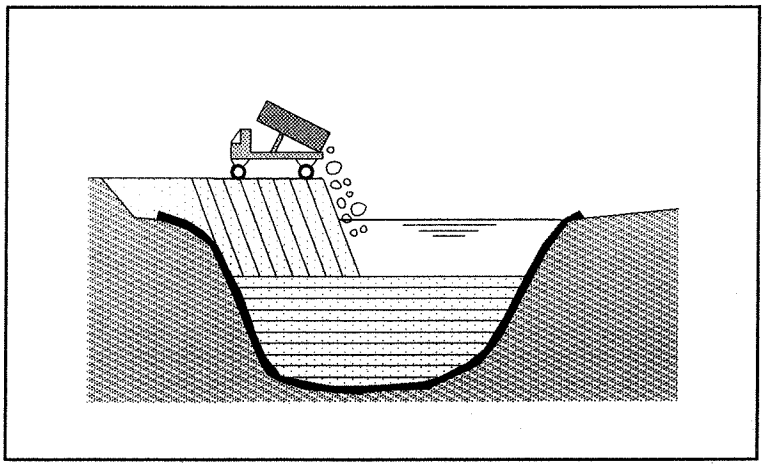


Figure 1.4: A combined closure

After a sill has been constructed, the gap could also be closed with large elements, such as caissons. In the actual version of the simulation program, this possibility has not been considered as of yet. In figure *1.5* the caisson closure type has been illustrated:

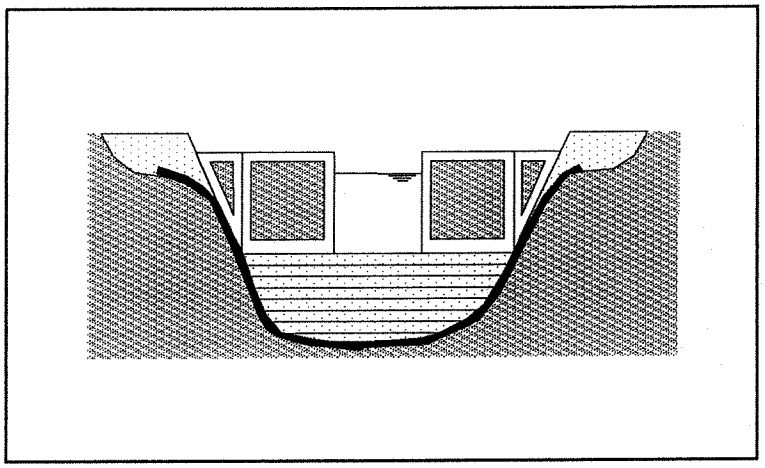


Figure 1.5: The caisson closure method

1.5 *Structural distinction of closure works*

Based on structural and material characteristics of the closing practice, three types of closures can be discerned:

* *A sand closure:* when this method is chosen, only sand is used to close the gap. It will be clear that a closure with sand only will not be possible in every circumstance. Both the hydraulic conditions (currents and waves) as well as the availability of sand are aspects to be taken into account. In the Dutch situation, especially the availability of sand is a positive aspect when compared to the use of quarry stone, which needs to be imported.

  The sand closure usually occurs horizontally, in which the sand is brought into suspension and discharged into the gap from the banks. In other occasions the sand is dumped vertically from hopper dredgers or split barges, after which the dam is completed horizontally.

* *Gradual closure with larger material:* in this case quarry stone, concrete blocks or gabions are used to close the gap. This construction method has been applied most frequently. This closure type can be applied either horizontally or vertically. Often, the larger material is not available, or transporting it to the closure site would cause this option to be too uneconomic. In this case, concrete blocks could be made.

* *Closure with caissons:* If a gap is closed applying caissons, first a sill is constructed vertically to provide a foundation for the closure elements. After that, the elements are brought into the gap and placed on the flattened bottom. Special elements can be used such as open caissons; when these are placed in the closure gap, they can be closed suddenly. Such a sudden closure could be interesting because that way gaps with high occurring velocities could be closed.

In the following chapter, the sand closure and the gradual closure with larger elements have been considered in detail.

1.5.1 Sand Closures

1.5.1.1 Method of Closure

A *sand closure* is executed by the application of great volumes of sand, which are supplied either by a hopper dredger or split barge from which the sand is dumped, or by a pipeline which takes a water-sediment suspension to the head of the dam.

A general characteristic of sand closures is the shifting and loss of the construction material. A sand closure is based on the requirement that the production of sand is larger than the loss. Also when the current velocities in the closure gap are not very high, sand losses occur. Losses can be distinguished in *net loss* and *gross loss* [Huis in 't Veld, 1987]. Gross loss is sand that is moved to another place than where it was dumped, but still fits within the future dam profile; net loss is material which is taken beyond the profile of the final dam. It can be seen, that after construction the latter material has been of no use, and that net loss should be avoided as much as possible. In figure *1.6* the different types of losses have been illustrated:

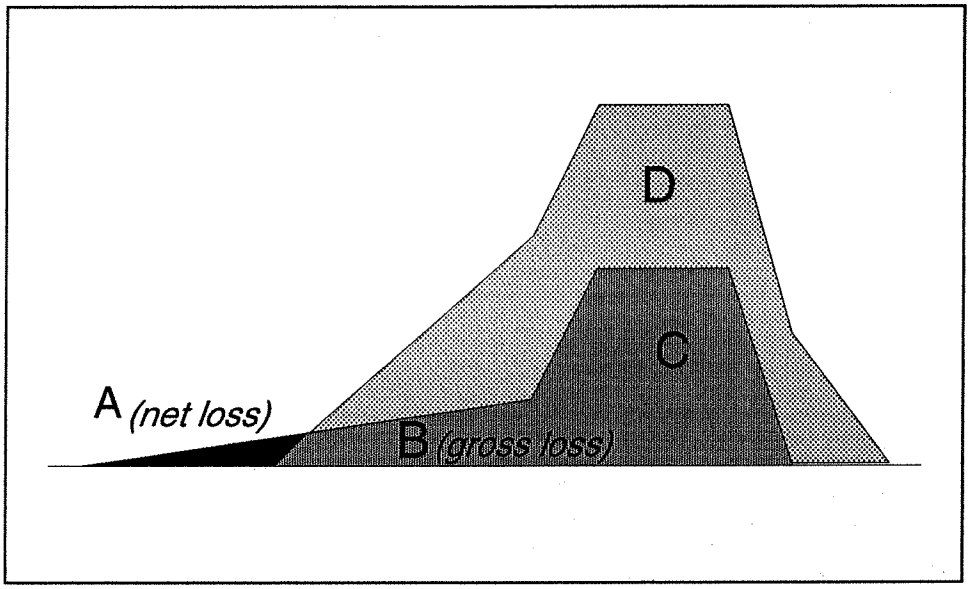


Figure 1.6: A sand dam cross section

In this figure, *D* indicates the dam volume that will finally be constructed. The smaller volume *C* indicates the stage, in which the gap has been closed already. Hence, *B* has been "lost" during the construction of *C*, but will be useful because it reduces the required volume for the final volume *D (gross loss)*; volume *A* though, has been lost during the construction of *C* and will not be useful *(net loss)*.

In the simulation program, both volumes *B* and *A* will be considered as loss. It is as if the task for the user of the program is to construct only *C*. The losses *B* are especially important because they reduce the sand production for the closure and hence the advancement of the closure.

In studying whether a sand closure will be possible, attention has to be paid to the maximum losses which will occur per tidal period. When the greatest possible production (also per tidal period) is higher than this maximum loss, the dam closure by sand will be possible. The production always needs to be higher, because of the insecurities in the values of both calculated losses and predicted productions.

As shown in figure *1.7*, the losses as a function of the gap area can be described by a curve with one extreme [Konter et al, 1992]. In most cases, these extreme losses will take place when the gap area is about 0 to 30 % of the initial gap area.

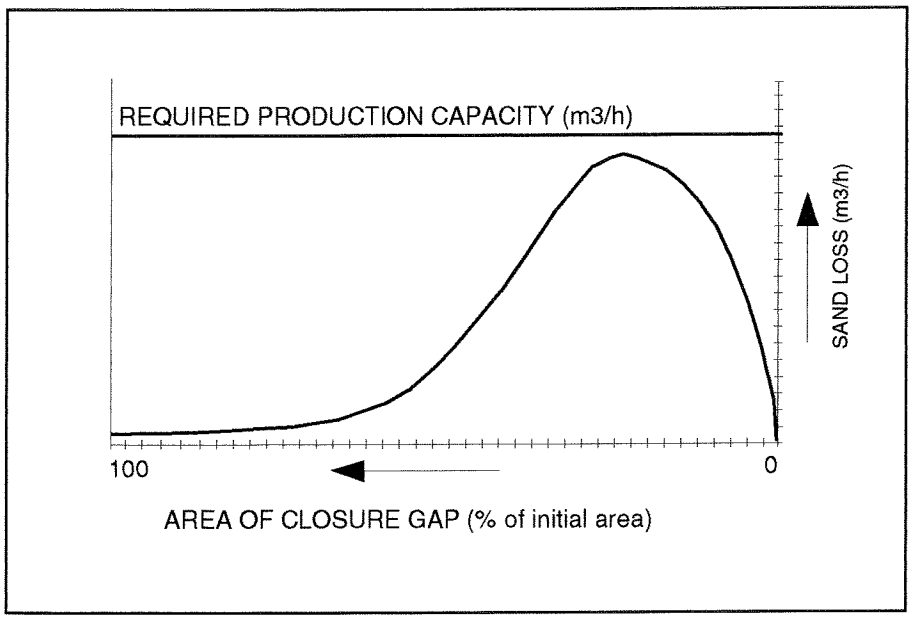


Figure 1.7: Sand losses during closure

It is interesting to see, that the maximum sand loss does not occur when the gap is nearly closed. Then, the current velocities are high, but the width at which erosion takes place is so small that the total sand loss is small too.

In general it can be concluded from experience, that sand closures are possible when the maximum current velocity does not exceed 2.0 to 2.5 m/s [Konter et al, 1992].

1.5.1.2 Cross sectional Design of a Sand Dam

To choose the right design and construction method of a sand dam can be very important for the dam cross section and the losses that will occur. The most chosen construction method for sand dams is the horizontal one in which pipelines discharge sand in suspension from the banks into the gap.

Crest width

In case the dam construction is horizontal, the crest width depends on the amount of pipelines that will provide the filling material. In practice, it is favourable to restrict the crest width, because it reduces the dam volume and both gross and net loss. A wider crest has the advantage that it leaves more available working space.

Equipment

In the actual version of the program no attention has been paid to equipment - pumps, pipe lines which have to be lengthened, etcetera -, but closing activities are simply entered as a production (in *volume per time*). Future versions of the program could be extended with respect to this, and thus make the program to be more realistic.

Particle sizes

Especially in the case of a sand dam, the dam slopes depend a lot on the size of the particles. Hence, when smaller material sizes are used, there will be more losses and at the same time the total dam volume (and thus the required amount of material for the closure) will be larger as a result of the gentle slopes.

1.5.2 Gradual closures with larger elements

1.5.2.1 Method of Closure

When a gradual closure is made with quarry stones or concrete elements, the construction can be either vertical or horizontal. In case of a horizontal construction the dam heads are exposed to the strongest current attack. The upstream side of the dam head has to withstand the majority of the load. If the closure is vertical, the hydraulic load is less severe, but occurs over the whole width of the gap.

In both construction methods, the velocities that will occur during closure are different. As has been said before, when the closure is vertical, at a certain moment the velocity will not increase anymore because the flow on the weir has become critical. At that time, the velocities do not depend on the level difference, but only on the energy height of the upstream water with respect to the sill. Hence, when the closure is continued, velocities will be lower. For horizontal closure, the velocities increase until the whole gap is closed. The velocities for horizontal and vertical closure have been shown in figure *1.8* on the next page. Also the possibility for a sudden closure as mentioned in section *1.5* is indicated.

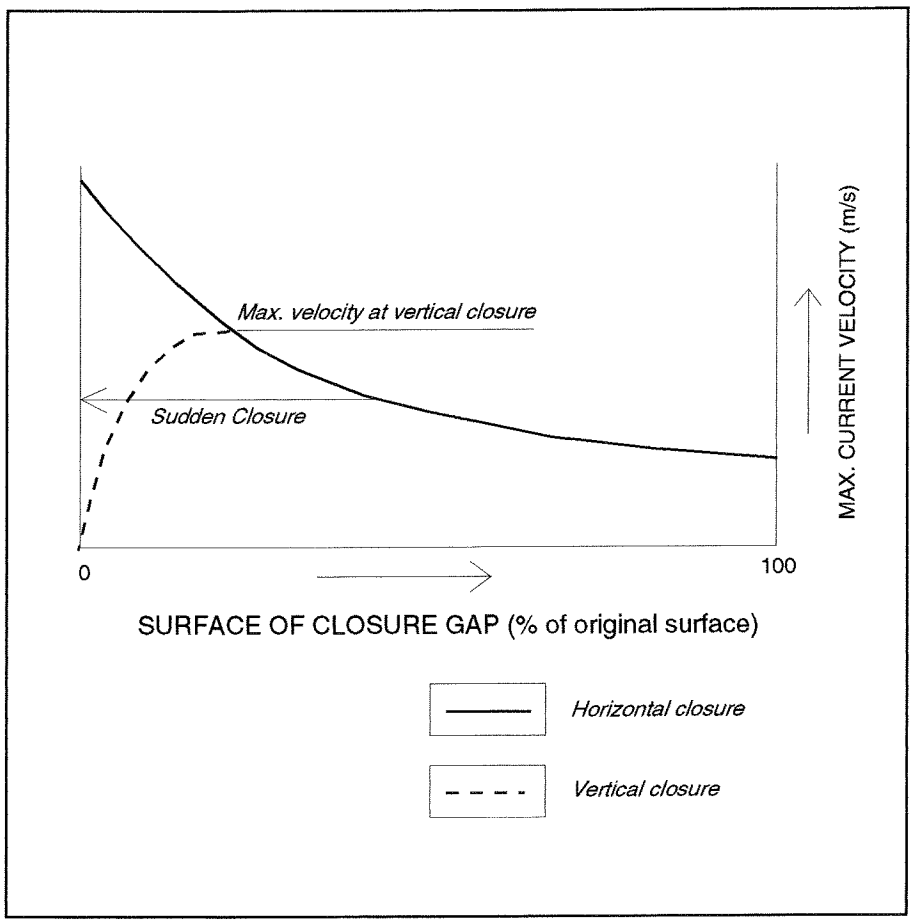


Figure 1.8: Current velocities during closure

As a result of this difference, a vertical closure can be executed with smaller stone sizes than a horizontal closure.

1.5.2.2 The gradual horizontal closure

If a closure gap is constricted horizontally, the current velocities increase in proportion to the gradual decrease of the cross section area. Therefore, for the last gap to be closed, heavy material is needed since the current attacks are severe then. In areas with a large tidal range, the horizontal closure is not favourable because of these high velocities. If it is still decided to close horizontally, the last gap could be closed by a larger element (*sudden closure*).

Another solution would be, to build a sill first, maybe even so high that critical flow is achieved, and then continue the closure horizontally. This is called a *combined closure.*

1.5.2.3 The gradual vertical closure

Characteristic of a gradual vertical closure is the raising of the sill over the whole width of the gap. It is possible to construct in horizontal layers, but also to follow the shape of the cross section of the channel to be closed. Quarry stone, clay- or sandbags are used for a vertical closure. The dumping of these materials can be done in various manners, as:

* By *manual labour* or by *trucks* over a (temporal) bridge;
* By *floating equipment,* such as stone dump vessels or floating cranes;
* By *cableway;*
* By *helicopters.*

As has been said, in the simulation program no attention has been paid to the various equipments that can be used. Before a vertical closure is initiated, the user has to decide whether the closure will be done from *above* the water line (e.g. by cable way or helicopters) or from *below* the water line (e.g. by dump barges). This is, as will be explained later, important for the calculation of the dam advancement.

1.5.2.5 Bottom Protection

When a dam is built to close a tidal basin, the inflow and outflow through the narrowed gap will be reduced, causing a decrease in the tidal range in the basin. But at the same time, the current velocities through the remaining opening will be higher. As a consequence, the scouring effect on the bottom near the dam will be increased, which endangers the stability of the river bottom and thus the foundation of the structure to be built on this bottom. This implies that the bottom, when consisting of easily erodible sand, must be protected by current-resistant material. Naturally, these problems will not be encountered at places where the bottom material consists of rock.

When a bottom protection is laid, scour holes will develop at the end of this protection. These scour holes, which depend on the magnitudes of turbulence and velocity and the period of exposure, could threaten the construction for stability reasons. The optimum

length of such a protection should be considered. In the picture, an example of closing structure, bottom protection and scour hole has been given.

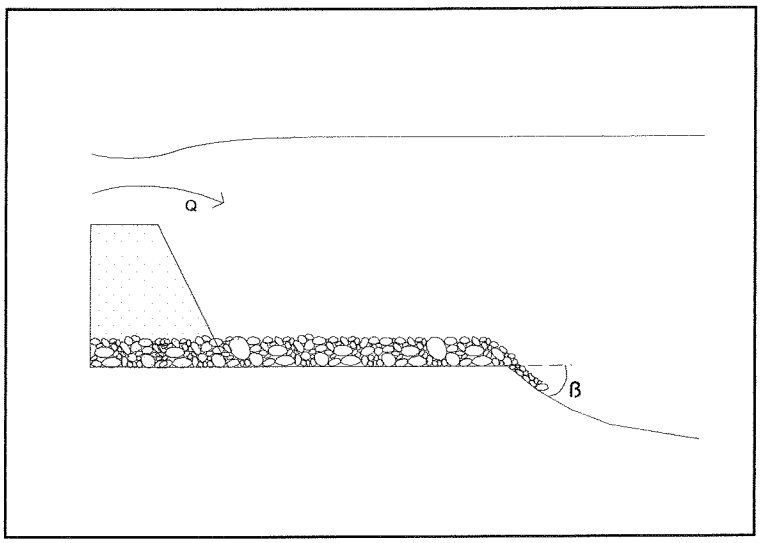


Figure 1.9: The bottom protection

The longer the protection, the more expensive; but a longer protection is effective in two ways: the scour holes that develop will be smaller, because the flows will be less turbulent at the end of a bottom protection, and at the same time the threat of the scour will be less, because it will occur further away from the structure.

For the protection of the bed, there are various options available to the designer. The protection can be prefabricated, or constructed in situ; it can be permeable or impermeable; many materials can be used. Each of these types of bottom protection will respond differently to the loads that will be present.

After a scour hole has been formed behind the protection, this hole could collapse when a certain depth or slope is reached. The behaviour of the bottom protection in case of such a collapse is an important point of attention. If the protection settles, but its function is not destroyed by this settling, the lowered bottom is still protected and the maximum scour still occurs at the same location as before the settlement; an example of such a bottom protection could be achieved when geotextiles are used. If settlement does damage the protection, the situation changes. Collapse of the scour holes will then lead to a shorter bottom protection: the location of the maximum scour approaches the dam construction and the total scour will increase, because closer to the dam the vortices are more intense. In this case, collapsing scour holes behind the bottom protection will soon threaten the construction itself.

In the actual version of the simulation program, no attention has been paid to the materials that can be used to protect the bottom. The quality aspect though, has been considered: the user of the program can decide how a bottom protection will respond to a collapse of the scour holes behind it, by marking the quality of the bottom protection to be made as being good or bad. A *good* bottom protection will keep its function when it has settled, while a *bad* protection, once distorted, will not protect anymore.

The quality of the bottom protection is important with respect to the scour depths (which depend on the length of the bottom protection), and to the place where this scour occurs (at the end of the protection). In figure *1.10*, this has been explained:

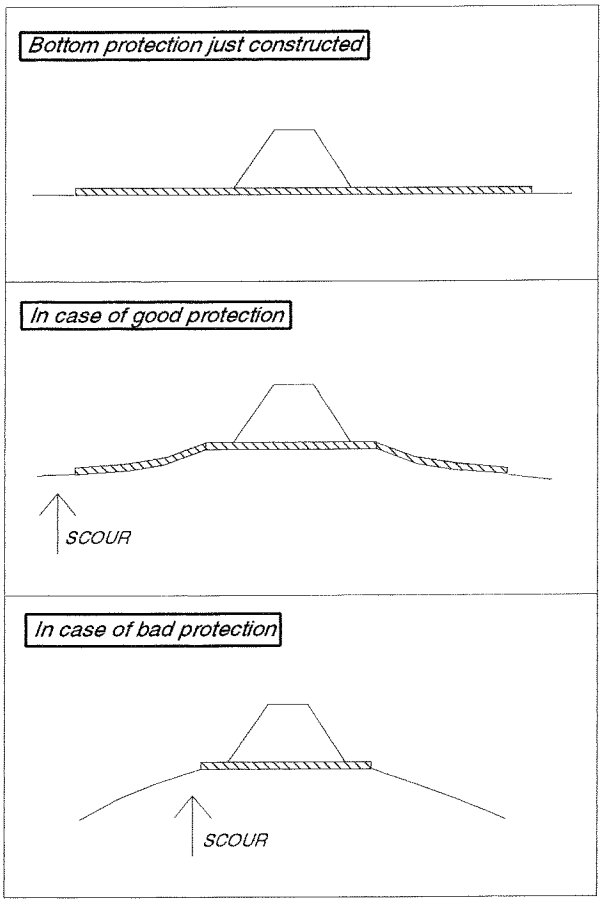


Figure 1.10: Protection and settlements

In case of a good protection, the scour will go on at the same location as before the collapse. In case of bad quality though, for this section the bottom protection length will be decreased. Scour will then occur right behind the new end of the bottom protection.

1.5.3 Sudden Closure

The sudden closure such as with caissons has not been considered in this version of the program. When the program is extended to cover this closure type, a series of extra calculations have to be included as well, because the design of a sudden closure has several other implications than a gradual closure.

2. Simulation of Closure Works

2.1 The Simulation Program

In the previous chapter, an introduction in the closure practice has been given. In order to be able to get acquainted with the main phenomena that occur during closure, the simulation program *CLOSIM v 1.0* was written. The actual version of the program contains a lot of assumptions and restrictions that should be worked out in future versions.

In the following chapters *3* to *6*, the calculations which have been done for the simulation program have been explained in detail. The user is reminded to the fact that most of the assumptions and calculations are based on rough estimations and do not give more than just an impression of the parameters and variables they deal with.

2.2 Scope of the Simulation Program

As said before, the phenomena that occur during closure are complex and therefore the scope of the simulations in *CLOSIM v 1.0* has been restricted. The program only deals with gradual closures; sudden closures with caissons could be added in future versions.

With respect to the *input* of the program, there are various limitations. The dimensions of basin, gap and structure are roughly schematised; the maximum number of closure gaps is three. The bottom material is assumed to be granular and non-cohesive; further details about the soil conditions are not included. Of the weather conditions, only storm is regarded; the direct impact of waves on the structure has not been taken into account.

The *calculations* apply existing designing relationships, which are extended where needed to be suitable for the simulation program. As will be explained in section 3.2.2, the flow calculations are executed with the *storage model*, which has several implications for the cases that can be considered. For the slopes that will occur when the material is dumped, rough estimations had to be done, because no better information was available. Also on the quantitative erosion of coarse dam material, no investigations have been done yet. In the program, existing relationships have been extrapolated, as will be explained in chapter *5*. For these erosion calculations, various flow regimes should be counted for. An important parameter for these calculations is the *porosity* of the structure, which has not been dealt with, but which should be included in future versions. Contraction effects of the currents that pass the dam head, have been ignored up till now.

Important is also the limited number of failure mechanisms that has been included. The most important of these mechanisms have been illustrated in figure *2.1*:

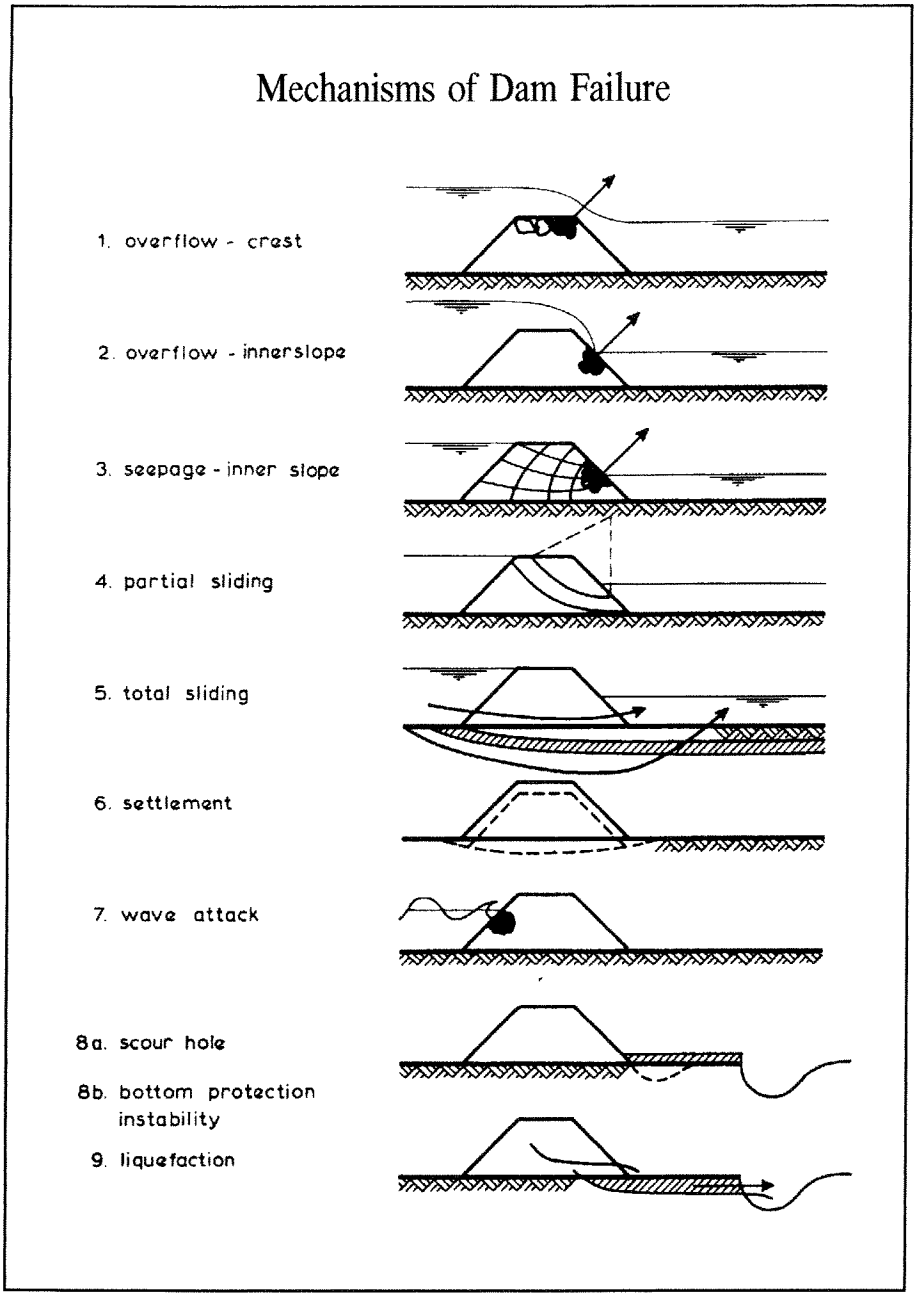


Figure 2.1: Failure mechanisms for closure dams

When these mechanisms are observed, it is clear that a lot of main failure mechanisms should be included in future versions of the program. In *CLOSIM v 1.0*, most of the attention has been paid to the mechanisms *1* and *8*.

The most important of the actual calculation procedures which the program contains are:

* *Determination of Conditions (Soil, Tide, Storm, River);*
* *Flow calculation with conditions and gap dimensions;*
* *Calculation of dam advancement (construction activities);*
* *Calculation of loss of dam material (erosion);*
* *Calculation of Scour behind Bottom Protection;*
* *Check of Dam Stability.*

In the following chapters each of these calculations will be specified.

3. Determination of Conditions

3.1 Boundary Conditions

In this chapter the conditions are discussed which are assumed to be left unchanged by the closure of the tidal basin. Attention will be paid to the geotechnical conditions, the tidal conditions, surges because of storm, and the inflow of rivers into the basin.

3.1.1 Geotechnical conditions

In the simulation program, the conditions of the subsoil have not been considered. In the program is assumed, that the bottom exists of granular material, of which the particle size can be entered. In future versions attention should be paid to this aspect, especially concerning the failure mechanisms which have to do with the subsoil (see *chapter 7* of this manual).

In practice, before a closure is started geotechnical data is obtained by sample borings and penetrometer tests [Huis in 't Veld et al, 1987], which are done on location of the dam to be built. Important parameters are determined, in order to obtain information about the foundation on which the dam will be constructed. In case of a soft soil, these parameters are cohesion and angle of internal friction; for granular materials, the grading curve, the permeability and the mass density are of highest importance. The packing is a special point of attention, especially when the content of fine particles is high (low permeability).

The strength of the soil foundation should be compared to the final load which will act on it, because of the construction weight. The phasing of the construction is also of importance for the stability of the subsoil, with respect to the induction of the loads. When the foundational strength is exceeded, in general there are three ways in which collapse might occur [Van Roode, 1994]:

* *Sliding.* In case a large, rather steep slope is created (either by construction activities, or by an erosion pit which occurs next to the construction), the dam profile might become instable and slide. Whether this occurs or not, depends much on the internal friction in the material, the height of the slope, the quality of the subsoil and the water levels. The slope safety factor against sliding away may be determined with stability calculations like *Bishop's method,* by which the driving force (the weight of the slope) is compared with the resisting force (the friction). In figure *1.11*, an example of sliding is given:

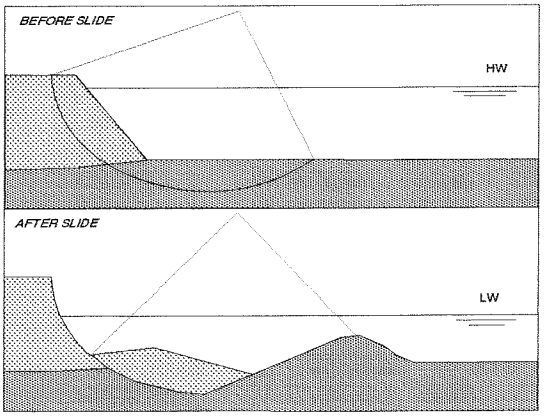


Figure 3.1: Sliding

* *Squeeze.* If the dam material is stable and the dam construction advances, a different kind of instability might occur. If the subsoil is weak and the construction imposes a great pressure, it will escape squeezing away from below the structure. This is indicated in figure *1.12*:

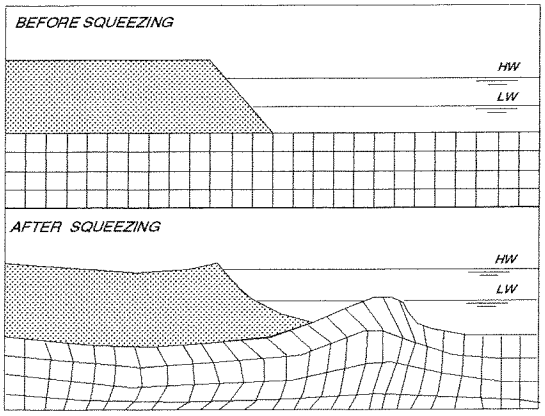


Figure 3.2: Squeezing

Although both mechanisms of failure are very unlike, during construction it might be difficult to identify which mechanism that has taken place [Van Roode, 1994].

* *Liquefaction*. Liquefaction occurs when a structure is built on or with loosely packed sands. If the voids between the grains are filled with water, any change of the packing will result in an overpressure condition, which makes the water withstand part of the load. The grains will then no longer make contact and the mix of sand and water will be *quicksand*, which behaves like a fluid:

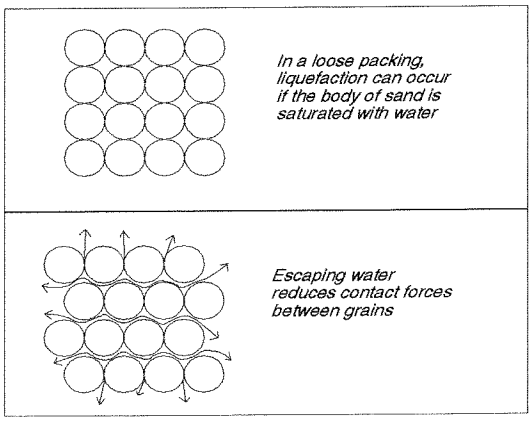


Figure 3.3: Liquefaction

Liquefaction especially occurs in loosely packed sand with a low permeability. Because a sand structure which is built under water always has a loose packing, the permeability (and thus the content of fines) will be important. If the sand is not very permeable, a small impact might already change it into quicksand.

In all cases of instability, restoration of the dam is needed. The instability has to be considered seriously; to ignore it and continue constructing could deteriorate the situation. Sometimes just waiting to let the water overpressure release, will help to improve the situation.

3.1.2 The Tidal Conditions

At the start of the closure, the level in the basin will very closely follow the tidal variations that occur outside of the basin. When the final stage of closure has been reached, the phase and amplitude of the level fluctuations in the basin will be totally different from the unchanged sea level fluctuations. This difference will be a reason for higher velocities in the gap. It is obvious that for the closure practice, a thorough knowledge of the hydraulic behaviour during the closure is required.

Continuous variation in sea water level is due to *tidal influences*, a phenomena caused by gravitational attraction of earth, sun and moon. The influence of the moon can be about two times greater than that of the sun. The oceanic tidal waves propagate over the continental shelves to coastal seas, where they cause variations in water levels.

The tide is thought to occur by two types of forces which exist between the celestial masses. The first are the *attracting forces*, which are described by Newton's law of gravitation. If the earth is considered as a mass entity, these attracting forces are balanced by the *centrifugal forces* due to the rotation of the bodies around their common centre of gravity. But because of the water in the oceans, the planet cannot simply be considered as a mass entity. The distribution of the forces over the earth's area differs.

The attraction forces will cause the water to flow into the direction of their resultant, making the layer of water around the earth take on the form of an ellipsoid rather than that of a sphere. Where the layer is thick, is spoken about *flood* (high water levels), where the layer is thinner, this is called *ebb* (lower water levels). As a result of the several rotations, the minima and maxima of the tidal wave propagate around the earth.

For the explanation of *semi-diurnal tides* (tidal waves with a twelve hour period) it should be realized that due to the earth's rotation two high tides and two low tides will occur during one day at a certain place on the area. For sun and moon these influences are called the *S2 tide* and *M2 tide* respectively.

In case earth, moon and sun are aligned, the effects of moon and sun are in phase and cause a so-called *spring tide*, which takes place at full moon and new moon. Between those spring tides, when the celestial bodies are not aligned, the opposite occurs, which is called *neap tide*. Neap tide takes place during the first and third quarters of the moon.

The motion of the moon is not in the plane of the earth's equator, but an angle exists between the moon's orbit and the earth's equator. This angle causes the tides to have a *daily inequity*, in which two successive high tides or low tides may differ during the same day. These *diurnal tides* may equal or even dominate the semi-diurnal tides. Some examples of diurnal tides are named *K1, O1* and *P1*.

In reality, more phenomena with respect to the tides occur. For example, in shallow seas and continental shelves the tidal waves are influenced by the bottom topography and coastline geometry. The tidal wave is distorted non-linearily, because its crest moves faster than its trough (causing an *M4* tide), and the bottom friction leads to an *M6* component. The main goal in considering the tides in the simulations is to be able to simulate the main characteristics of the tidal conditions, such as ebb, flood, spring tide and neap tide. For the consideration of these phenomena in the simulation program, four components of the tide have been regarded to be sufficient. The user of the program can therefore enter the amplitudes of the *M2, S2, K1* and *O1* tide. Another important input is the phase which these components have at $t = 0$ (start of the closure); if all phases would be zero, the closure would always start at spring tide.

When the amplitudes and phases at $t = 0$ are varied in the program, the various aspects of the tidal conditions can be simulated.

3.1.3 Basin Storage Area

For the flow calculations, the volume of water which enters and leaves the estuary has to be known. This volume is the product of the tidal range and the storage area of the basin. This storage area though is not a constant, but depends on the varying level in the basin. In figure 3.4 this has been illustrated:

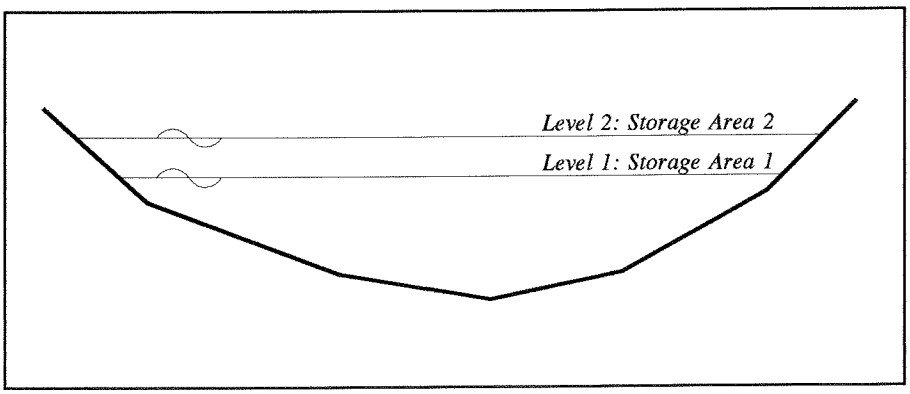


Figure 3.4: The basin storage area

The function for this storage area, together with the level in the basin, gives the actual storage area which can be used in the flow calculations. In the program five values of this function are entered, between which the storage area is determined by interpolation:

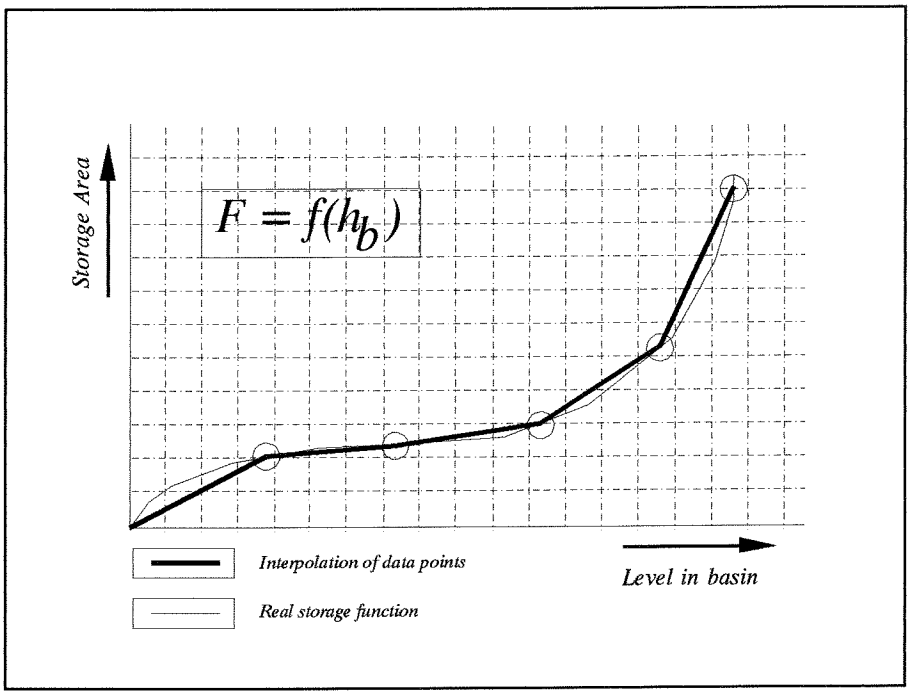


Figure 3.5: Schematisation of storage function

3.1.4 River Discharge

A river which flows into the basin, has been schematized as a constant discharge. When such a discharge is present, a spillway or sluice should be constructed, so after construction the water can escape. In the program such a spillway is simply entered as an extra flow area.

3.1.5 Storm

If a storm occurs, this has several consequences for the construction. In the simulation program, a storm effects the closure as a wave setup (a surge of the water level) and an increase of the velocities in the closure gap. The maximum wave height and setup, as well as the duration and probability of the storm, can be entered by the program user.

The raise of the water level due to storm setup has been schematized in the program as a cosine function, according to

$$\Delta H_{storm} = H_{\max} \frac{1-\cos^2(2\pi \frac{t}{\Delta t})}{2}$$

in which

ΔH_{storm}	the actual storm surge *(in m)*
H_{max}	the maximum storm surge *(in m)*
t	the moment in time that the storm is prevailing *(in s)*
Δt	total storm duration *(in s)*

This schematisation of the development of the setup during a storm, has been drawn in figure *3.6*:

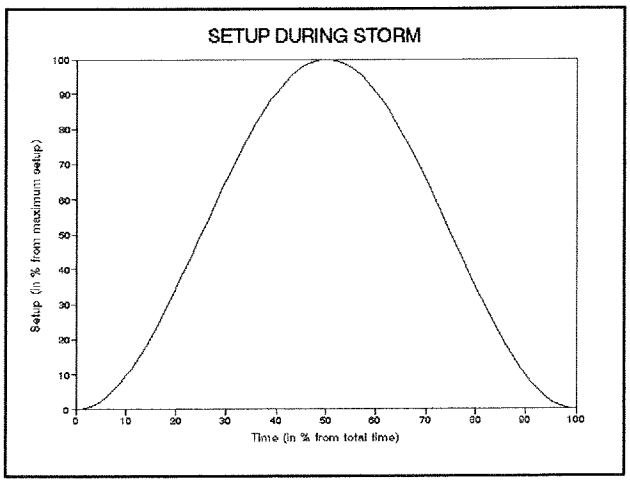


Figure 3.6: Setup during storm

The waves which enter the closure gap, affect the velocities that occur and are important for the material losses. The waves which are caused by the storm are taken into account in the calculation by adding one third of their height to the level difference for the *velocity* calculations. These momentaneous increases of the level above the crest, do not have any influence on the discharges through the gap.

The phenomena of wind waves and storm factors could be considered in greater detail, giving a more realistic touch to the simulation program. The wave module, which can be found in the calculation unit of the program, can be expanded upon easily in future versions of the program.

3.2 Boundary Conditions during Construction

The local hydraulic conditions at the closure location are of special importance in respect to the structural design of the closure. The information which is primarily required consists of the level differences, the flows and the current velocities.

3.2.1 Dimensions of Closure Gap

When in the simulation program the closure activities start, the tidal branch is assumed to be stable. The flow area, in which erosion and sedimentation take place caused by the tidal currents, will be more or less in equilibrium. When the construction activities take place, depth and width of the gap are changed.

For the flow calculations, the gap dimensions have been schematized as indicated in figure *3.7*. The calculations are done with the greatest depth and the average width of the gap:

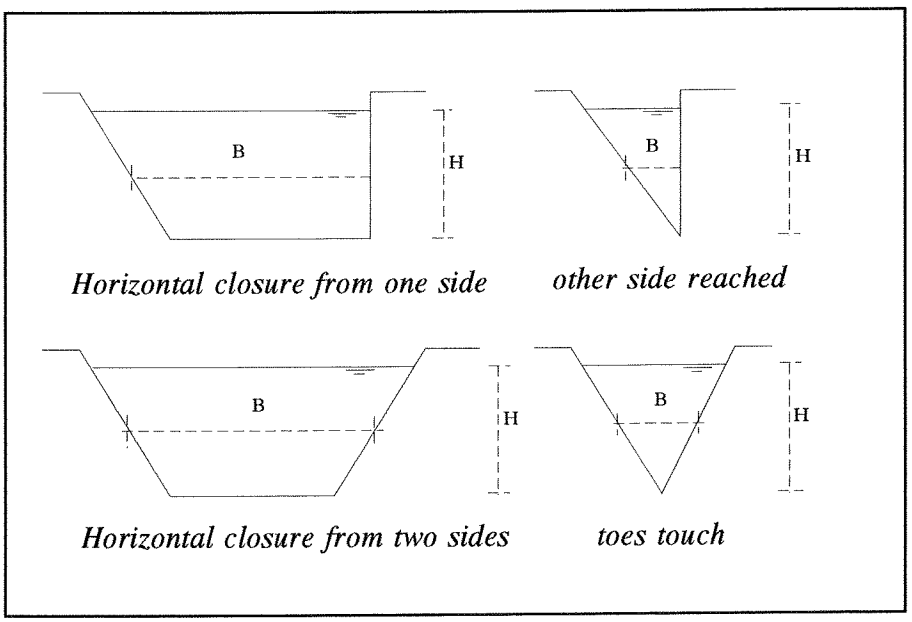


Figure 3.7: Dimensions of closure gap

3.2.2 The Equilibrium Flow Profile

The equilibrium profile, which is the profile which occurs when the tidal current is flowing through a branch during many years without any human action, is important for several reasons. As said, when the first calculations start, the user is referred to this equilibrium situation, and the program gives the possibility to adapt the gap dimensions.

In order to get an expression for the equilibrium flow area of a tidal inlet, investigations have been done [Steyn et al, 1991]. A linear relation was found [Niemeyer, 1990] between the tidal prism (the volume of water that flows in and out the basin during every tidal period) and this area. In figure *3.8*, this relation has been drawn:

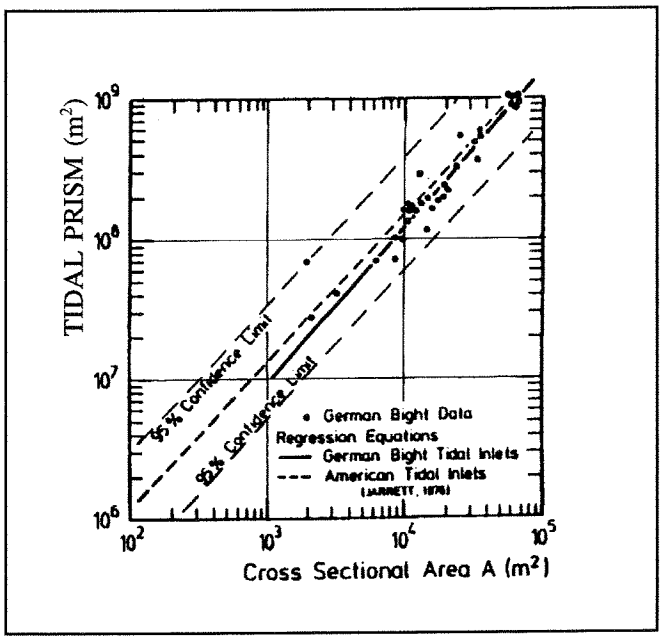


Figure 3.8: Tidal prism vs. cross sectional area

Each time the tidal range or the basin area function are entered or changed in the program, the tidal prism is determined and from this the equilibrium profile is calculated. For this calculation, a tidal amplitude should be given. This amplitude is time-dependent because of the various components of which the tide exists. To calculate the tidal prism, for this value 76 % of the sum of the partial amplitudes has been used. In *Appendix E* the theory behind this choice has been discussed.

3.2.3 Models for Schematisation of Flow Calculation

For the determination of the local hydraulic conditions, three calculation models are available [Klatter et al, 1990]:

* *The Storage Model:* In this schematisation an estuary is simply considered as a storage area. The mass inertia and frictional values of the water volume have been ignored; conditions to be entered in the calculation can be the sea level, the gap dimensions and a river flow. The closure gap is schematised as a weir. In figure *3.9*, an example of a storage model is given:

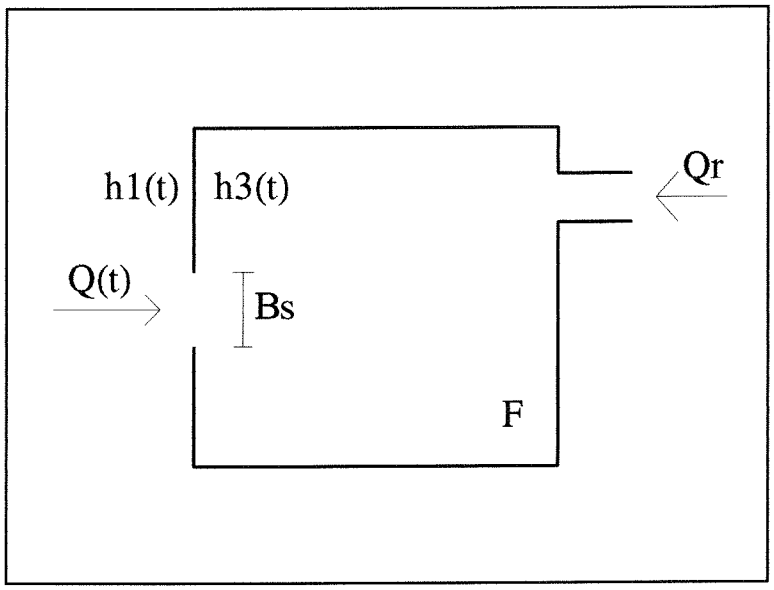


Figure 3.9: Schematisation of the estuary

In this figure, *Bs* indicates the gap width; *h1* and *h3* represent the inside and outside water levels respectively; *Q(t)* is the inflow (or, when negative, the outflow) of the estuary; *Qr* is the river flow. *F* is the storage area of the basin.

The more the flow area has been decreased, and how shorter the basin in comparison to the length of the tidal wave, the more accurate this schematisation can be. In case mass inertia and frictional values become more important (e.g. when the estuary is long and consists of various branches), especially in the beginning of the closure process the storage model calculations would produce irrealistic quantities. Then, the use of other schematisations may be considered.

* *The 1-dimensional model*: In this model the water movement is divided into branches, which are connected together in nodes. In both branches and nodes, a storage can be entered in the calculations, and also mass and friction can be taken into account.

* In more complex situations or when more detailed information is required, the conditions can be determined by a *2-dimensional calculation*, in which the water movement is described in two horizontal dimensions.

In this simulation the *storage model* has been applied. Application of the other models would clogg the program with a tedious slowness whereas the more precise information is not appropriate within the scope of the program. A result of this is, that the estuary to be closed is simply entered as a storage area. When in later updates of the program there would be a requirement to simulate the closure of longer basins, the 1-dimensional model could be considered, using more accelerated and advanced computers.

3.2.4 The Storage Model

The difference between the level at sea and the level in the basin, causes a flow in the closure gap. This flow causes the level inside of the basin to change, and the new basin level can be used in the next time step.

The *storage model* [Konter et al, 1992] offers a numerical solution to this relation between levels and flows. The model is based on a combination of the *weir formula* and the *continuity equation*. The weir formula represents the relationship between levels and velocities:

$$U_0 = \sqrt{2g\,(H_1 - h_2)}$$

while for a normal, non-critical flow:

$$h_2 = h_3 \qquad \text{for } h_3 > \frac{2}{3}H_1$$

and for a critical flow:

$$h_2 = \frac{2}{3}H_1 \qquad \text{for } h_3 < \frac{2}{3}H_1$$

in which	U_0	the specific velocity through the gap, given by U_{av}/μ	*(m/s)*
	g	the gravitational acceleration	*(m/s^2)*
	H_1	energy level	*(m)*
	h_2	water level in the gap with respect to the sill	*(m)*
	h_3	water level in the basin with respect to the sill	*(m)*

When with this velocity the discharge is calculated, this results in:

$$Q = \mu A \sqrt{2g\,(H_1 - h_2)}$$

in which μ is the contraction coefficient in the gap (a dimensionless factor).

The difference between the situations of *critical flow* and *non-critical flow* is shown in figure *3.10* on the next page:

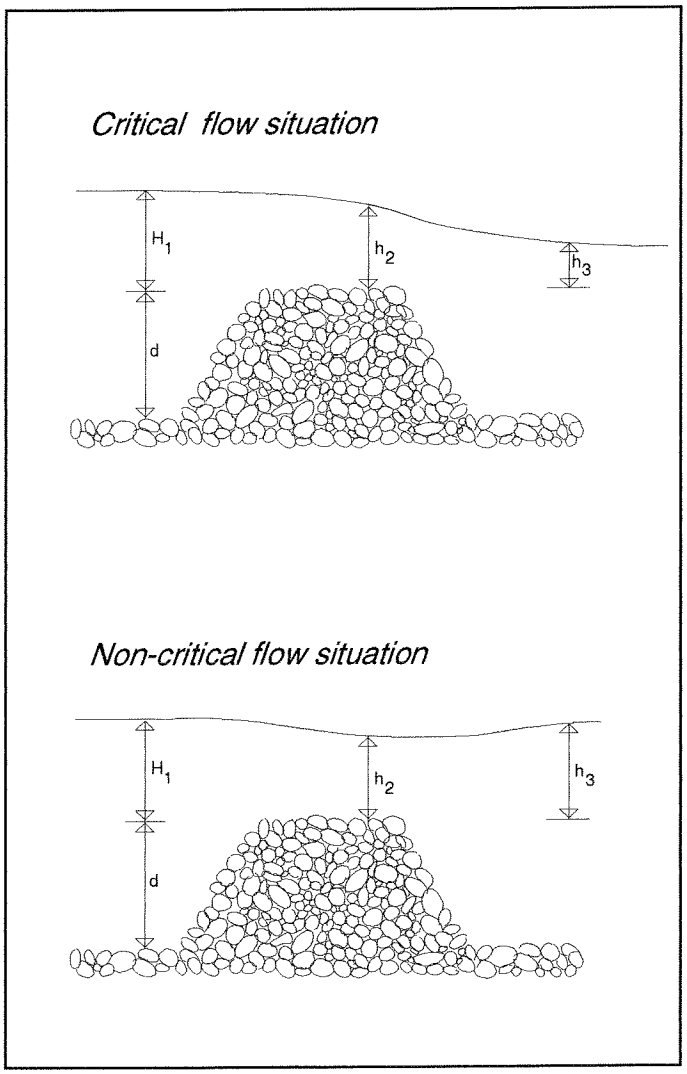


Figure 3.10: Flow regimes

4. Schematisation of Construction Activities

4.1 The Construction Advancement

As mentioned before, the closure activities can be realised horizontally and vertically. The horizontal closure can be accomplished from one or two sides, while it can be combined with a sill, constructed vertically in advance of the horizontal closure. In practice there are two ways to close the gap vertically: dumping the material at the bottom (directly at full width of the dam, for example by using vessels), or dumping from higher than the water level (e.g. by means of a cable way). The possibilities to be considered are shown in figure *4.1*, in which the vertical closure can either be done by vessels or by cable way:

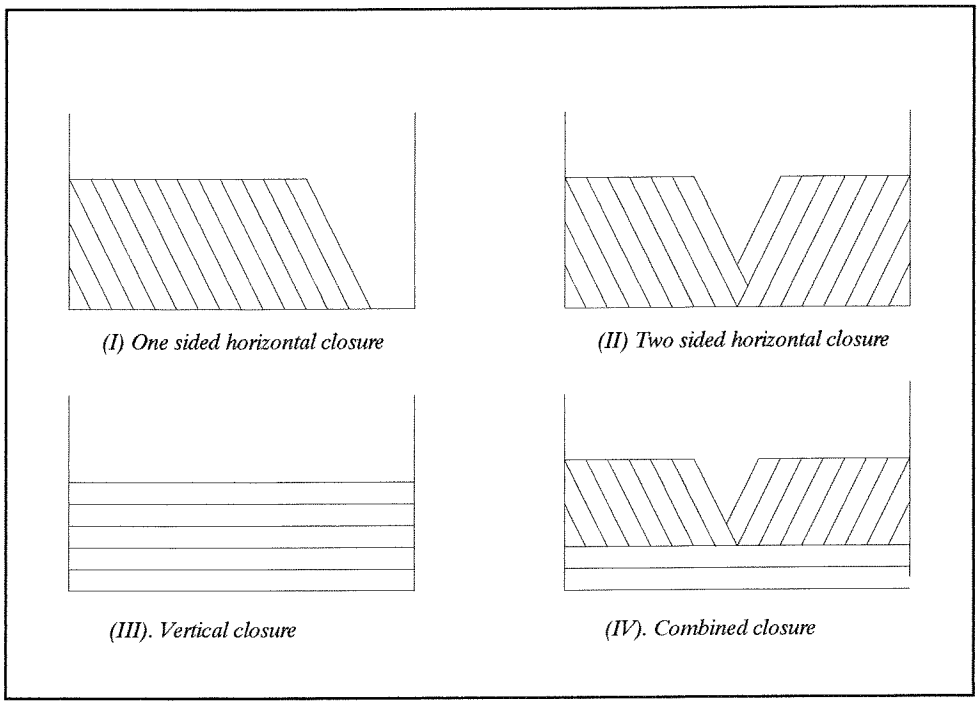


Figure 4.1: Closure schematisations

Distinction between these construction methods is important, especially for the hydraulic condition that will develop during construction.

To show and calculate the construction process, the dimensions of the dam to be built should be clearly defined. In the program, the dam head has been schematized. In this way, the calculation of the dam advancement can be two-dimensional. A summary of the variables by which the dam profile is defined, is given as:

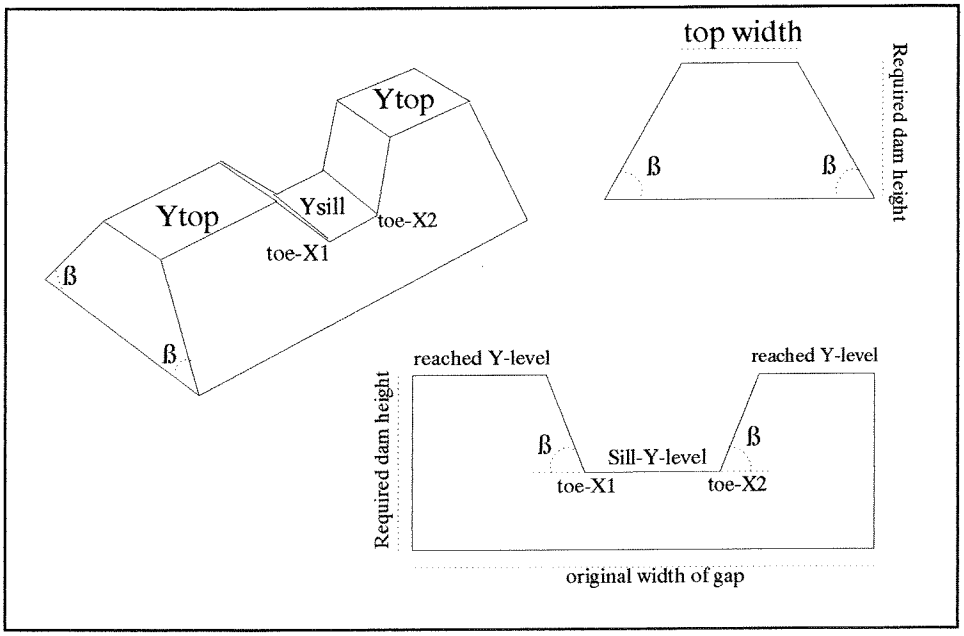


Figure 4.2: Schematisation parameters

In the case of horizontal construction, the advancement of the construction has been schematized by thin trapezoidal layers, although the real form of the added volume will be different. These layers will be added to the dam construction, which is modelled as shown in figure *4.3*. The model of the dam head for horizontal filling is as follows:

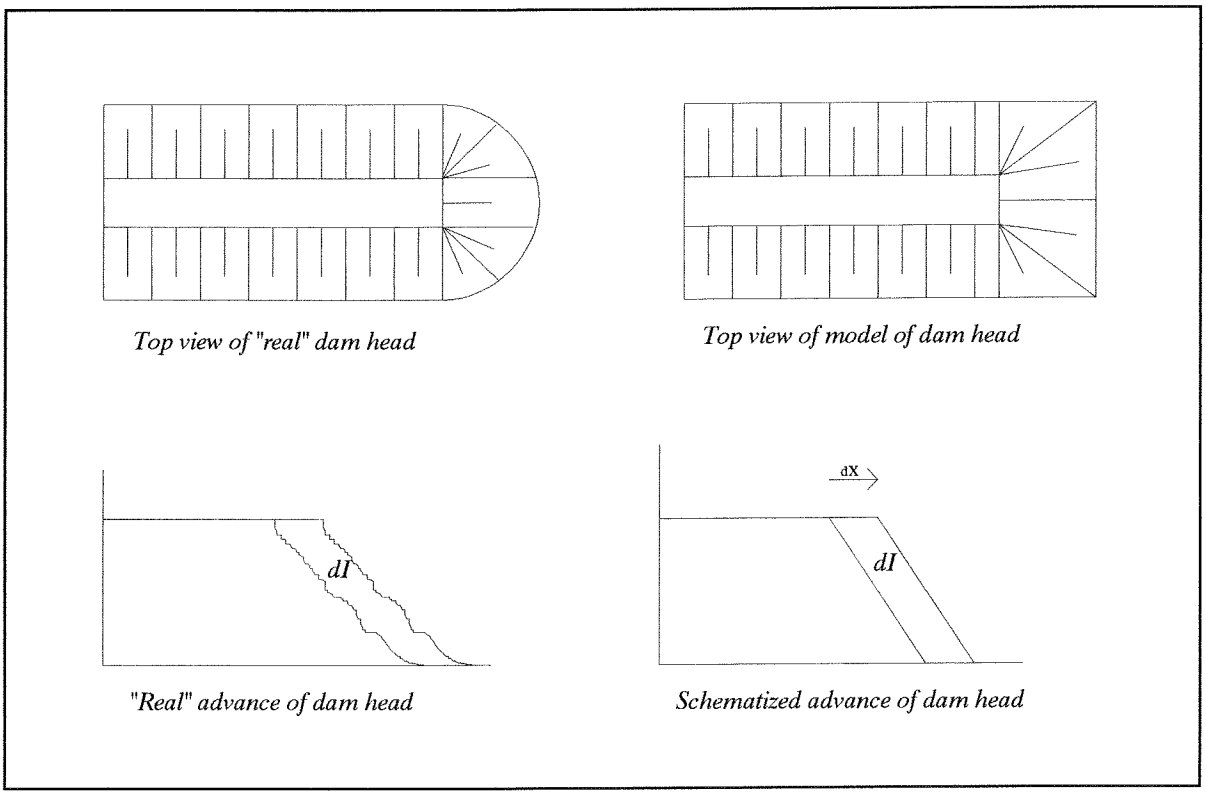


Figure 4.3: Schematisation of dam advancement

The relation between the dam advancement (*dX* for horizontal advancement or *dH* for vertical advancement) and the dumped volume (*dV*) is quite complex. The most important goal of these calculations is, that any amount of material, entered from any direction, can be translated in certain progress in the construction of the dam. In the following, details concerning the calculation of the dam advancement are given.

4.2 Horizontal construction

To calculate the process advancement, while the actual geometry and the quantity of dumped material are given, various stages and possible situations in the construction process have to be considered. For horizontal construction, there are four:

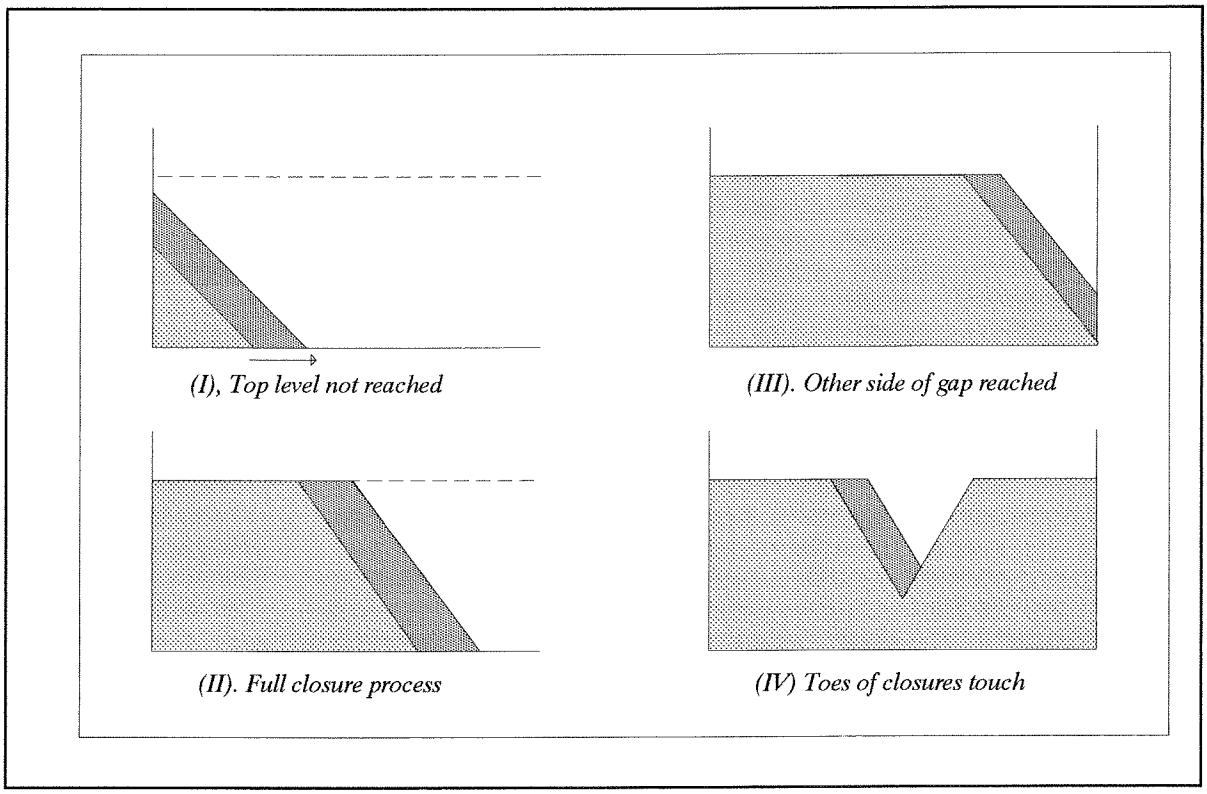


Figure 4.4: Stages in horizontal closure process

Also combinations of these layers are calculated - for example, when the toes of the structures touch but the maximum top level has not been reached yet.

4.3 Vertical closure from below water level

For vertical closure from below the water level, for example if the dam is made by dump vessels, there are three situations to consider:

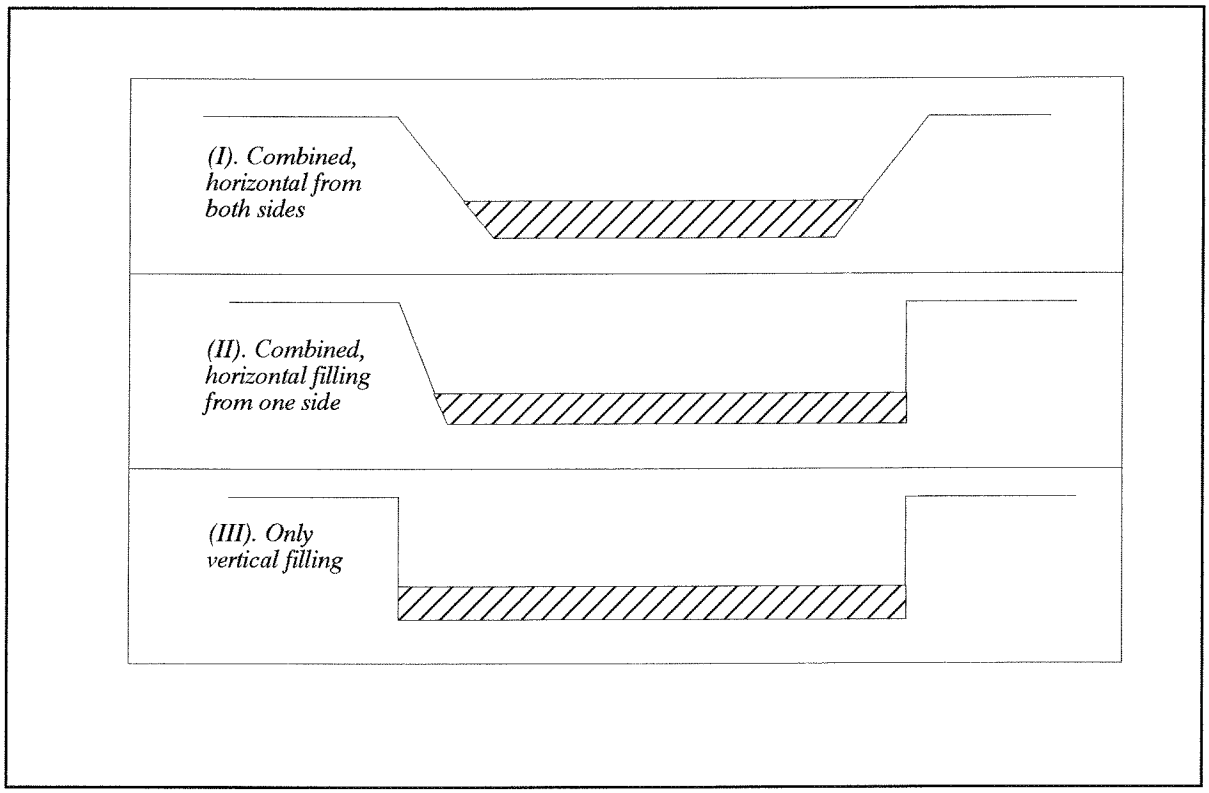


Figure 4.5: Vertical layers

The advancement of the construction should not be expressed in dX, but in the layer height dH. The three cases shown above only differ in whether there should be added prismatic edges to the layer.

In *Appendix B* the calculations for all advancement possibilities are displayed.

4.4 Vertical Construction from above water level

In case of vertical closure from above the water line, for example if the dam is made using a cable way, the calculations will be different from those in case the construction is done by dumping from vessels. Cross sections of both ways of construction have been made clear in figure *4.6*:

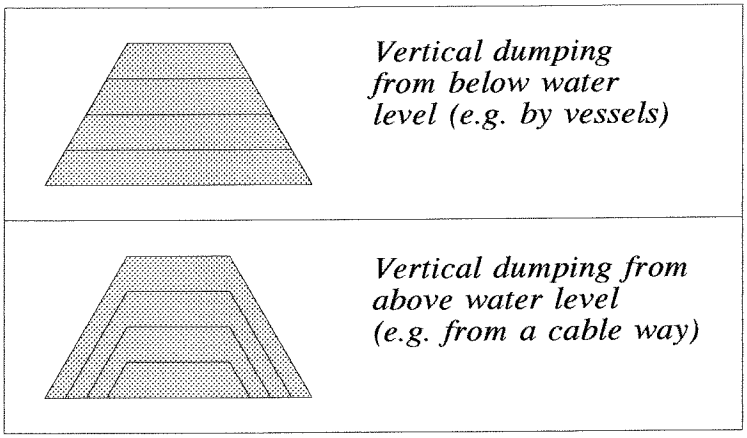


Figure 4.6: Comparison vertical dumping methods

The main difference between those two construction methods is, that in case of vertical dumping by vessels, the last stage of the dam will be constructed much faster than when a cable way is used. This can be seen in the figure; in the first case, the volume of the last dam layer is minimal, while in case of the cable way, the last dumped volume is spread over the whole dam surface.

Because of the different dam advancement, the dam calculations in case of a cable way will not be compatible with those when the closure is horizontally or by means of vessels. Hence, vertical dumping from a cable way can not occur at the same time as dumping using other methods. After a sill has been constructed using a cable way though, another method can be chosen by means of which the dam is finished. The program checks whether the sill is broad enough for the rest of the structure.

The dam advancement calculations for both cases have been shown in *Appendix B*.

4.5 Side and Head Angles of Dam

When the material for the closure is dumped, the slope of the construction has a certain angle which is mainly determined by the construction method and the size of the particle. For these angles, no theories are available yet. The calculations of the simulation program interpolate between some known combinations of construction method, particle size and slope angle; these combinations have been deduced from values which occur in practice, but can be changed in the program source. It should be stressed that the values for the slope angles are very rough estimations. More accurate data on slopes dumped in flows, can be added in future extensions of the program.

The values which actually have been applied in the program calculations, are presented in the following graphs. For sand filling by pipelines, this graph is:

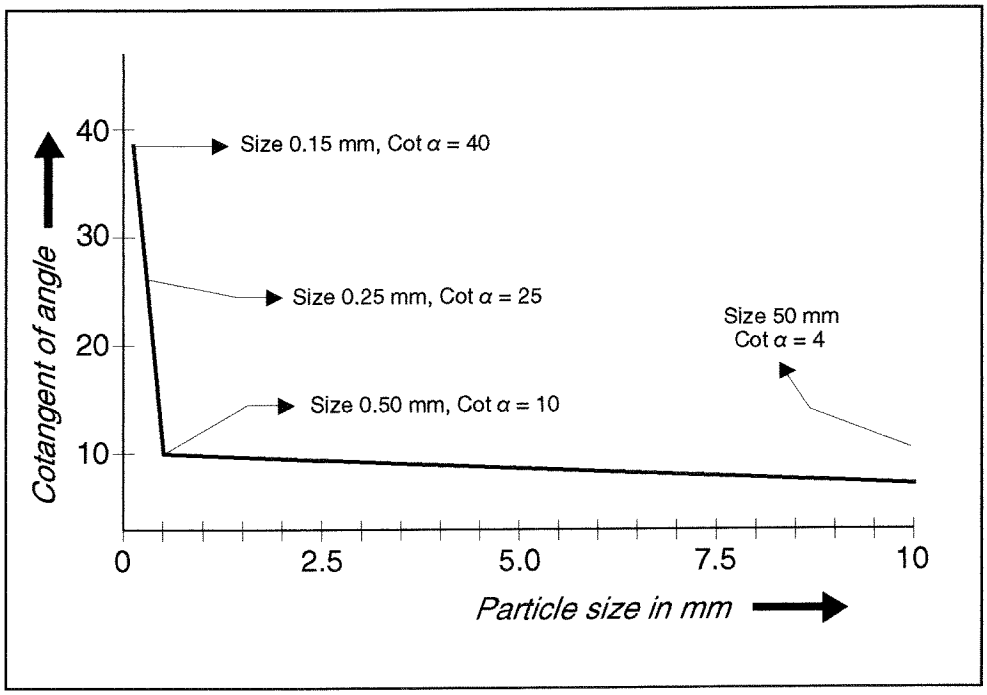


Figure 4.7: Slope angles vs particle size (sand)

The slopes when sand is filled by pipe lines are less steep than the slopes made by trucks or vessels for coarse materials:

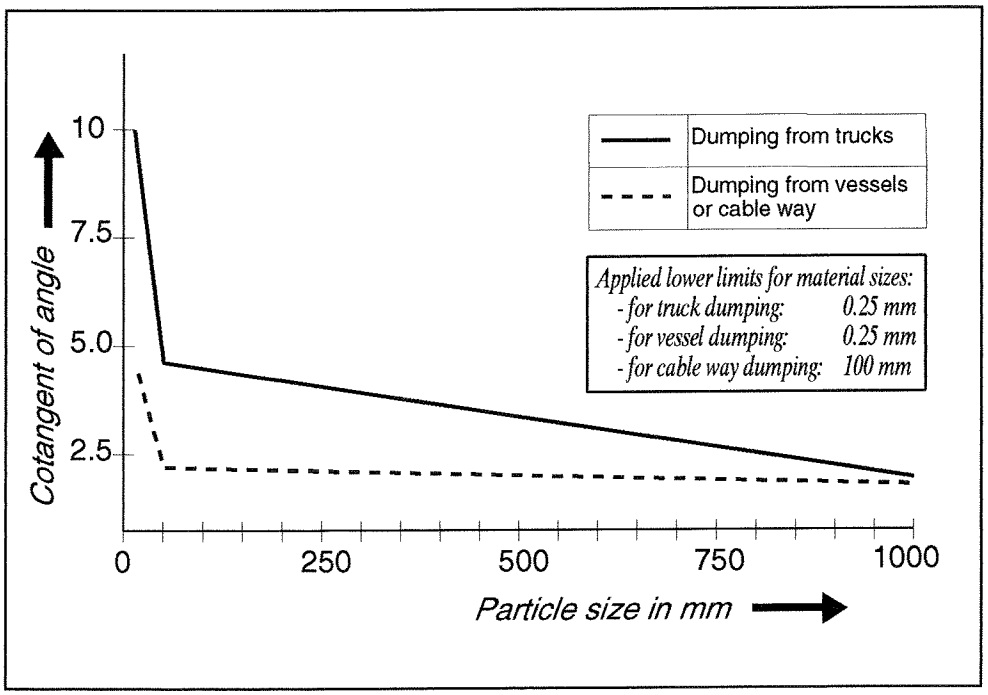


Figure 4.8: Slope angles vs particle size (larger materials)

4.6 *Repercussions of a Material Change*

Because the slope angles depend on the size of the dumped material particles, a change in material will cause the total volume of the dam to change.

In the modelling of the dam in the program, the advancement is indicated by the toe location. Together with the dumping angles, all other parameters of the actual dam can be compiled.

When the user of the program chooses another material size (always bigger than the last used material, and hence causing steeper slopes), the toe locations stay the same, and all of a sudden the top of the dam would seem to have advanced without any addition of material. In figure *4.9* this has been illustrated: the toe location (parameter for the dam advancement) has remained the same, but the volume that should be dumped before the calculations could continue, should not be ignored:

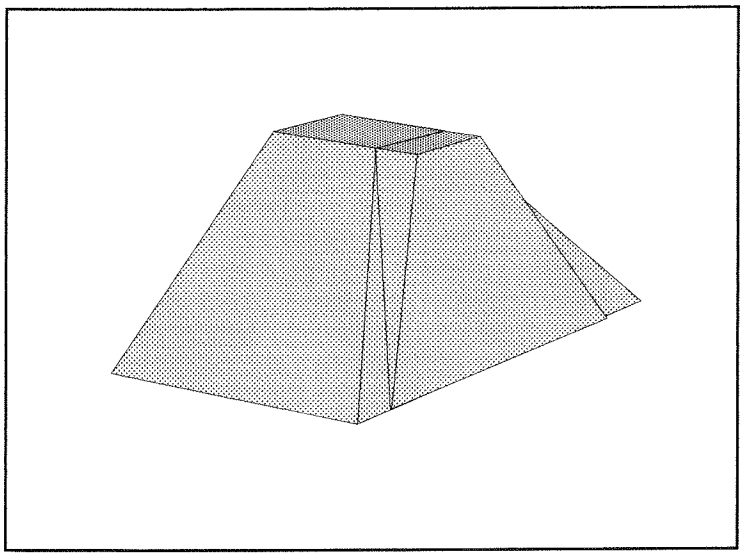


Figure 4.9: Volume correction

In *Appendix B* the calculation of this volume has been described.

5. Loss of Dam Material

5.1 Dam regression subject to erosion

The procedure that has been written to include the construction activities, contains the calculation in which a certain volume of dumped material (horizontally from left or right, vertically) can be transformed in a value for the construction progress. Each volume V which is entered into the process, is represented by an equivalent head or sill displacement ΔX or ΔH.

In order to calculate the *regression* of the dam construction due to erosion of the dam material, it seems usefull to rewrite this procedure so, that the displacements ΔX and ΔH can also result negative, when a negative volume V is entered into the process. This is the case when the eroded volume is greater than the volume of dumped material. The calculations as described in *Appendix B* have been adapted to be applicable to this purpose.

The volume V that is entered in the advancement calculations can be written as:

$$\Delta V = \Delta V_{pos} + \Delta V_{neg}$$

in which ΔV_{pos} is the volume of dumped material, and ΔV_{neg} the volume of eroded material.

Reason for erosion

Because of the decrease of the flow profile by the dumping activities, the velocities at the location of the dam will be higher than the velocities in other places of the channel. This causes the eroding capacity in the reduced section, which depends on the power of the velocity, to be much greater. In figure *5.1* this means, that the eroding capacity *S2* is much greater than *S1* and *S3*.

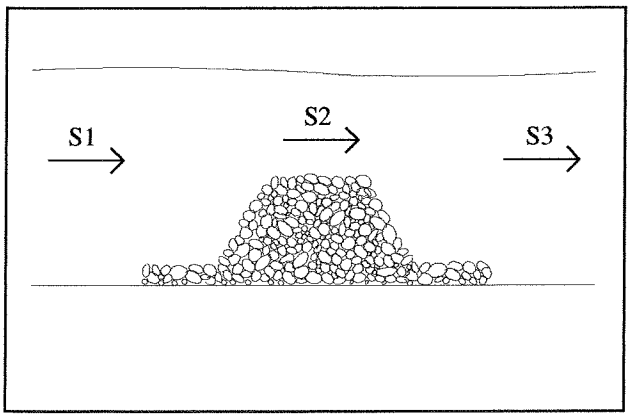


Figure 5.1: Erosion of material

In the program, the velocity in the gap is assumed to be distributed equally. In the formulas for eroding capacity, this means that for the velocity is taken $u = Q/A$.

5.2 *The Transport Formula*

For the calculation of the erosion *S2*, there are various possibilities [Van Rijn, 1989]. Because of the liberty of the program user in choosing whatever material he wants to construct the dam, purpuse has been to look at the possibility to calculate all losses of material using one formula. Another, more important reason for this thought is, that for bigger grains there are no such transport formulas as the ones which exist for sand. For those large grains, the only existing relationships focus on the threshold of movement. In the simulation program though, not *when* the larger grains start to move, but also *how many* grains are removed is subject of interest.

Extrapolation of the existing transport formulas for small particles has been proposed, to be able to use those for larger grains also. If the considered range for the grain sizes is regarded (0.15 mm to 2 meters), it is obvious that this extrapolation would lead to a very rough schematisation and results that are nothing more than estimations.

In general, transport formulas have the following form:

$$s = \frac{S}{B} = M u^n$$

5.2.1 Application of Engelund & Hansen's Relationship

Engelund & Hansen found [Engelund/Hansen, 1967] for the general transport formula as shown above:

$$q_s = \frac{0.05\ u^5}{\Delta^2 \sqrt{g}\ d_{50} C^3}$$

in which	q_s	*Erosion*	*in* $m^3/s/m$
	Δ	*relative density* $(\varrho_s - \varrho_w) / \varrho_w$	---
	ϱ	*density (water or stones)*	N/m^3
	u	*current velocity*	m/s
	g	*gravity*	m/s^2
	d_{50}	*average particle size*	m
	C	*Chezy roughness*	$\sqrt{m/s}$

In this formula, the value for the *Chezy roughness* depends on the pattern of the bottom. The occurrance of ripples and dunes, will affect the rougness. In the simulation program, the user can define a value for this rougness, and this value is applied in the calculations. When coarser material is used, the bottom pattern is not important and a value for C could be obtained as

$$C = 18 \log \frac{12R}{k}$$

with	R	*hydraulic radius*	m
	k	*equivalent sand-roughness*	m

In practice, for k values of 2 - 3 *D50* are applied, while for broad channels R can be assumed to be equal to the water depth.

When the formula of Engelund & Hansen is used to calculate the movement of larger stones, a problem would exist in the fact that the formula gives a sediment transport for each current velocity, ignoring any threshold of movement. Many investigators found that for movement a certain *critical shear stress* is needed; their statement was that at shear stresses smaller than this, there would be no movement at all. One of these investigators has been Shields [Shields, 1936], who defined an expression for a critical shear stress which if exceeded would cause motion:

$$\psi = \frac{\tau_c}{(\gamma_s - \gamma)d} \qquad \psi_{cr} = f\left(\frac{u_* d}{\nu}\right)$$

with:

ψ	*Shield's parameter*	---
ψ_{cr}	*Shield's threshold of movement*	---
τ_c	*Critical shear stress*	N/m^2
γ	*Specific weight*	kg/m^3
d	*Particle side*	mm
u_*	*Shear velocity*	m/s
ν	*kinematic viscosity*	m^2/s

The shear parameter is a function of the Reynolds number. Shields did experiments on the critical shearstress, as shown in figure *5.2*:

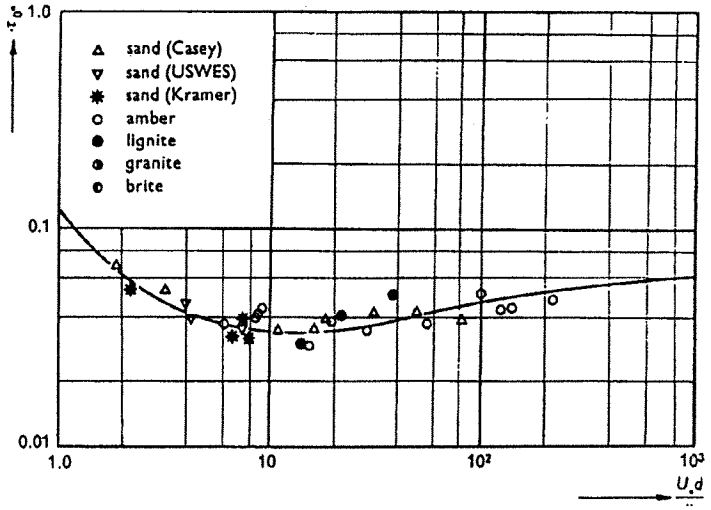


Figure 5.2: Shields' threshold of movement

Engelund & Hansen's emperical relation can be applied in combination with Shield's condition for movement. In the case of small grains Shield's condition doesn't affect the transports very much: the critical shear stress for these grains is reached at low velocities. In figure *5.3* an example is given for grains of 0.001 in a waterdepth of 5 meters.

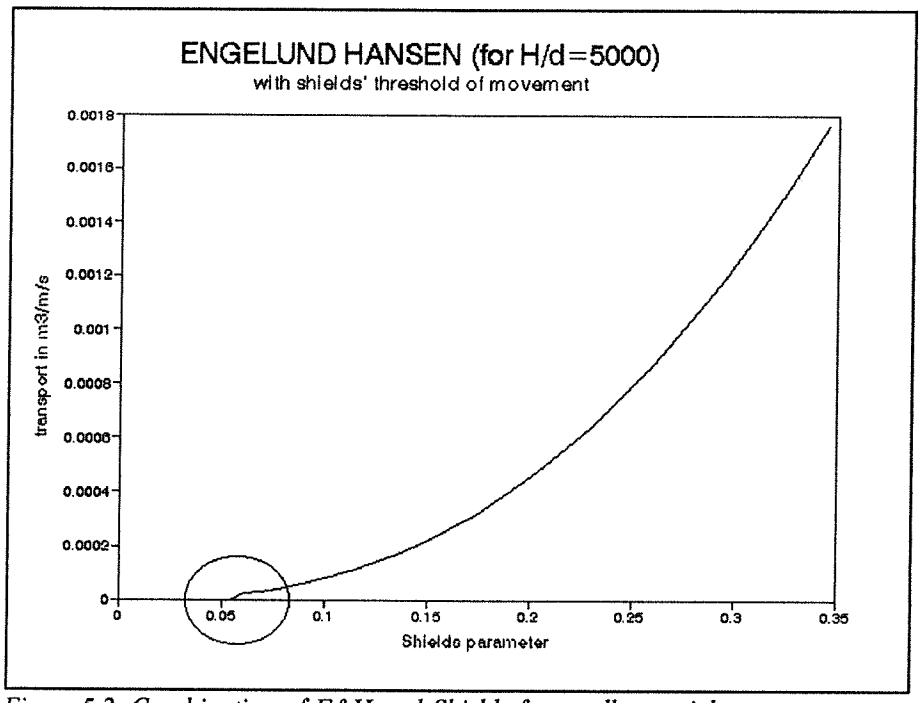


Figure 5.3: Combination of E&H and Shields for small material

If the combination of Engelund & Hansen's transport formula and Shields' condion for movement is made for the movement of bigger stones though, this picture gets different. Then, for a rather large range of the velocity, there would be no movement of sediment at all; but as soon as the critical bottom shear stress is achieved, the movement would suddenly have a rather high value:

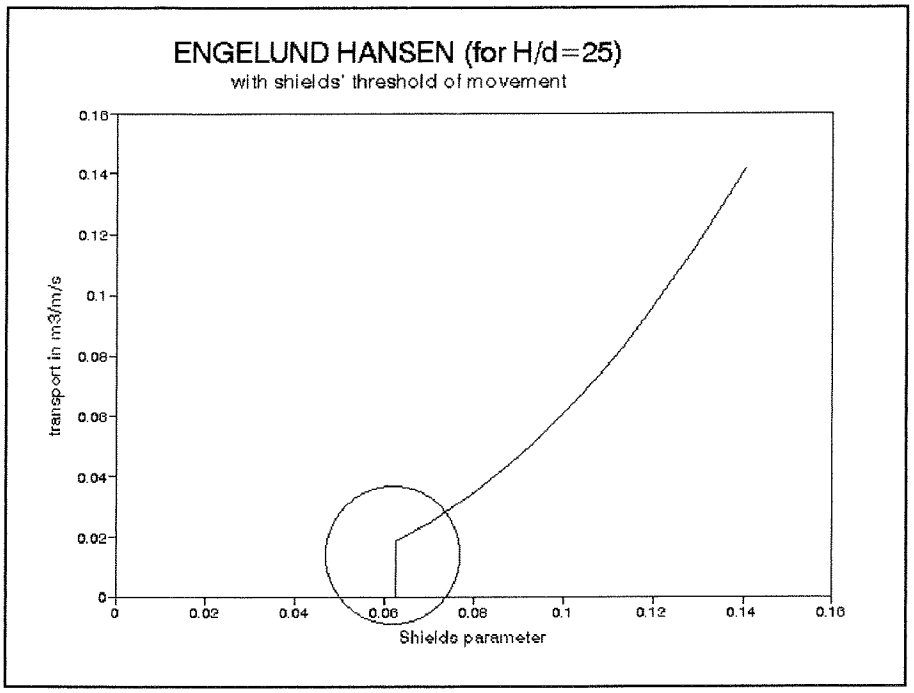


Figure 5.4: Combination of E&H and Shields for larger material

The sudden increase of sediment transport as soon as the critical shear stress has been reached, is something that also follows from experiments. But the assumption that this occurs immediately at a fixed shear stress is rather rough, hence it would be interesting to take a closer look to the transport near this critical shear stress.

5.2.2 Paintal's Formula for transport at small shear stresses

In 1970 A.S. Paintal [Paintal, 1970] presented his studies on the problem of defining critical flow conditions with regard to the initial instability of bed material particles, in which he compared new measurements with the existing concepts on the threshold of movement.

Paintal's experiments were done using bigger grains at low transports (D_{50} = 2,5; 7.95 and 22 mm). Investigating the correlation between transports and shear stresses, he found a correlation, in which at certain shear stress and transport the gradient changed. Paintal showed the two branches in a figure:

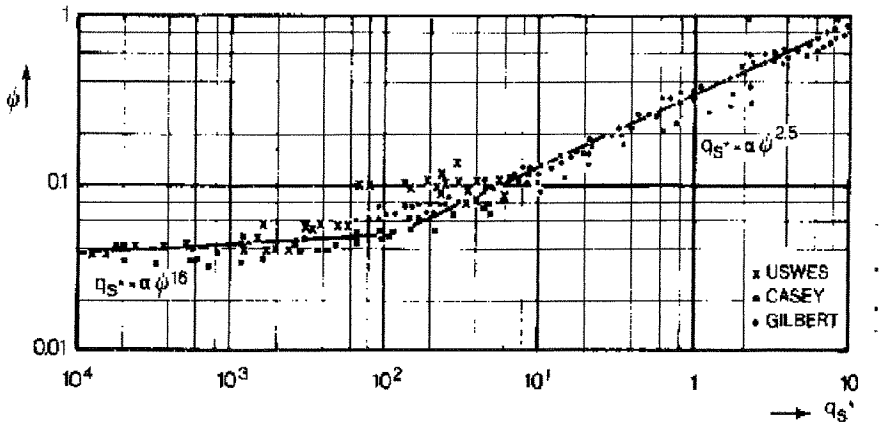


Figure 5.5: Variation of bed load transport for various shear values

For the right branch, for which Paintal did not execute new experiments but used results of previous investigations. Paintal found a 2.5th power relation between shear stresses and transports, just like *Engelund & Hansen*; in that case, the transport depends on the 5th power of the velocity. Because no new experiments were done to come to this 5th power relation, both branches of the graph can be considered separately. The left branch describes the transport at low shear stresses. The curve Paintal fitted through his own measurements, which is the left branch of the figure above, was:

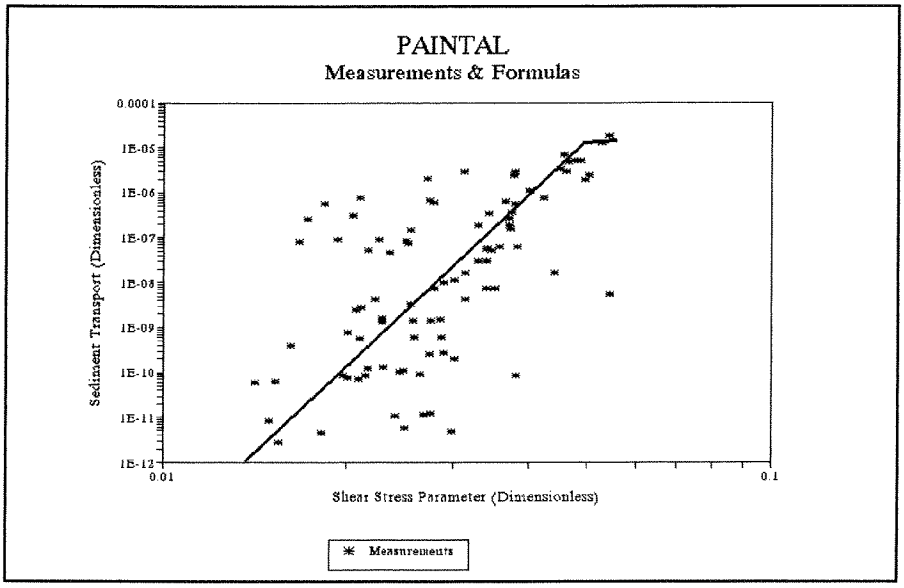


Figure 5.6: Paintal's measurements and formulas at low shear stresses

Paintal concluded that the transports at low shear stresses were related to the 16th power of the shear stress, so that at low shear stresses there is a 32th power relation between velocities and transports. The most important conclusion of Paintal was, that transport does not simply commence at a certain *critical shear stress*, like Shield proposed, but that also below this critical shear stress there are measurable transports.

5.2.3 Combination of Formulas of Engelund & Hansen and Paintal

In *Appendix C* is explained how the formulas of *Engelund & Hansen* (for normal transports) and *Paintal* (for low shear stresses) have been applied together in the simulation program.

5.3 Crest factor with respect to Flow Regimes

5.3.1 Theory on Flow Regimes

The experiments for the investigations of *Paintal* and *Engelund & Hansen* were executed in a long uniform flow. In the calculations for the program though, the situation is different because the flow is obstructed by the closure works. The flow regimes which are caused by this obstruction, should be considered in the calculations.

As mentioned in paragraph 5.2, for rockfill closure dams no relations are available for the determination of the transport of material. For the outline design of rockfill dams, relations for the threshold of movement of material have been elaborated. These formulas are based on the tailwater parameter, $h_b/\Delta D$, in which h_b is the tailwater elevation relative to the dam crest, and ΔD is the stone size parameter. The mentioned flow regimes which occur during the closure of a tidal inlet, have been considered in the determination of these stability relations. Four typical flow regimes have been discussed, dependent on $h_b/\Delta D$:

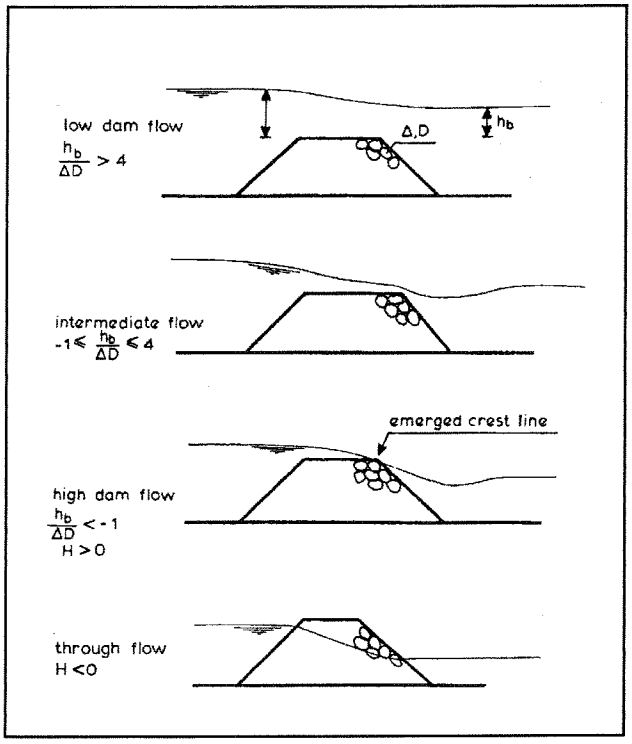


Figure 5.7: Flow regimes

In this distinction of flow regimes, besides of the overtopping stability, also the porous dam flow has been considered. For the four typical flow regimes this yields:

* *Low dam flow ($h_b/\Delta D > 4$):* drowned flow, no influence of porosity;
* *Intermediate flow ($-1 < h_b/\Delta D < 4$):* free flow, flow penetration into the porous crest of the dam;
* *High dam flow ($h_b/\Delta D < -1$ and $H > 0$):* Submergence of downstream crest line, rough shute flow on the inner slope of the dam;
* *Through flow ($H < 0$):* the full discharge passes through the dam body, outflow on the inner slope.

The porosity of the closure dam has not been included in version 1.0 of *CLOSIM*. It would be interesting though, to pay attention to the various flow regimes. The effect of the regimes on the stability of the dam material, should be translated into an effect on the material transports in the simulation.

From the investigations resulted that in the *low dam flow situation*, a uniform flow approach fits the mean data well. For the outline design of the dam this means, that criterion like *Shields'* (see paragraph 5.2) could be applied; for the simulation program it implicates, that the transport formulas for uniform flow can be used without any adaptation.

After the free flow situation is reached, raising the dam will lead to an increasing flow attack on the downstream part of the crest and the inner slope, even though the discharge does not increase significantly. It was found [DHL, 1985] that this is caused by flow penetration into the porous crest, which increases the local current velocity up to values far beyond the critical velocity at the instant of free flow. In the figure is shown how H and q can be used to predict the stability. It shows that in this flow range $(-1 < h_b/\Delta D < 4)$, the stability decreases significantly when the dam is raised:

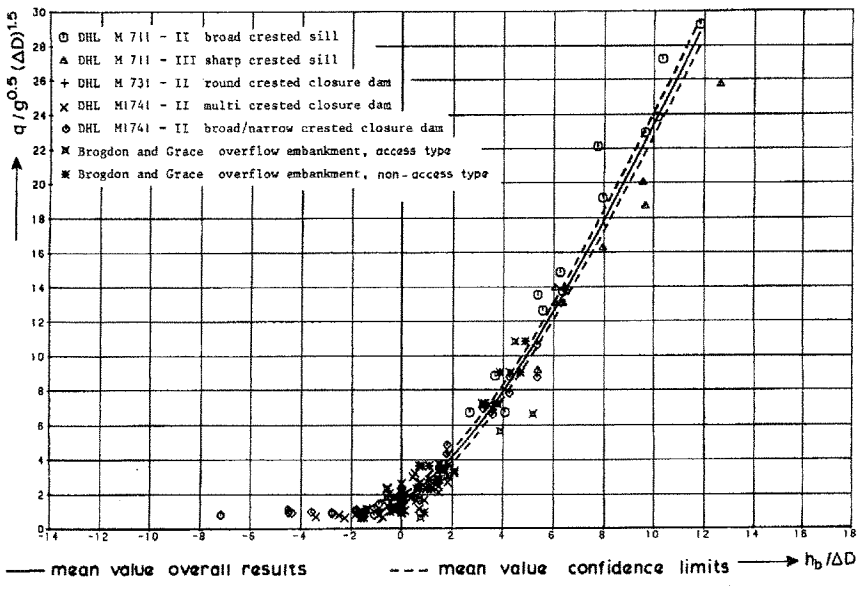


Figure 5.8: Material stability on raising sill

For the existing stability criteria, various correction factors have been proposed. If the critical velocity $\sqrt{(2/3)gH}$ is substituted by the discharge $(1.7\ H^{1.5})$ divided by the actual tailwater depth h_b, while a water depth correction equal to D is added to account for the flow penetration [Izbash, 1932], the underestimation of the current velocity is compensated to some extent.

When the construction height is increased further, the *high dam flow situation* is achieved. In this case the development of the current velocities is even less defined, because of the extremely rough type of flow. The stability of the inner slope proved to be described fairly well by *Knauss'* relationship for steep shute flow [Knauss, 1979] , in which the slope angle of the dam is entered:

$$\frac{q}{\sqrt{g}(\Delta D)^{1.5}} = 1.48 - 1.87 \sin\alpha$$

For a uniform flow, the right term of this relationship is about 1.7 [Izbash, 1932]. The correction for high dam flow provided by Knauss, could also be useful to correct the material transport in case of a high dam flow.

In a *throughflow* situation there is no overtopping, and only the porous flow is determining. In the simulation program, the dam porosity has not been included yet, and the flow through the closure gap is assumed to be zero when there is no tailwater. In practice, the throughflow situation will normally be stable, if the inner slope is not too steep, because of the highly reduced discharge [DHL, 1985] .

Any additional wave attack is taken into account by adding 1/3 of the significant wave height to the overtopping height (this extra height does not influence the discharges).

5.3.2 Application in the Calculations

The consideration of the various flow regimes with respect to the dam crest, only regards stability criteria for the threshold of movement. As said, in the simulation program the *quantities* of the material transport have to be calculated. In order to apply the theories on the flow regimes for this purpose, the extra load has been interpreted as an increase of the local velocity.

1. For the stability calculations for *low dam flow*, *Izbash'* relationship has been widely used without adaptations. Some other proposals have been done [DHL, 1985]; for the simulation program, the uniform flow theories have been applied with a velocity factor of *1.0* (no influence on the flow pattern).

2. For *intermediate dam flow* a free flow condition, in which the flow gets critical, can be reached. When the tailwaterlevel is lowered relative to the dam crest, the discharge over the crest is not affected. But locally the velocity still

increases with the lowering of the tailwater level because of the streamline curvature and flow penetration into the permeable crest. For various stability parameters investigations have been done [DHL, 1985]. Some of these investigations considered a straight-forward application of the Izbash criterion, which then resulted with an enlargement factor γ in the right term of the equation [DHL, 1985]:

$$\frac{H}{\Delta D} = \frac{3}{2}\frac{(1.7)^2}{\gamma}$$

For the enlargement factor γ was found:

$$\gamma = \sqrt{2.4 - 0.35\, h_b/\Delta d}$$

This relation has been illustrated (for γ^2) in the graph below:

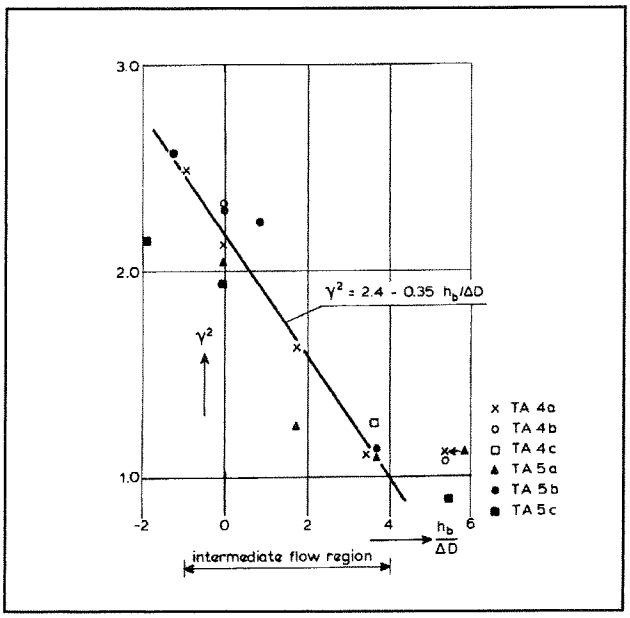


Figure 5.9: Velocity factor for intermediate flow

For the relation that exists between H and u, this enlargement factor can also be applied for the velocities, which causes:

$$u_0 = \gamma \sqrt{\frac{2}{3} g H}$$

For intermediate flow, this multiplication factor has been applied in the simulation program. It must be mentioned, that to apply this theory, certain porosity of the structure has been assumed. In future versions γ could be restricted for non-porous materials:

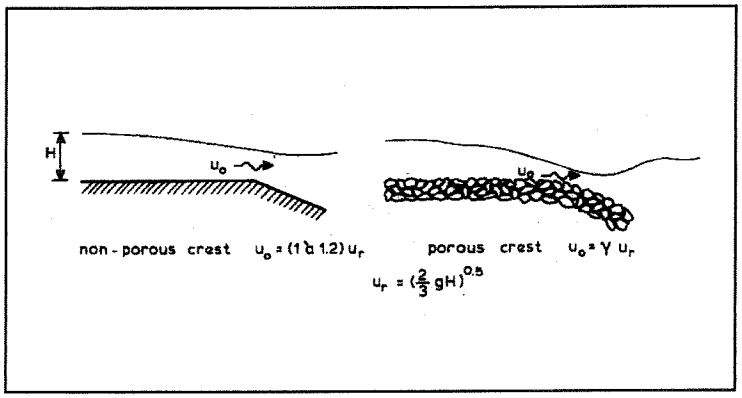


Figure 5.10: Velocity factor vs. crest porosity

3. *High dam flow* and *through dam flow* are only possible when the structure is porous. In future versions of the program the porosity will be included (see *chapter 7*), but in the actual version of *CLOSIM* the discharge in the flow gap is zero as soon as the tailwater level above the crest of the structure is zero.

5.4 Construction Area exposed to erosion

In the situation of a horizontal closure from two sides (which can be combined with vertical closure), the area exposed to erosion can be defined as:

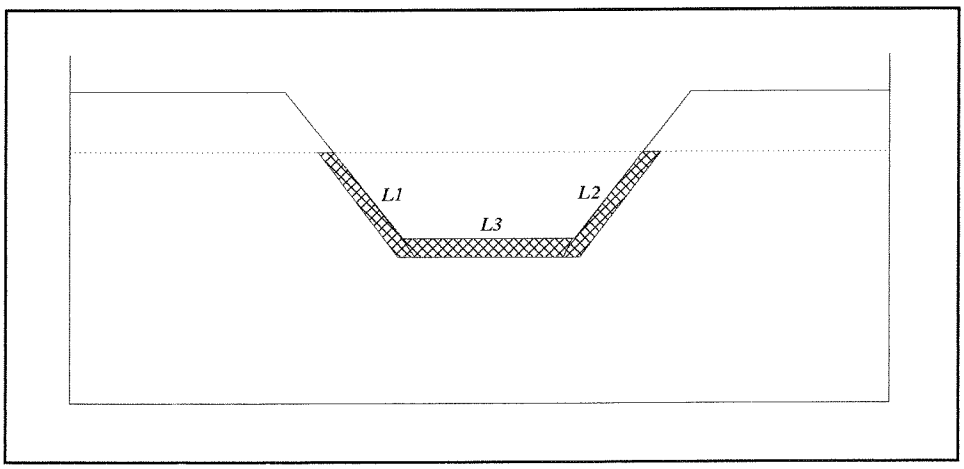


Figure 5.11: Area exposed to erosion

The total length of this area is given as $L = L1 + L2 + L3$. The total volume of the erosion (in m^3 / s), can be calculated as $S1 = L.s$. The fact that also s changes for the eroding zones, due to contraction effects, has not been considered yet.

As said, the eroded volumes can simply be applied as a negative construction activity. Hereto, the *'negative volumes'* are entered in the construction procedure:

for horizontal construction from left side: $$\Delta V_l = \frac{S1.L1}{L}$$

for horizontal construction from right side: $$\Delta V_r = \frac{S1.L2}{L}$$

for vertical construction: $$\Delta V_v = \frac{S1.L3}{L}$$

5.5 Erosion of Bottom Material

Where the bottom material is exposed to the currents, for example between the dam heads of a sand closure from two sides or if there is no dam at all, the bottom material of the channel will erode. It has been mentioned already that the bottom material is assumed to be granular and without cohesion. The erosion of this bottom material can be calculated from within the same formula as where the erosion of the dam is calculated, but two things should be kept in mind:

* The bottom material has another *grain size.*

* The erosion of bottom material will take place *over a certain distance.*

Both aspects have been illustrated in figure *5.12*, where the difference between slices of dam erosion and the slices of bottom erosion is shown.

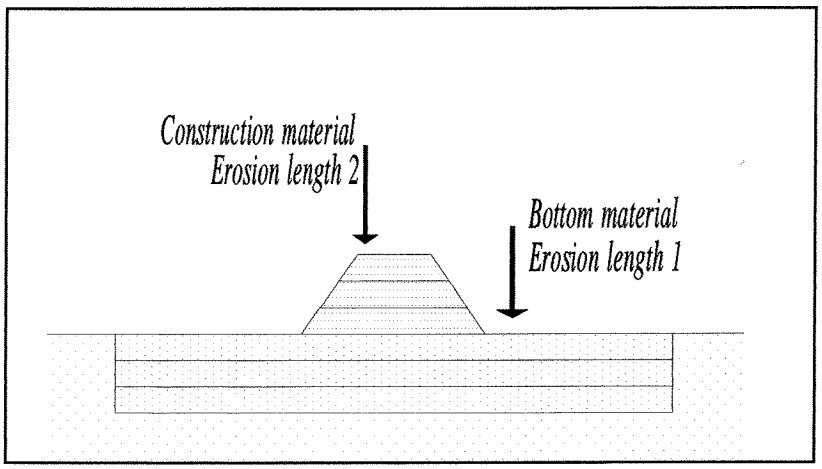


Figure 5.12: Erosion of dam- and bottom material

For the length over which the dam material is eroded if there is no dam at all (*erosion length 1* in the picture), the assumption has been made that this occurs over five times the initial gap width ($L_E = 5\ W_{GAP}$):

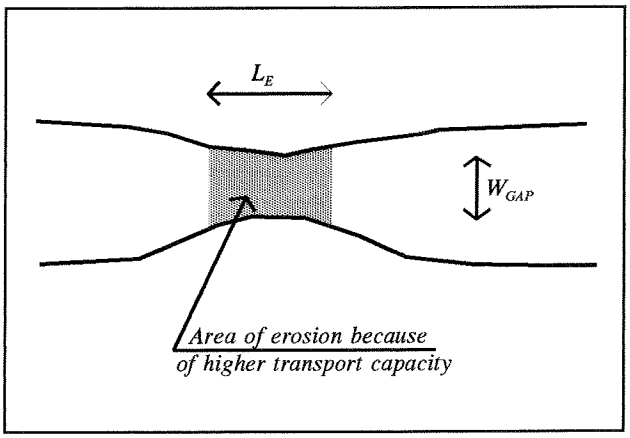


Figure 5.13: Erosion of bottom material

In the area in the picture erosion takes place. In the same way in a broader (or deeper) area, accretion can occur. Both erosion and accretion will finally cause an equilibrium situation in the tidal channel.

5.6 *Instability caused by a vertical Dumping Inaccuracy*

In case of vertical closure, special attention must be paid to the dumping accuracy. In the schematisations up till now, the top of the vertically dumped layers has been considered as perfectly horizontal. This will never be the case, especially as the dumping of material is not a continuous process. The top of the structure could be like shown:

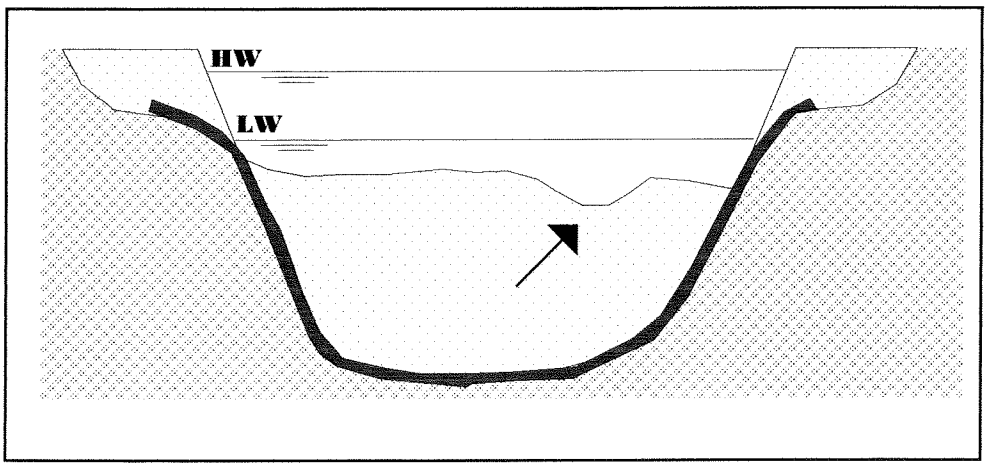


Figure 5.14: Irregularity in vertical dumping

The dumping inaccuracy as indicated, can start a failure mechanism that will undermine the dam very fast. As soon as the weir flow above the crest of the structure gets critical, the velocities do not increase any further. It can occur though, that while the weir flow is critical, the flow above the dumping inaccuracy is not. Then, the velocities would be much higher there, and could remove the particles in the deepest place of the dumping inaccuracy; because of that, the gap would be deeper, which would cause the velocities to be even higher, and so on. In this way, a vertical gap would develop and cause the vertical closure to become a horizontal closure, with its higher velocities:

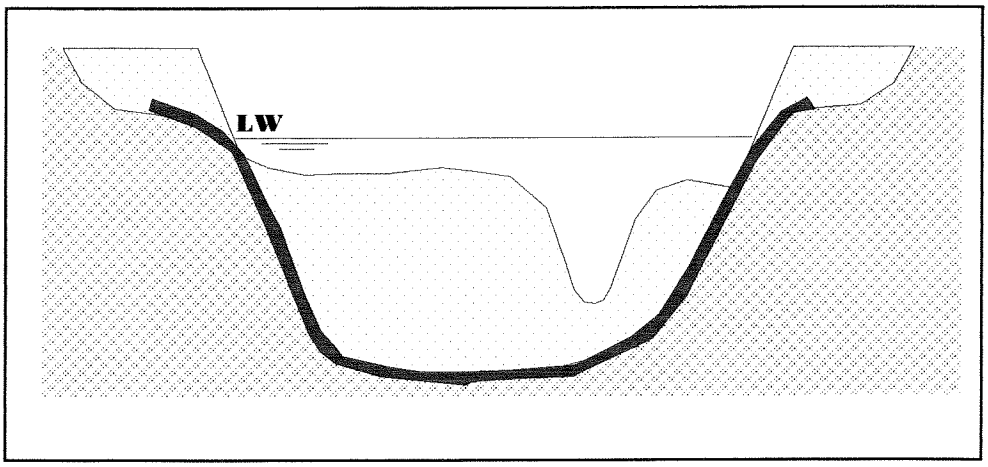


Figure 5.15: Vertical instability

In the velocity diagram of horizontal and vertical closures, this mechanism can be made clear. When the flow regime in case of vertical closure has become critical, the velocities are lower than they would be in case of horizontal closure with the same reduction of the flow profile. The failure mechanism is a change from one closure method to another, which has been indicated in figure *5.16*:

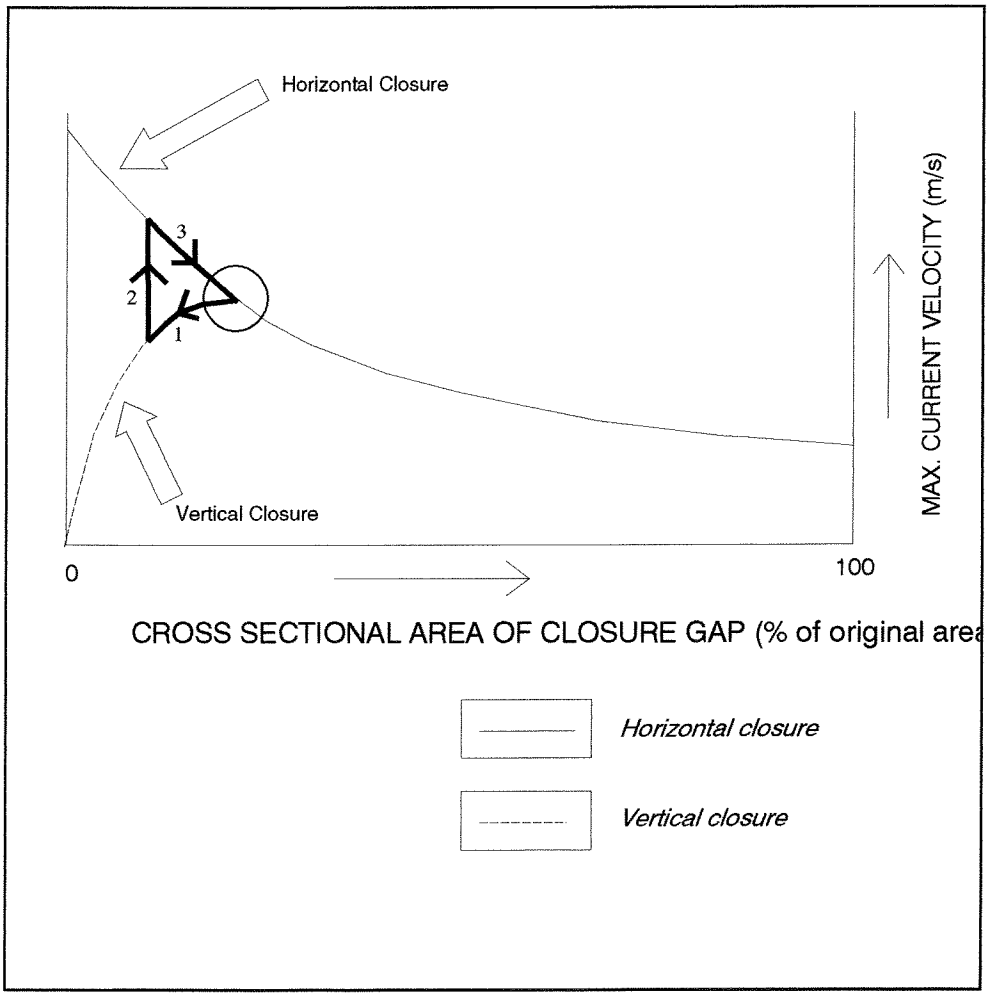


Figure 5.16: Mechanism for vertical instability

The failure mechanism is indicated by the numbers in the figure. When the vertical closure advances, critical flow is achieved and the velocity decreases according to line *1*. The cross sectional area decreases. Line *2* though, indicates the situation when a vertical dumping inaccuracy gets instable. At the location of the inaccuracy, the velocity is higher, according to a normal weir situation as would occur when closing horizontally. This higher velocity can cause the cross section to increase (line *3*). At last, the situation will remain as indicated with the circle, just at the diversion of horizontal and vertical closure (or of normal and critical flow) in the graph.

6. Scour behind a Bottom Protection

6.1 Scour Holes

During closure, the dumping of large stones, concrete cubes or other flow resistant elements into the gap, reduces its cross-section. The stability of the adjacent bed, when it consists of erodable material, is endangered by scour and there is a need for bottom protection to ensure a stable foundation to the closure dam. This bottom protection decreases the threat of scour in two ways. Firstly, it causes the scour holes to develop at some distance from the construction, so that it cannot be undermined by them; secondly, it provides the necessary distance for the flow to spread over the whole section again and to become less turbulent. In this way, the scour holes occur at a certain distance from the structure, and they are less deep.

The development in time of scour holes behind the bottom protection has been illustrated in figure *6.1* [Vinje, 1975]:

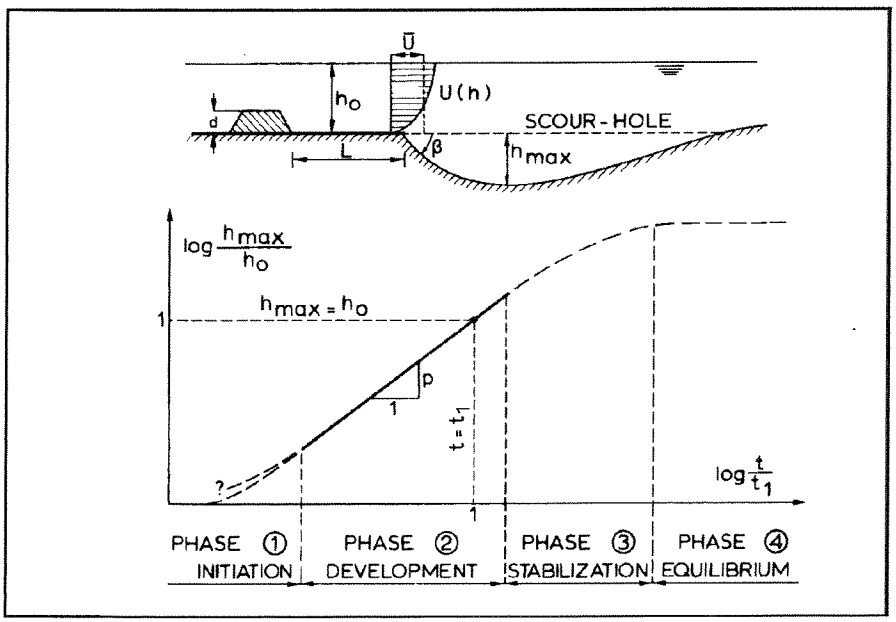


Figure 6.1: Scour hole development

The graph shows, that after some time the depth of the scour holes reaches a state of equilibrium.

Breusers [Breusers, 1965] derived an expression for the scour depth behind a protection:

$$h_s(t_{hours}) = \frac{(\alpha \bar{u} - u_c)^{1.7} h_0^{0.2}}{10\Delta^{0.7}} t^{0.4}$$

in which

h_s	the actual depth of the scour hole (in m)
t	the time (in hours)
h_0	the initial depth of the gap (in m)
α	a factor for the turbulence (-)
u	the depth-averaged velocity (m/s)
u_{cr}	the critical velocity (m/s)
Δ	the specific mass factor, $\Delta = \varrho_s / (\varrho_s - \varrho_w)$ (-)

The turbulence of the flow is expressed in the calculations as an increase of the velocity, multiplied with a turbulence factor α. On each location at the end of the bottom protection, this factor is defined. But especially at the dam head (horizontal construction), where the velocity gradient is high (sudden change from $v \approx 0$ to $v = Q/A$), a mixing layer will occur in which locally the turbulence will be higher.

6.2 *Calculation of Turbulence and Velocity*

In case of horizontal closure, the dam head will progress. The location where the mixing layer touches the soil shortly behind the bottom protection, changes constantly. For every location behind the protection, in every time step the turbulence α and velocity u change.

In order to be able to calculate the scour for the varying velocities and turbulences that occur in time, the end of the bottom protection has been divided into a certain number of sections; the scour will be considered in each of these intervals:

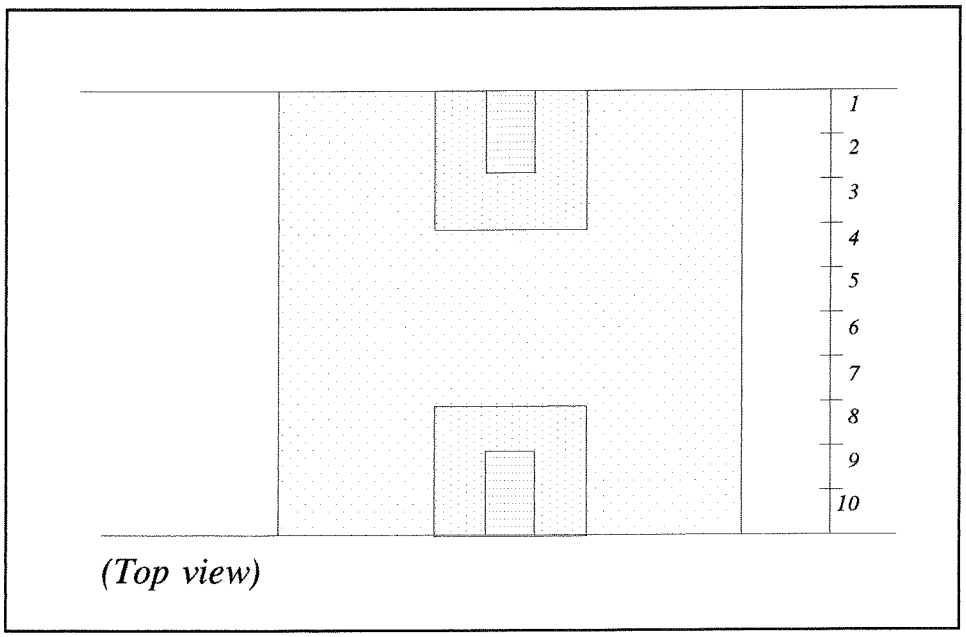


Figure 6.2: Bottom protection sections

For each of these intervals, the velocity u and the turbulence factor α are determined, and according to Breuser's formula the scour depth is calculated.

Vertical Closure

If the closure is entirely vertical, the velocity u, the turbulence α and the scour duration t are equal for all intervals. The resulting scour holes then will all have the same depth. Still, for each of the section borderlines the depth is stored, because later on changes in α and u might happen (due to the hydraulic conditions and the closure activities in the gap), so that the increasement in scour might differ in every time step.

Horizontal and Combined Closure

As soon as the profile is restricted horizontally, the scour depth contour behind the protection will not be uniform. This happens for example when after vertical construction of a sill, the closure is finished horizontally. The turbulence α will be *locally* higher as a direct result of the vortex street that occurs due to the dam head contraction. Also in this case α is a linear function of the width decrease. In the program, both turbulence factors are summed.

In the situation of horizontal construction, three zones can be distinguished [Vinje, 1968]:

1. A *shadow zone*, in the protection of the constructed dam, in which the product αu is assumed to be zero;

2. *The flow zone*, in which u is defined as the total flow, divided by the flow profile at the end of the bottom protection; α is given as a function of the depth decrease and the length of the bottom protection;

3. A *zone of local turbulence*: the mixing layer. In this zone the flow has extra turbulence, which causes an increase of α. Within this mixing layer, a diffusion process converts the velocities from zero (in the shadow zone) to u (in the main flow). This velocity conversion is assumed to be linear in the program.

These different zones for the calculation of u and α have been shown in figure *6.3*:

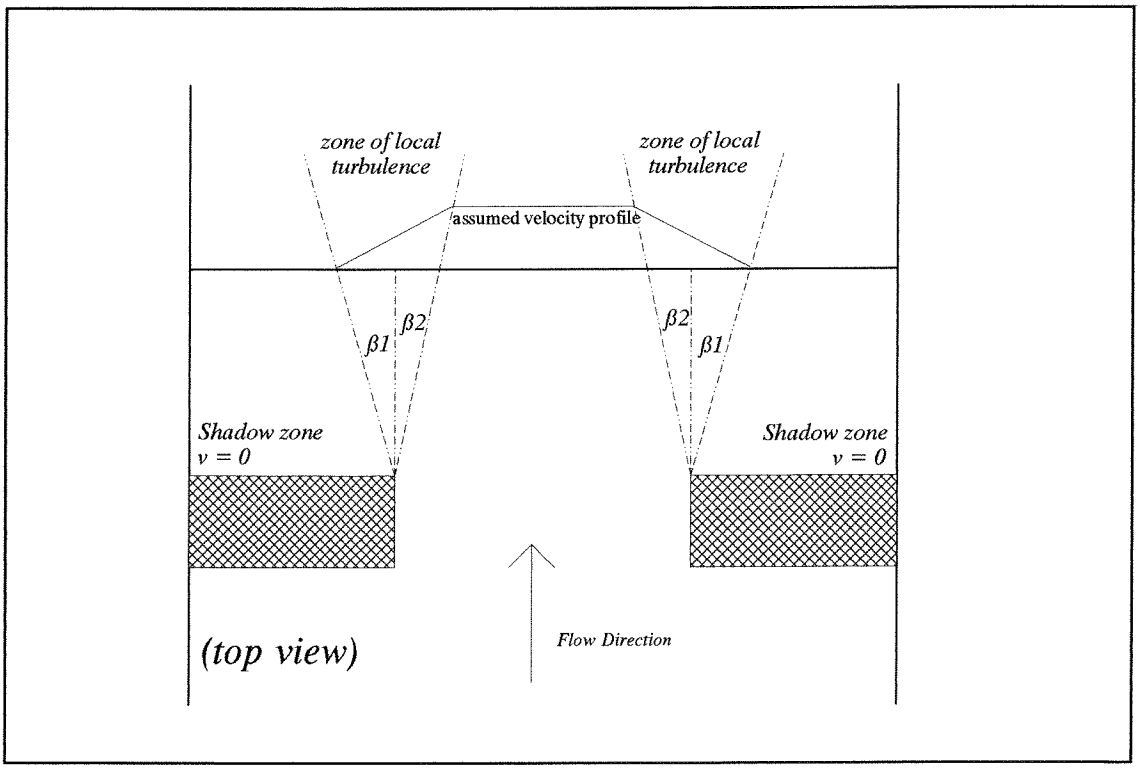


Figure 6.3: Various flow- and turbulence zones

In the figure, the *vortex street* (causing the zone of locally higher turbulence) is indicated by two angles, $\beta 1$ and $\beta 2$. For these angles, assumptions have to be made. In the actual version of the simulation program, $\beta 1$ has been taken as *1 : 6* and $\beta 2$ as *1 : 8* [Breusers, 1965].

In every time step, for each of the interval borders it is determined whether it is in a shadow zone, a zone of local turbulence, or in the flow zone. With the resulting α and u, the contribution to the scour depth is calculated.

Determination of the turbulence factor α

The turbulence factor is calculated as a function of L and h_0 [DHL, 1981] according to:

$$\alpha(\frac{L}{h_0}) = 1.5 + (1.57\ \alpha_{10} - 2.35)\ e^{-0.045 L/h_0}$$

In this function, a determined value α_{10} is used, describing the turbulence factor for $(L/h_0) = 10$. This factor α_{10} is a function of the *depth restriction* and of the *width restriction* in the flow gap [DHL, 1985]. In the program, this has been split up in a *profile restriction*, leading to a factor α_1 and a *width restriction*, leading to a turbulence factor α_2 for the contraction effects at the dam head location. These contributions to the turbulence are summed, and yield:

1. *For the shadow zone:* $\alpha = 0$;
2. *For the flow zone:* $\alpha = \alpha_1$ *(profile restriction);*
3. *For the zone of local turbulence:* $\alpha = \alpha_1 + \alpha_2$

The range for both turbulence factors can be set in the program source. The actual values have been given in figure *6.4*:

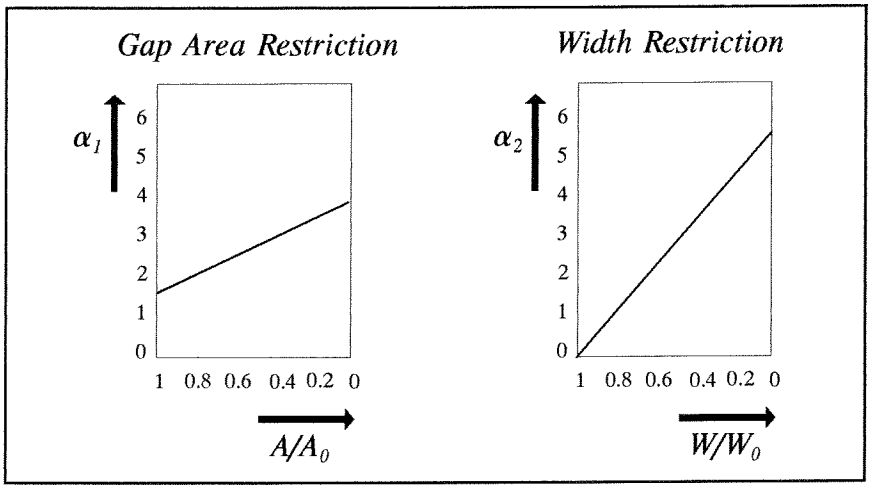


Figure 6.4: Determination of turbulence factor α

In *Appendix D* and in chapter 5.1.5 of *volume II* of this paper is presented how these calculations have been effectuated in the program.

6.3 *Application of Breuser's Formula*

Breuser's relation for scour depths could be written as:

$$h_s = P\, t^{0.4} \quad \text{with} \quad P = f(\alpha u)$$

If in this relation P changes for every time step, the calculation of the total scour depth h_s could be as follows. Each of the intervals behind the bottom protection has certain depth at the beginning of every time step. In the next time step, the scour calculation has to be continued for another process speed P. This means, that at every time step the scour continues with another velocity:

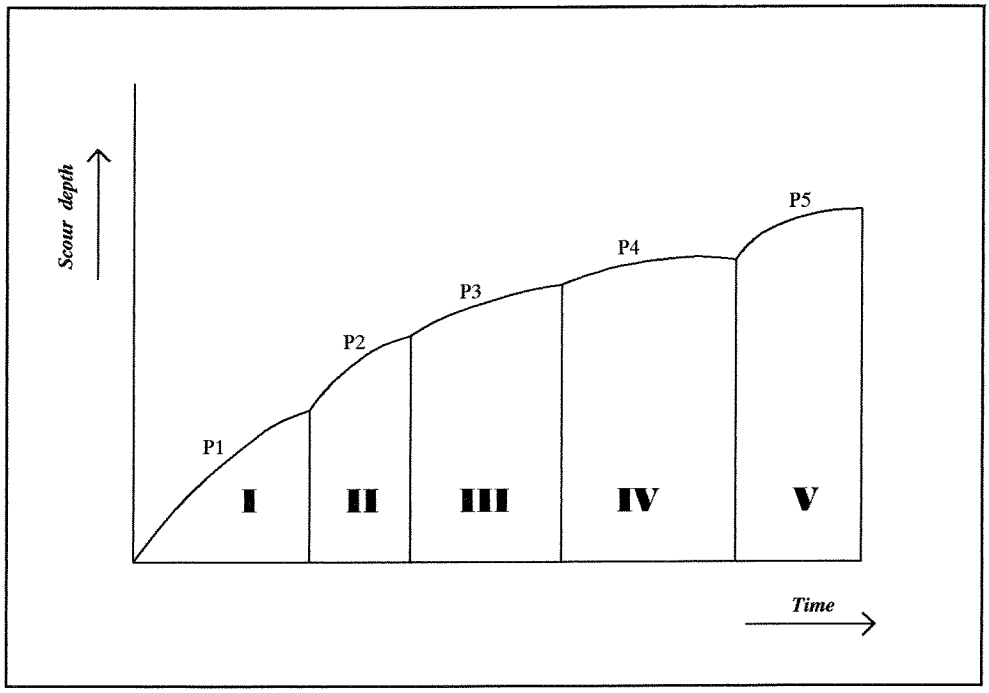


Figure 6.5: Scour depth increase in time

At the beginning of every time step is calculated, how long the actual scour process (with process speed P) should have needed to achieve the same scour depth. The scour process is then continued during the time step ΔT. For one time step, being the second, this concept has been explained in figure 6.6:

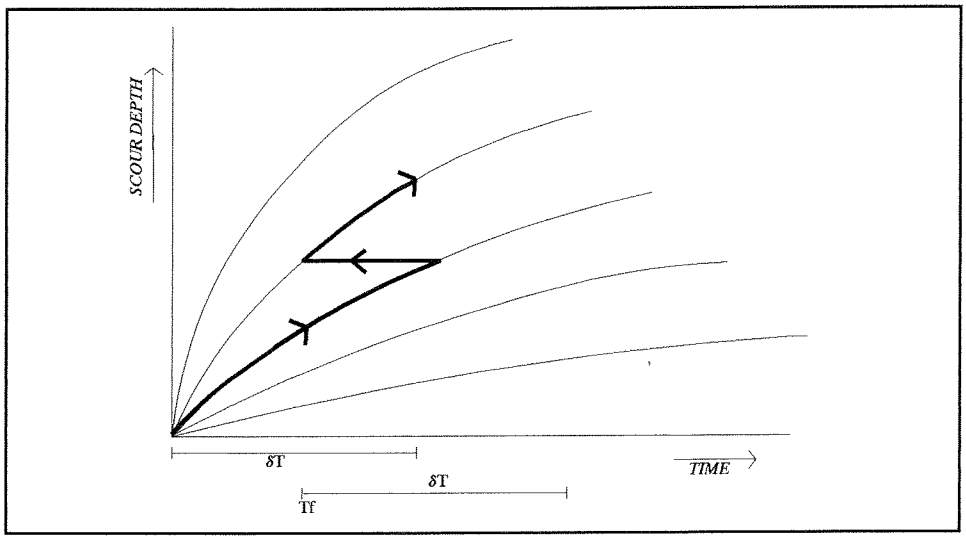


Figure 6.6: Scour hole calculation

6.4 *Calculation of the erosion angle*

The angle ß at which the scour occurs, seems not to be dependent on the duration of the scour process [DHL, 1985]. This can also be seen in figure 6.7:

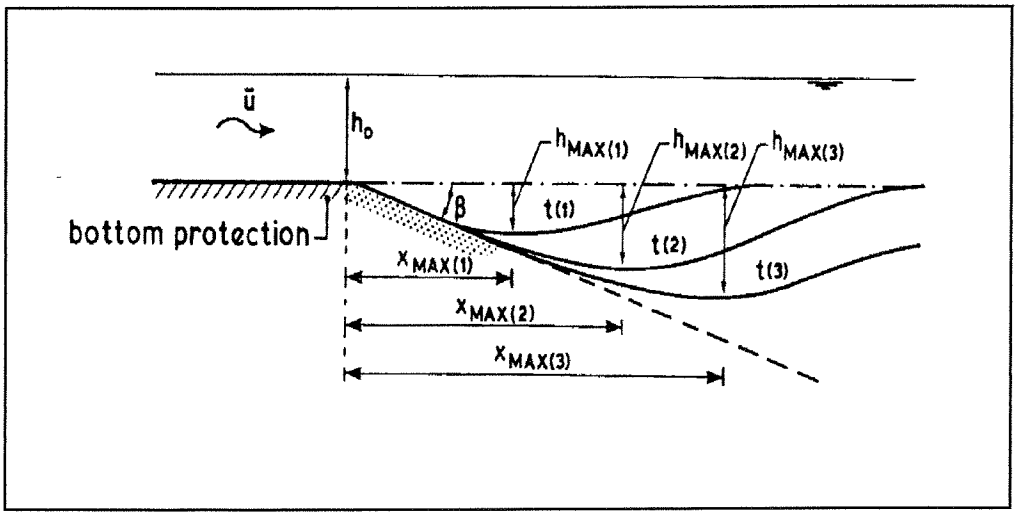


Figure 6.7: Scour hole progress

The scour holes are uniform in time, and only the depth (and location of the greatest depth) varies.

An expression for the angle ß at which the scour holes occur is found to be [Huis in 't Veld et al, 1987]:

$$\beta = \arcsin[3.10^{-4}\frac{u_0^2}{\Delta g d_{50}} + (0.11+0.75r_0)f_c]$$

$$\text{with} \quad f_c = \frac{C}{40} \quad (F_c = 1 \quad \text{for} \quad C \leq 40)$$

During one tidal period, the velocity fluctuates between its extremes. The erosion angle and the depth of the scour holes do not respond immediately to these fluctuations; their value depends more on the average velocities that occur.

Therefore, it seems useful to calculate both values only once per tidal period, using average values. For Breuser's relation and for the expression for the angle of erosion, every time step the program integrates for each flow direction:

$$(\alpha u - u_c) * dt \qquad (u_0^2) * dt$$

Each time when the level difference is zero (slack water), these integrals are divided by the time which passed since the last slack water occurred. The results then are used in the formulas for the calculation of h_s and β. With this, Breusers' formula reads:

$$h_s(t_{hours}) = \frac{\int_0^T (\alpha\bar{u} - u_c)^{1.7} dt \quad h_0^{0.2}}{10\Delta^{0.7}} \; t^{0.4}$$

6.5 *Reduction of scour hole depth*

In reality, also without any form of obstruction in the channel, a continuous process of

erosion and sedimentation takes place. A state of equilibrium exists because the area of the profile develops such, that the erosion and sedimentation on long term average have the same value. At every location during a certain period the erosion is compensated by the sediment supply. Breusers' relation holds for a disturbed profile in which the upstream sediment supply is a limited amount.

During construction, the discharge that passes through the channel is lower, due to the restricted gap area. But according to this flow, there is still a sediment supply, which tends to settle in the scour holes that exist [Vinje, 1975]. Especially in the first stage of construction, when scour already occurs but the total flow has not decreased very much, the scour holes would be calculated as much too deep if this supply would be ignored.

In order to calculate the upstream sediment supply, the velocities that take place in the unrestricted channel have to be calculated. This initial channel size is present at some distance from the construction; the area of this section is the same as the equilibrium profile, which has been calculated from the tidal conditions and basin characteristics (section *3.2.2*). In figure *6.8* this has been made clear. The volume of water which passes through *A1*, which is known, also has to pass through *A2*. The velocities which occur in *A2* lead to a certain transport, which can be implemented in the calculations on the scour depth.

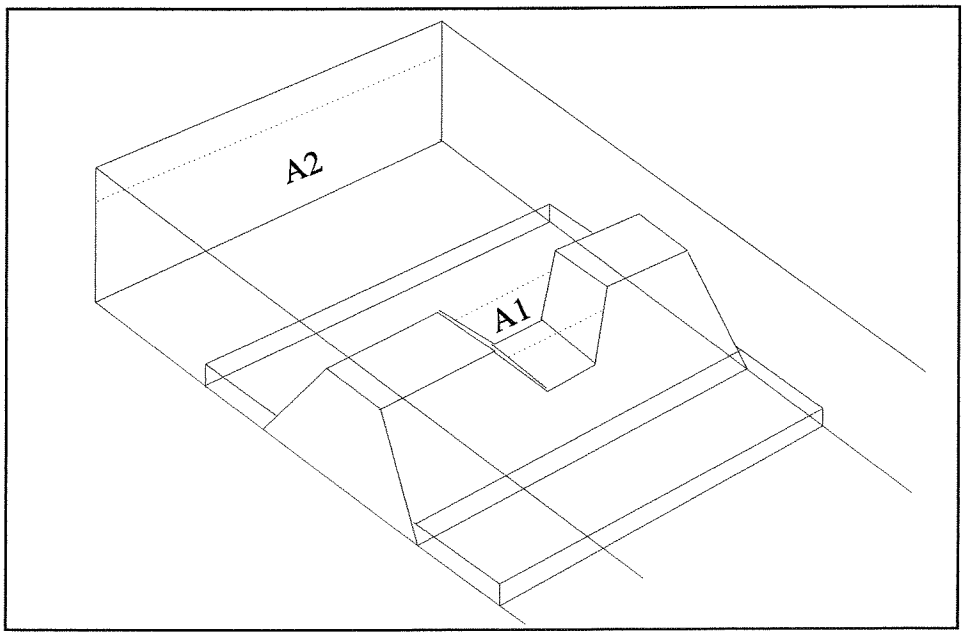


Figure 6.8: Perspective of flow sections

One important issue is, to determine how much of the sediment, supplied from section *A2*, really passes through *A1* and reaches the scour holes at the other side of the construction. The material which does not pass the construction, will settle down in front of it. A simple assumption for the material that passes the construction can be given:

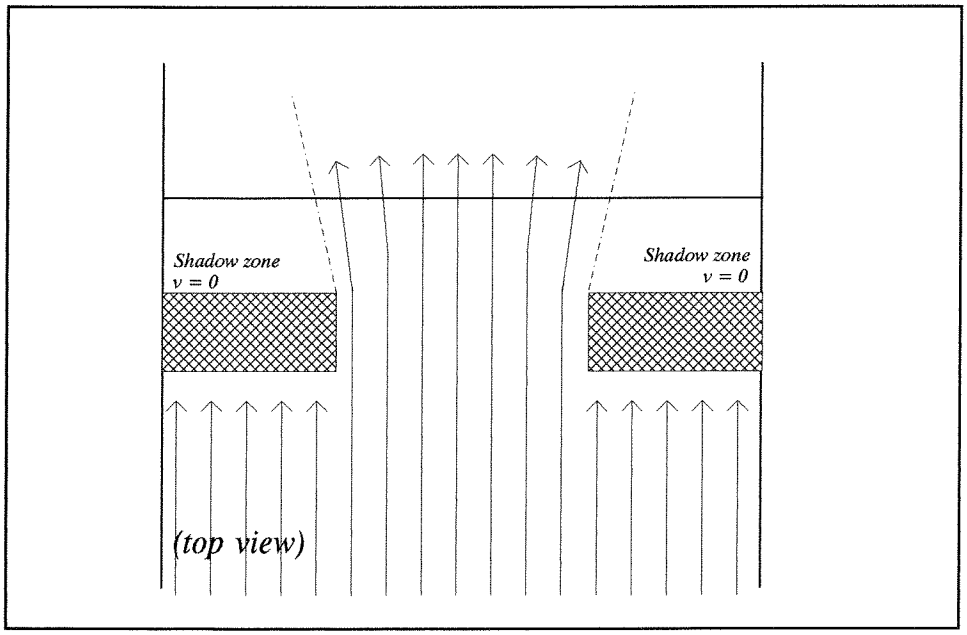


Figure 6.9: Assumption for passing of sediment

In this approach the sediment transport, which is calculated per meter width, is assumed to continue straight through the gap profile: sediment behind the dam is not considered. An idea could be, to have this material accrete in the scour gaps behind the dam.

The sediment per meter width in the channel can be calculated from the velocity, using Engelund & Hansen's formula. This material enters the scour holes behind the bottom protection and settles down, reducing the depths. To be able to calculate the storage of material in the scour hole, the dimensions of it have to be estimated. These are [Vinje, 1975]:

$$I_t = b\, h_{\max}^2(t) \qquad (per\ m\ width)$$

in which b is a shape factor and I_t is the quantity of material discharged until then. If the upstream material supply is T_s per meter width, the reduced quantity of material will be:

$$I_{t,red} = I_t - T_s t$$

because of this reduced scour hole volume, the reduced depth can be expressed as:

$$h_{\max,red}(t) = \sqrt{h_{\max}^2(t) - \frac{T_s t}{b}}$$

And if this is completed with Breuser's expression:

$$h_{s,red}(t_h) = \sqrt{\frac{(\alpha \bar{u} - u_c)^{3.4} h_0^{0.4}}{100\Delta^{1.4}} t^{0.8} - \frac{T_s t}{b}}$$

6.6 Scour Hole Collapse

As explained before, at both ends of the bottom protection various scour hole depths can occur, according to the scour history of each area. At a certain depth, these scour holes will collapse, causing the bottom protection to settle. This settling might cause the protection to loose functionality.

For the calculation of this collapse, various methods are available. To be able to implement these processes in the program, some schematisation is needed. The scour depths at the end of the bottom protection have been calculated per section of the initial width. A possible scour hole contour after a horizontal closure from one side has been indicated in figure *6.10*:

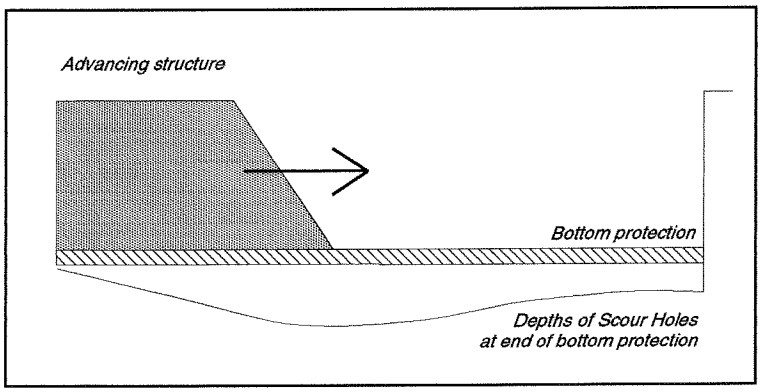


Figure 6.10: Scour hole contour during closure

In this way it could be, that at some location a scour hole with a critical slope has been eroded, while at another location the situation is stable. For the calculation of the collapse of this slope, a section in which the critical situation has been reached is considered, schematized as follows:

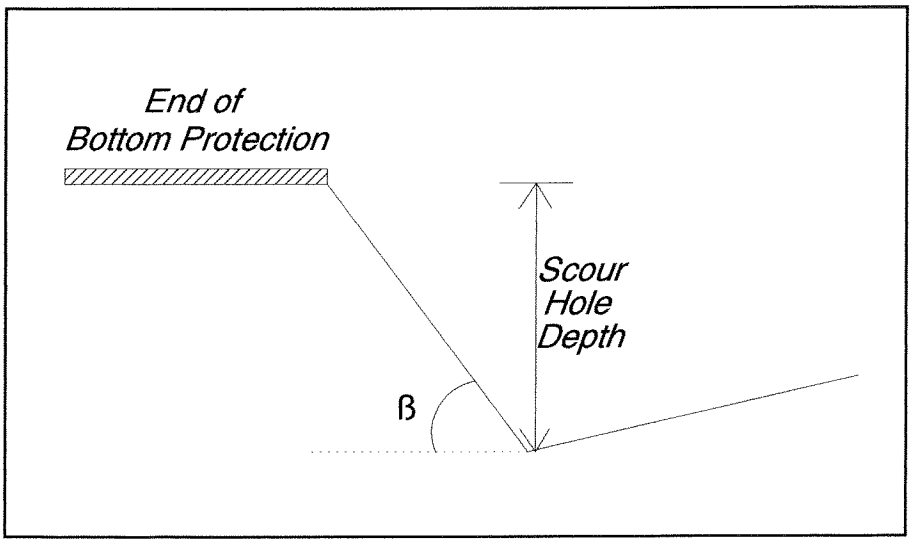


Figure 6.11: Scour hole depth and angle

When the bottom protection is of good quality, after a scour hole has collapsed this protection will settle. In that case it can occur that not the slope and depth of the scour hole itself, but another combination of angle and depth will exceed the critical values. In the program, all combinations of depth and angle have to be checked. A possibly critical situation has been shown in figure *6.12*:

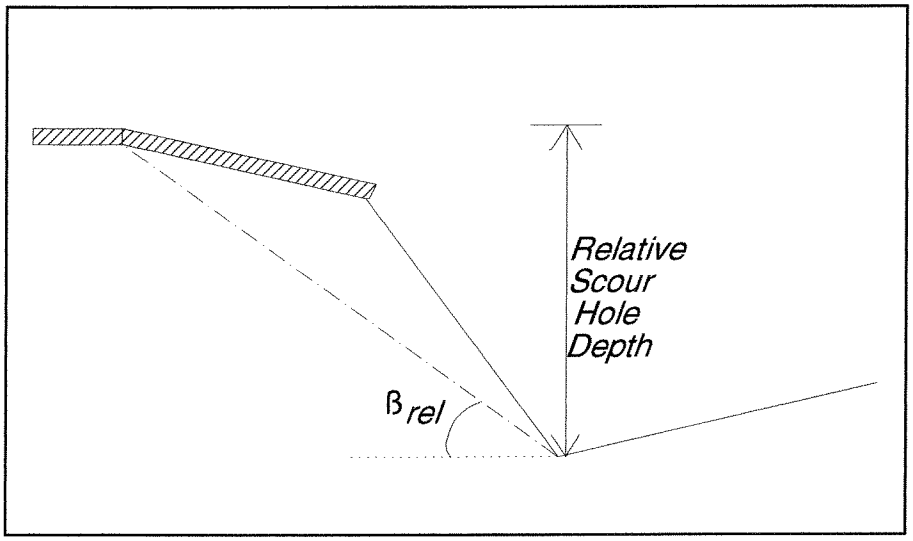
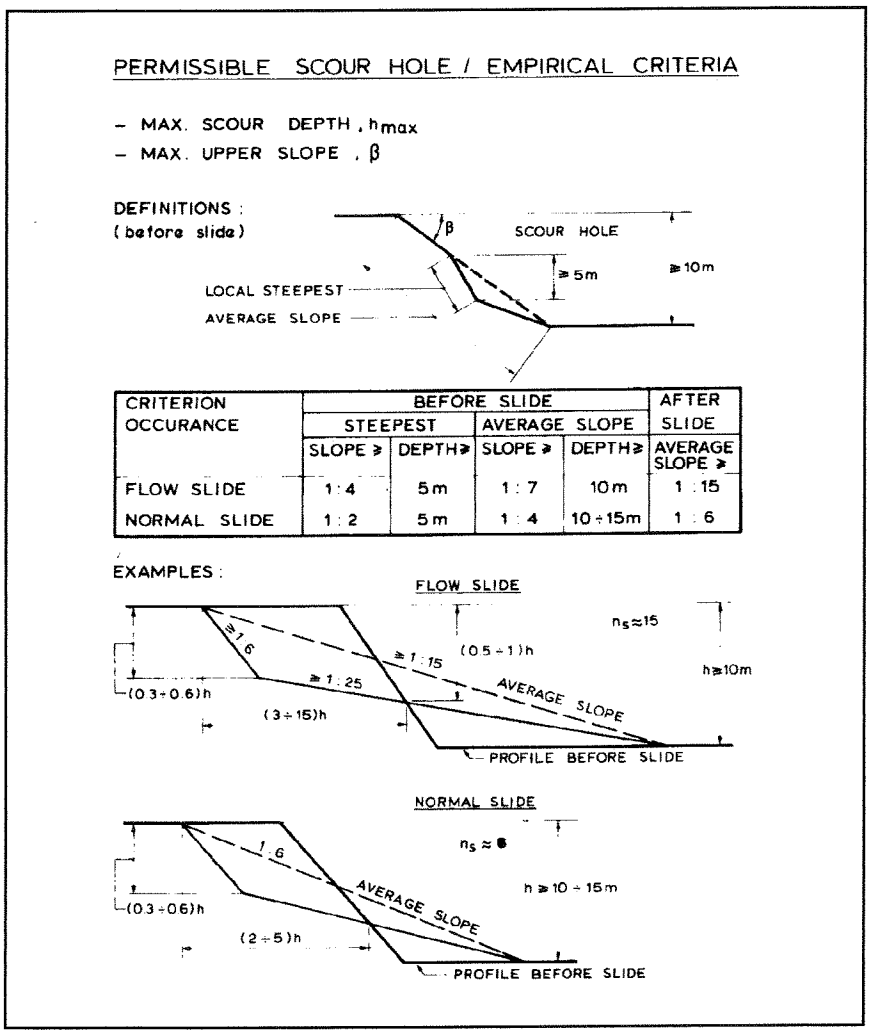


Figure 6.12: Relative scour hole depth and angle

It is important, that the program can determine the condition for instability. Using the *Bishop method*, a large number of sections and possible centre points of calculation should be calculated for every time step, each consisting of various layers of material. This would take much calculation time and would make the program run much slower. But more important: the Bishop method will not specify the new equilibrium slope, it only determines whether a slope is in a stable condition *(F>1)* or not *(F<1)*.

The mechanism of flow slides is not yet fully known. Therefore, it will be difficult to predict the damage and rate of backward erosion when a slide occurs. The combination of maximum depth and upper slope for which sliding occurs, depends largely on the soil mechanics of the bottom. From empirical data, scour hole criteria have been deduced which consider this combination of maximum scour hole depth and maximum upper slope. As soon as the hole is too deep *and* the slope to steep, the hole collapses, and a new, prescribed slope and depth results. The schematisation in this method is quite simplistic, but its calculation is fast and the main goal is reached: scour holes which are too deep or have upper slopes that are too steep, collapse and take on a new state of equilibrium. The principle has been explained in the figure on the next page [Davis, 1978]. In order to calculate every possible slide for each of the width sections, the bottom protection is divided into *20 x 20* grid areas. For these areas, the stability for sliding is checked.

As soon as the collapses of the scour holes approach the dam toe, the weight of the structure itself will play a role in the sliding stability calculations. In this version of the program, this has not been considered. In the program, in this case failure will occur because the structure is assumed to fail, when the collapsing of scour holes causes the dam toe to settle.



CRITERION OCCURANCE	BEFORE SLIDE				AFTER SLIDE
	STEEPEST		AVERAGE SLOPE		
	SLOPE ≥	DEPTH ≥	SLOPE ≥	DEPTH ≥	AVERAGE SLOPE ≥
FLOW SLIDE	1 : 4	5 m	1 : 7	10 m	1 : 15
NORMAL SLIDE	1 : 2	5 m	1 : 4	10 ÷ 15m	1 : 6

Figure 6.13: Slopes for collapsing scour holes

In the program, the *flow slide* mechanism has been included, in which the critical steepness and heigth, as well as the situation that will exist after collapse can be adapted in the program source code. In the top view of the construction, the actual state of the bottom protection will for each location be indicated with a different color. For the locations where the scour hole has collapsed and settlement has taken place, this is illustrated in figure *6.14*:

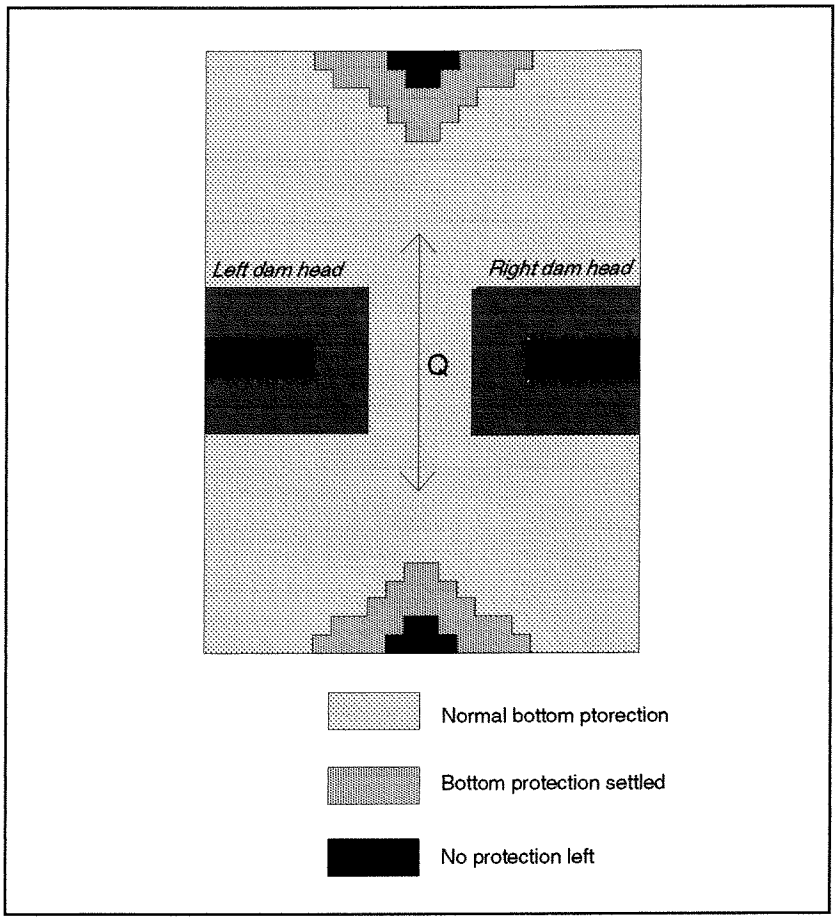


Figure 6.14: Damage of the bottom protection due to collapsing scour hole

In this top view of the structure, a horizontal closure is shown. Because of the turbulence in the vortices which are caused by the tidal currents, at the end of the bottom protection scour holes have developed. When these scour holes collapse, the bottom protection can settle (indicated by the gray area in the figure) or even loose its functionality (black in the figure).

7. Recommendations for Future Extensions

Future versions of the simulation program

With respect to a lot of aspects, the simulation program *CLOSIM* can be extended in future versions. Some features and calculations of the program can be reconsidered in more detail. In this chapter, several recommendations for future versions of the program have been done.

1. Consideration of Failure Mechanisms

In reality, the dam structure can fail according to many more mechanisms than actually have been dealt with in the simulation program. In figure *7.1*, the main failure mechanisms for the construction have been illustrated [DHL, 1985]:

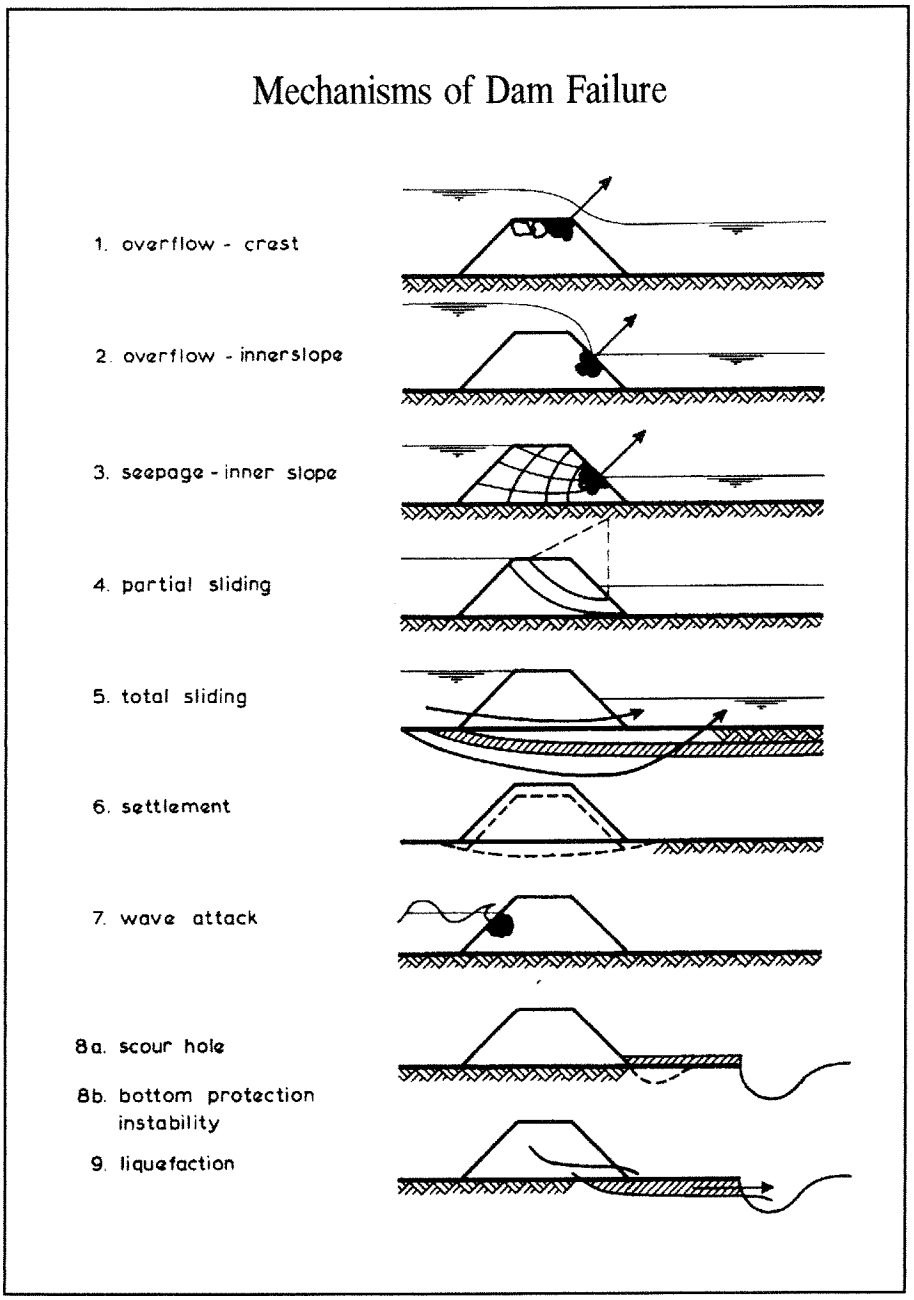


Figure 7.1: Failure mechanisms for closure dams

Most of these failure mechanisms have not been included in version 1.0 of the program. In future versions they should be paid attention to, especially because of the purpose of the program to give insight in the closure practice. In particular the mechanisms which have to do with failure of the subsoil should be subject of interest.

2. *Program Adapted for Sudden Closures*

In the actual version of the program, only gradual closures are possible. Because the sudden closure with elements like *caissons* is another option to close a tidal inlet, the program should be adapted in order to make sudden closures possible.

Together with the user module for sudden closure activities, also new calculation procedures should be added. For closures with larger elements, other failure mechanisms can be normative. Examples are *piping*, which is the flowing of water below and behind the structure, and the *sill stability*.

3. *Schematisation of Gap Dimensions*

In the actual version of the simulation program, the closure gap has a rectangular schematisation, in which the depth and the width of the gap are parameters. In reality, the depth contour of the gap will not be uniform. In that case the conditions for the closure are different. The activities will be more like, for example, shown in figure *7.2* [DHL, 1985], in which a cross section of the southern gap of the 'Brouwershavense Gat' closure in the Netherlands is given. It is clear that not in all cases a rectangular schematisation of the gap is realistic:

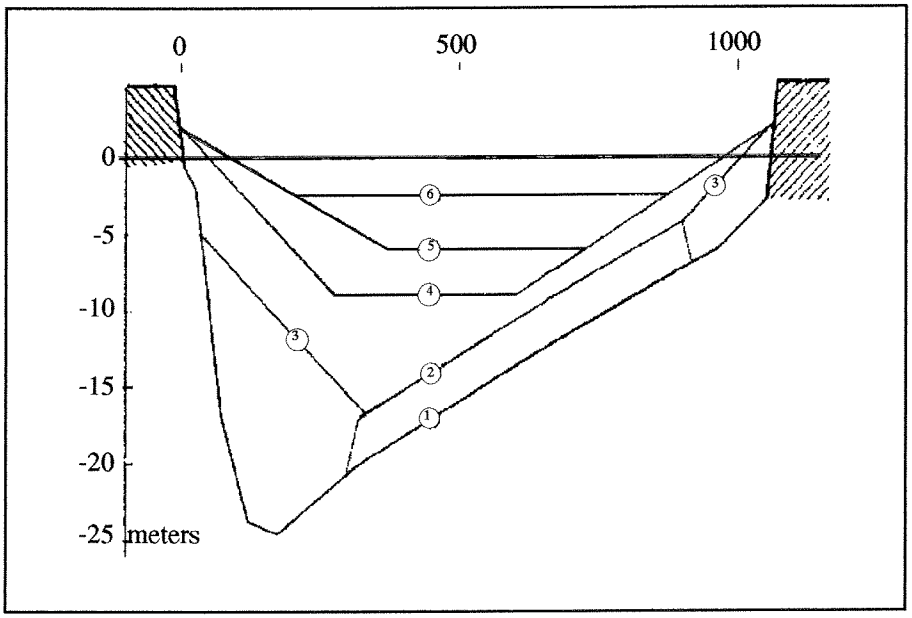


Figure 7.2: Phasing of an actual closure

More attention could be paid to schematisations of the gap profile in future versions of the program, in order to match better with the reality of the closure practice.

4. *Effects of Contraction on Erosion of Dam Material*

The head contraction effects which have been regarded for the calculation of scour holes at the end of the bed protection, also have a strong effect on the erosion of dam material. As for now (see chapter 5), the erosion has been assumed to be equal at all sides of the flow area. In reality at some location the flow gradients will be much higher, due to contraction of the flow.

This flow contraction has been taken into account in the calculations of the scour holes (see section *6.2*), but has also an influence on the erosion of the dam material which should not be ignored. When this influence is included in the calculations though, also the actual geometrical dam schematisation has to be reviewed, because the sill would then not always be horizontal. To apply the contraction effect on the head slopes, should be no problem. In figure *7.3*, the locations where due to head contraction locally a higher velocity occurs, have been indicated by the arrows.

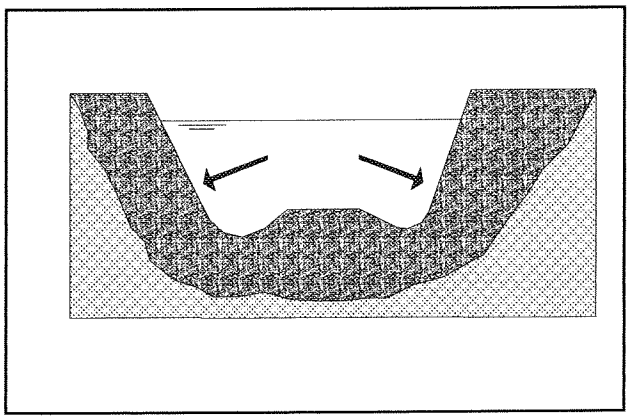


Figure 7.3: Effects of contraction

5. *Dam Porosity*

As stated in section *5.3*, the dam porosity is very important for the gap flow and the crest stability. In version 1.0 of *CLOSIM*, the structure porosity has not been included as of yet. It is an important parameter which should be regarded in future versions of the program.

6. *Stability of Bottom Protection*

In this version of the program, the bottom protection has been schematised in a very simplistic way: it is defined only by its length and its resistance to settlements. In reality, for a feasable design of such a protection many parameters have to be taken into consideration. Watertightness, resistance to flow, costs and roughness are examples of such parameters.

In the future extensions of the program, a *bottom protection menu procedure* should be programmed in which the design of a bottom protection can be done. To the calculation procedure of the program, a *bottom protection check* should be added in which the stability (with respect to filter properties, the water pressures and the flow resistance) is checked.

7. *Filter Requirements for Structure*

At this stage, the program allows the user to drop large materials on a sandy bottom, or to use a great variety of diameters within the structure. In reality, such a structure might collapse because the *filter requirements* have not been fulfilled. In extensions for the program, a check should be added which verifies this filter function. In a first adaption, the user could be warned whenever he would choose incorrect diameter combinations. Later, procedures could be added to check the filter function, which would cause a wrongly designed dam to collapse.

8. *Wave action during and after construction*

Waves which act on the structure, could cause material losses during construction, and damage after the structure has been finished. In later versions of the program, this wave action should be considered. In version 1, the wave action has been taken into account partially, with regard to the velocity increase in the closure gap during a storm (see section *3.1.5*)

9. *Limitations of Application of Storage Model*

In section *3.2* the storage model has been chosen to calculate the flow conditions in the closure gap. The storage model though, has a number of limitations, which restrict the number of cases that can be considered. If longer or more complex basins or partial closures have to be considered, a one-dimensional model with a limited number of nodes

could be applied, without decreasing too much the computational speed of the program. The exchange of the flow calculation routines for other ones is quite simple, and has been explained in the second volume of the manual.

10. General Possibility of Closure Solutions

In coming versions, the user's possibility for closure could be restricted. Availabilities of materials and equipments, logistical problems, unexpected weather conditions, and other aspects could add a realistic element to the simulations. In case of caisson closure, the required preciseness for the placing of the elements might not be reached; an unforeseen delay in supply of materials could affect the closure works. There are a lot of possibilities for extension of the program in this area.

11. Cost Aspects of Closure Works

In version *1.0* of the program, any gap can be closed by applying a long bed protection of good quality, and dumping big stones. It would be nice if the user would see himself restricted by the costs of material and equipment, which would stimulate him to build an economical dam structure. If the program is extended in this way, the routines needed for the calculation of the total dam costs could be written easily. A problem would be the determination and application of prices, because these are variable in time and geographical location. It could be an idea to have the prices for material and equipment entered by the program user.

Appendix A

Numerical Scheme for the Flow Calculations

For the flow calculations using the storage model, an explicit numerical scheme has been provided by taking into account one of the future conditions, the sea level in the next time step, which is known from the tidal function. Because of this, there are two equations and two unknown variables (Q and H), which can be calculated explicitly.

In the calculations, the conditions for sea side and basin side have been distinguished by indices *a* and *b*; the time steps have been indicated by *1* and *2*. The calculations take place according to the calculation scheme below, in which Q is positive from left to right:

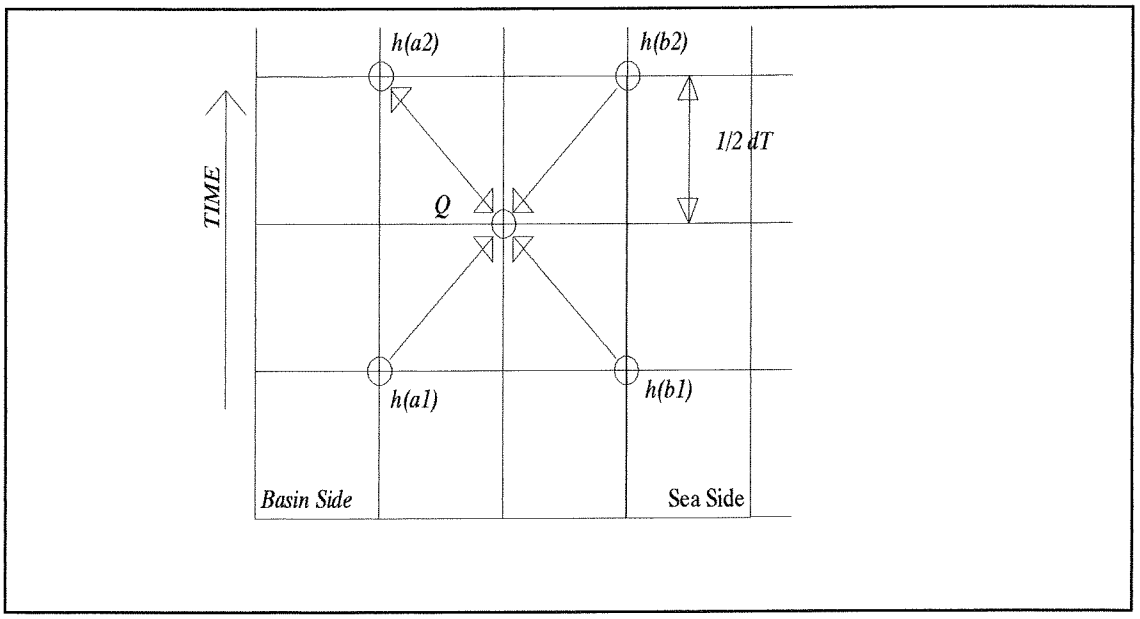


Figure A1: Numerical scheme for flow calculations

The formulas which have been used for this level-flow relation, are:

$$Q|Q| = (\mu A)^2 \left(2g\frac{(h_{b1}-h_{a1})+(h_{b2}-h_{a2})}{2}\right) \qquad \textit{(weir equation)}$$

$$h_{a2} = h_{a1} + \Delta t \frac{Q}{F} \qquad \textit{(continuity equation)}$$

In case of one closure gap, the flows mentioned in both formulas are the same. Because of that, the equations can be joined, by substituting h_{a2} from the continuity equation into the weir formula. Then, both formulas can be written as one quadratic equation:

$$\frac{Q|Q|}{\mu^2 A^2} + g\frac{\Delta t}{F}Q - g(h_{b1}+h_{b2}-2h_{a1}) = 0$$

When this is written in the standard form for quadratic equations, it yields:

$$C_1 Q^2 + C_2 Q + C_3 = 0$$

in which the real coefficients are:

$$C_1 = \pm (\mu A)^{-2} \qquad C_2 = g\frac{\Delta T}{F} \qquad C_3 = -g(h_{b1}+h_{b2}-2h_{a1})$$

in which the sign of C_1 will be equal to the sign of Q. An interesting note is, that the sign of C_3 will always be different from the sign of Q . In the final solution, this knowledge can be used: the sign of C_1 - which is the sign of Q - can be obtained from the sign of C_3, which is opposite to the sign of the level difference.

If the general solution of the quadratic equation is considered:

$$Q = \frac{-C_2 \pm \sqrt{C_2^2 - 4C_1C_3}}{2C_1}$$

It can be seen, that in this solution the product C_1C_3 occurs, of which is known that it is always negative. The determinant, given by

$$C_2^2 - 4C_1C_3$$

will be *always* positive, so there are no complex solutions of this equation.

Now there are still two real possible solutions: one in which the root has a positive sign and one in which it has a negative sign. Considering these possibilities though, it remains clear that this sign has to be positive. If not, a positive Q in the left side of the equation would contradict the right side of the equation, which always would be negative.

The final solution, which is always real, reads:

$$Q = -(SIGN\ \ OF\ \ C_3)\ \frac{-C_2 + \sqrt{C_2^2 - 4C_1C_3}}{2|C_1|}$$

When entering the results into the continuity equation, this yields the level difference for the next time step.

More gaps to close

In the case there are more gaps to close, the situation gets different. The flows in the weir formula and the continuity equation are not the same then. The resulting equations read:

$$\frac{Q_1|Q_1|}{\mu^2 A^2} + g\frac{\Delta t}{F}(Q_1+Q_2+Q_3) - g(h_{b1}+h_{b2}-2h_{a1}) = 0$$

$$\frac{Q_2|Q_2|}{\mu^2 A^2} + g\frac{\Delta t}{F}(Q_1+Q_2+Q_3) - g(h_{b1}+h_{b2}-2h_{a1}) = 0$$

$$\frac{Q_3|Q_3|}{\mu^2 A^2} + g\frac{\Delta t}{F}(Q_1+Q_2+Q_3) - g(h_{b1}+h_{b2}-2h_{a1}) = 0$$

The values for Q have to be estimated for each calculation. Thereto, some iterations can be needed to get a proper result. Again, each of these equations can be solved using the ABC formula, but in this case coefficient *C3* is different, for the three gaps:

$$C_3 = -g(h_{b1}+h_{b2}-2h_{a1}) + g\frac{\Delta T}{F}(Q_2 + Q_3)$$

$$C_3 = -g(h_{b1}+h_{b2}-2h_{a1}) + g\frac{\Delta T}{F}(Q_1 + Q_3)$$

$$C_3 = -g(h_{b1}+h_{b2}-2h_{a1}) + g\frac{\Delta T}{F}(Q_1 + Q_2)$$

In the other coefficients *C1* and *C2*, the flow is now the gap flow. The solution method for all gaps remains the same in this case.

N.B. In all equations no attention has been paid to the fact that the flow area A and the storage area F also depend on the water level. Neglecting this is fair though, because the time steps are small and per time step A and F hardly change.

Appendix B

Calculation of Dam Advancement

For horizontal construction of the dam, various types of layers can result in different dam advancement calculations. The possible layers have been indicated in paragraph *4.1*. As mentioned, for a dumped volume of material has to be calculated which advancement for the dam it causes. In this appendix these calculations have been done.

(I) Wished top level of dam not reached yet

The added volume, which is known, is expressed in *dX*, which is unknown. A general form for this expression is:

$$\frac{V}{B} = A_{tri} dX^2 + A_{rect} dX$$

From this equation *dX* is calculated; hence, the advancement of the dam in vertical and in horizontal direction is known. In this stage, in which the required top level has not yet been reached, the calculation can be done as shown in the figure *B.1*:

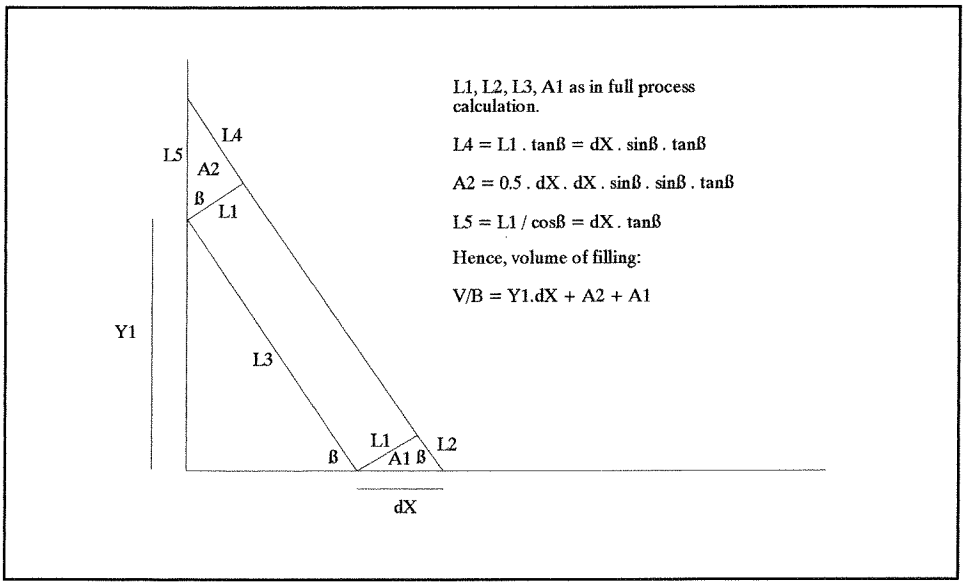


Figure B.1: Calculations for starting process

(II) Full process; wished top level of dam has been reached

As soon as the required top level of the dam has been reached, the equation gets much simpler. In this case, the triangles which can be expressed in the square of dX, are equal, because of which the second power disappears. The area *A1* has been calculated to be applied in the other calculations; in full process, the calculation is quite simple:

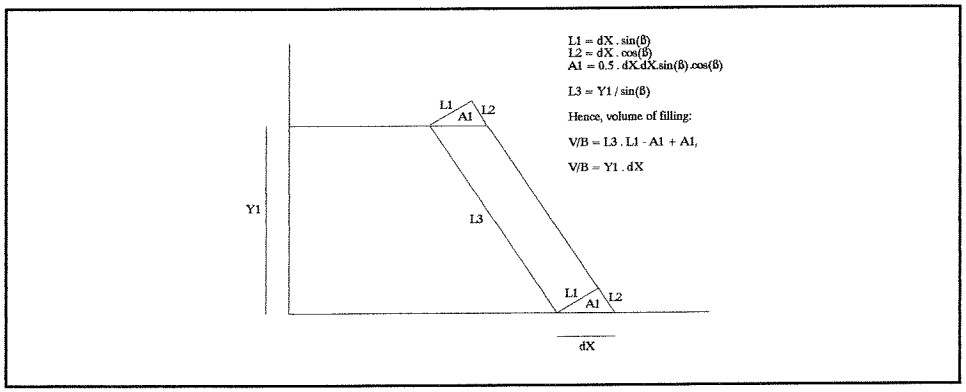


Figure B.2: Calculations for full process

(III) Other side of gap is reached

As soon as the other side of the gap has been reached, the dam has stopped expanding horizontally:

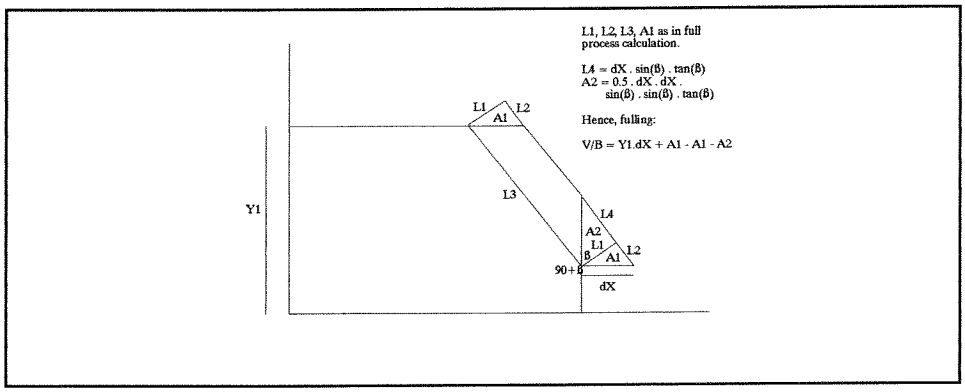
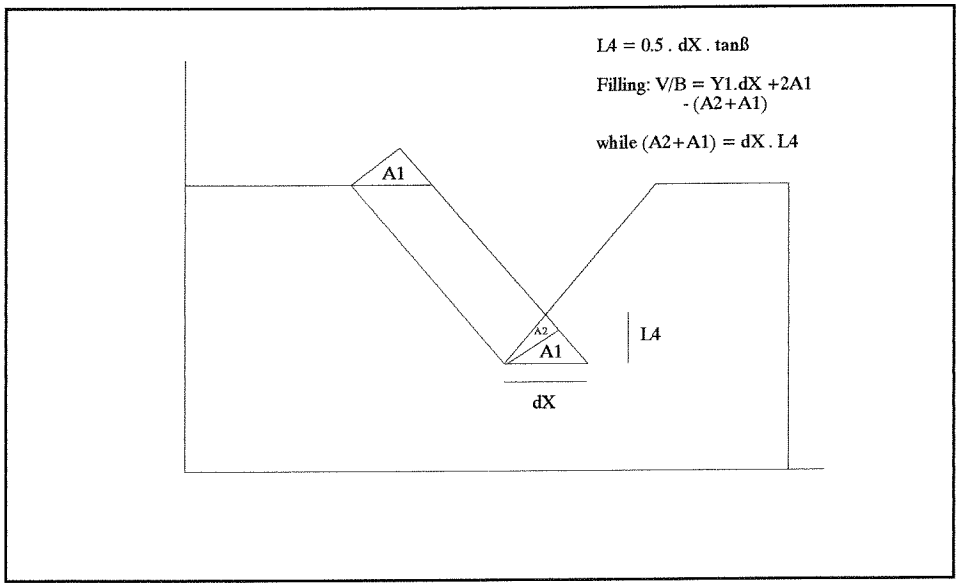


Figure B.3: Calculations when other side is reached

(IV) Toes of horizontal fillings touch each other

The last possible case consists of two sided horizontal closure, when the toes of the fillings in both directions touch each other. The advancement can be expressed as follows:



Figur B.4: Calculations when toes touch

The calculation of these phases, provides for the coefficients of the advancement equation, from which at any entry of material from the dam head, the advancement of the construction can be calculated.

Vertical construction from below water level

In the case that the toes of the horizontal fillings touch, the toe distance is zero; especially in this case, considering these edges is important. In the first of the cases (combined filling with horizontal construction from both sides, which makes that all prismatic edges should be calculated), the calculation becomes:

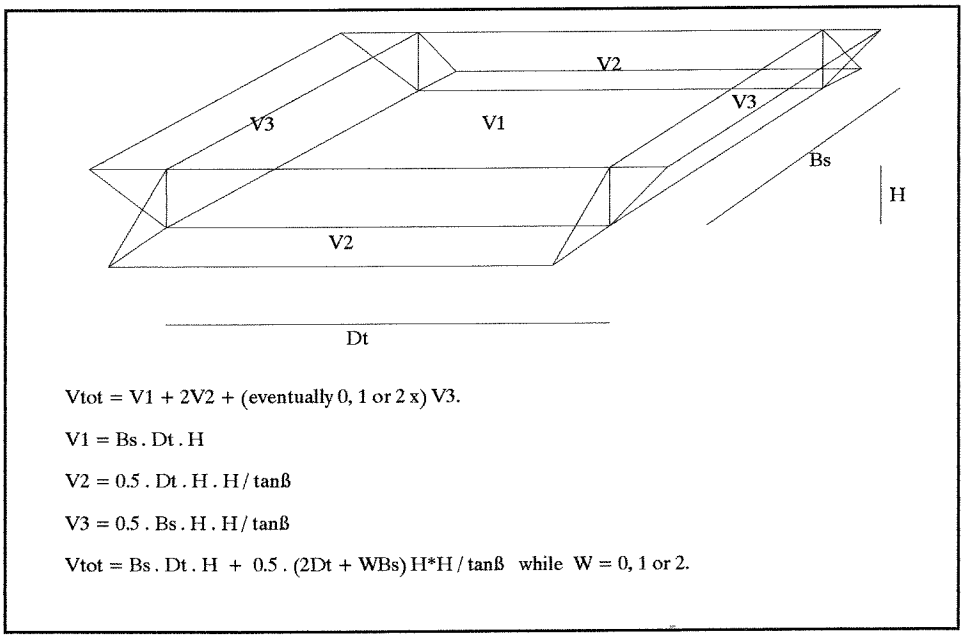


Figure B.5: Calculation of the vertical advancement

By which, for each of the cases, the filling can be expressed in an increase of the structure height.

Vertical Construction from Above Water Level

For construction by a cable way, the following calculation has been used in the program to calculate the dam advances:

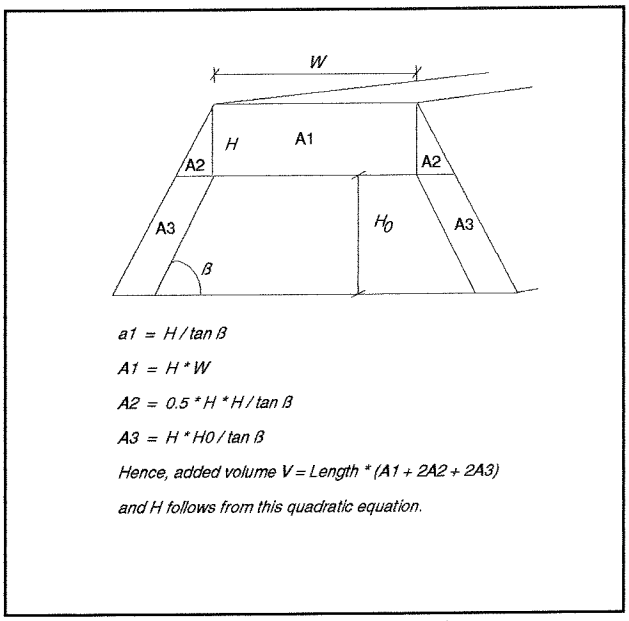


Figure B.6: Calculations for cable way dumping

Repercussions of a Material Change when Closing Horizontally

As described in chapter *4.6*, when the material size is changed and the closure is horizontal, the volume of the additional slice should be calculated and be corrected for the total dam advancement. Hereto, the slice is considered:

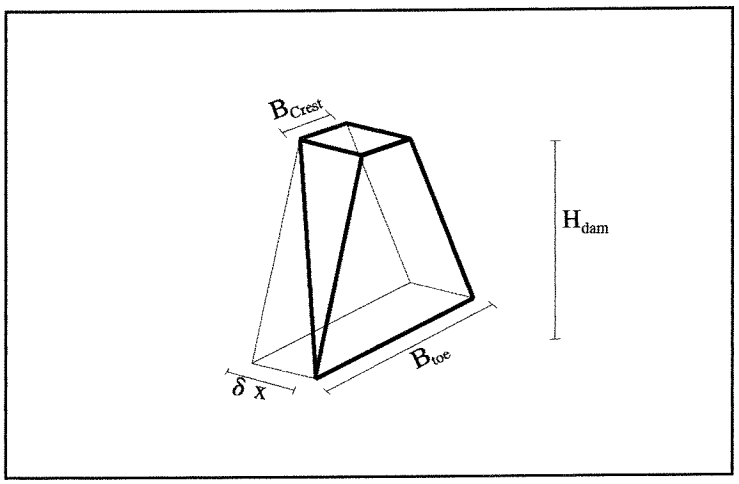


Figure B.7: Required volume correction

The volume of this slice can be calculated by considering the dam advancement two-dimensionally, and after the calculation of the section areas, multiplying these by the verage structure width. To calculate the section areas, the following figure is made:

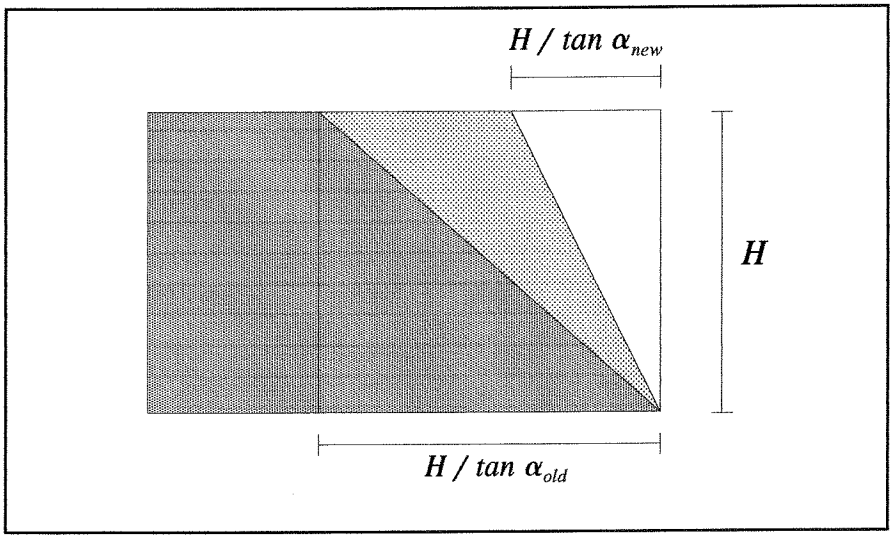


Figure B.8: Calculation of the volume correction

The calculation results in the following formula for the calculation of the volume correction:

$$\Delta V = \left(\frac{H^2}{2 \tan \alpha_{old}} - \frac{H^2}{\tan \alpha_{new}}\right) * \left(B_c + \frac{H}{\tan \alpha_{new}}\right)$$

Appendix C

Transport Formulas of Paintal and Engelund & Hansen

In chapter *5.2* of this volume has been described that the transport will be described by the transport relation of *Engelund & Hansen* [Engelund/Hansen, 1967], while at low shear stresses the empiric formula of *Paintal* [Paintal, 1970] will be used. *Paintal* found, like *Engelund & Hansen*, that normal transports depend on the fifth power of the velocity. Below a certain shear stress though, the transport seemed to depend on a much higher power (32th) of the velocity.

For the simulation program, a combination of this 32th power relationship with Engelund & Hansen's transport formula has been proposed. This was done, in order to avoid a strict limit as a treshold of movement, which would cause a sudden increase of transports at a certain velocity.

In this appendix the combination of both formulas is shown for various *H/d*, which is the most important parameter. *C* has been calculated with the mentioned formula (see section *5.2.1*)

For $H/d = 10$ (which causes $C=37.4\ \sqrt{m/s}$) the results are:

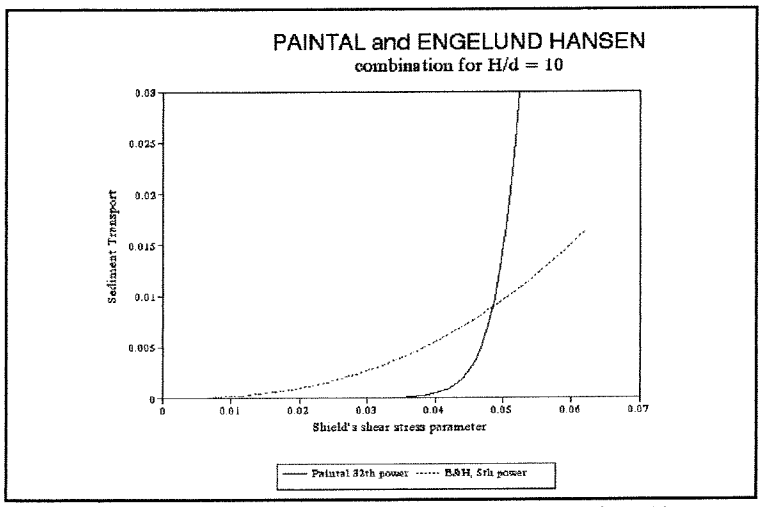


Figure C.1: Combination of Paintal and E&H for H/d = 10

For $H/d = 100$ (which causes $C=55.4\ \sqrt{m/s}$) the picture is similar:

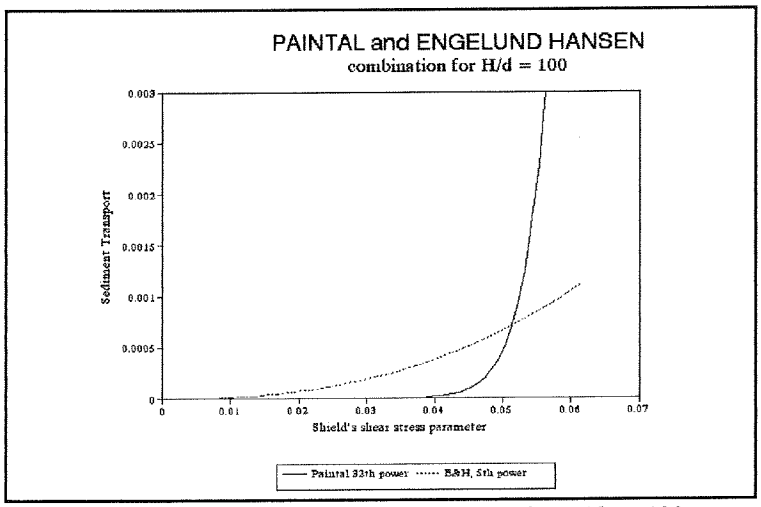


Figure C.2: Combination of Paintal and E&H for H/d = 100

For $H/d = 1000$ (which causes $C=73.4\ \sqrt{m/s}$):

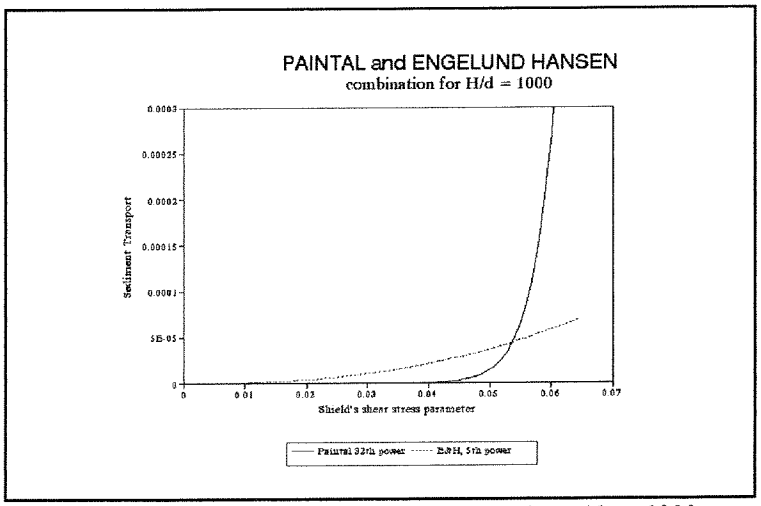


Figure C.3: Combination of Paintal and E&H for H/d = 1000

For $H/d = 10000$ (which causes $C=91.4\ \sqrt{m/s}$):

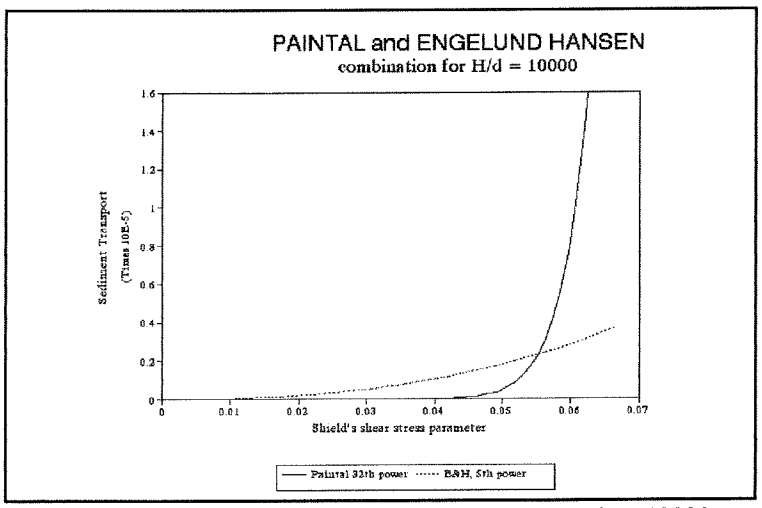


Figure C.4: Combination of Paintal and E&H for H/d = 10000

If the transports in the simulation program are calculated according to the minimum value of both branches, a gradient change will occur where both curves cross. In the graphs it can be seen that this happens at a shear stress parameter [Shields, 1936] around $\psi = 0.05$. This is about the same value as Paintal found himself as a crossing of the 5th and 32th power relations. If this is compared to Shield's conclusion about a threshold of movement at $\psi = 0.055$, the location of this gradient change shows a reasonable similarity.

Appendix D

Verification of the Calculations

For the most important calculations in the simulation program, procedures have been written to verify whether these calculations are correct and precise. In this appendix, the verification procedures and the results of some tests have been discussed.

Verifications have been executed from within the program source. This is done, by exchanging all program loops (menu and calculations) by one partial loop, in which only one specific calculation procedure is executed. For example, the whole simulation procedure can be exchanged for a procedure which only tests the routines which calculate the flow calculations. The results of this execution, which are written to disk, can be considered and verified afterwards.

1. Verification of the Transport Formulas

In order to verify the transport formulas, a test procedure has been written and the simulation loop has been replaced by this procedure. In the test routine, some required values are entered as constants. Then, for a range of velocities, the transport routine *DamDestruction* is called. The results of the calculations are written to disk.

The procedure is as follows:

```
Procedure TestDamDestruction;                                    {Procedure Name}

var CC: Integer;
{Counter}
    OutP: Text;                                                  {Output Channel}
begin
   Assign(OutP, 'TEST1000');          {Assign output channel to file 'TEST1000'}
   ReWrite(OutP);                                                {Reset File Pointer}
   NumOfGaps:=1;                                                    {Test one gap}
   DamD50[1]:=1000;                              {Test for dam material of 1000 mm}
   BottomD50:=DamD50[1];
   Chezy:=50;                                     {Test for Chezy value of 50 √m/s}

   for CC:=1 to 2500 do                                {Do for range of velocities}
   begin
      Velocity[1]:=CC/300;                                      {Calculate velocities}
      DamDestruction;                                           {Calculate Transports}
      WriteLn(OutP,CC, Velocity[1], ErosionPerMPerS[1]);     {Write results to disk}
   end;
   halt;                                                  {After test, stop program}
end;
```

The results of the calculations are written to disk and have been imported into a spread sheet. The graphs of these results have been printed below, for stone sizes of 10, 300 and 1000 mm. For small grains of 10 mm, the velocity/transport relation is:

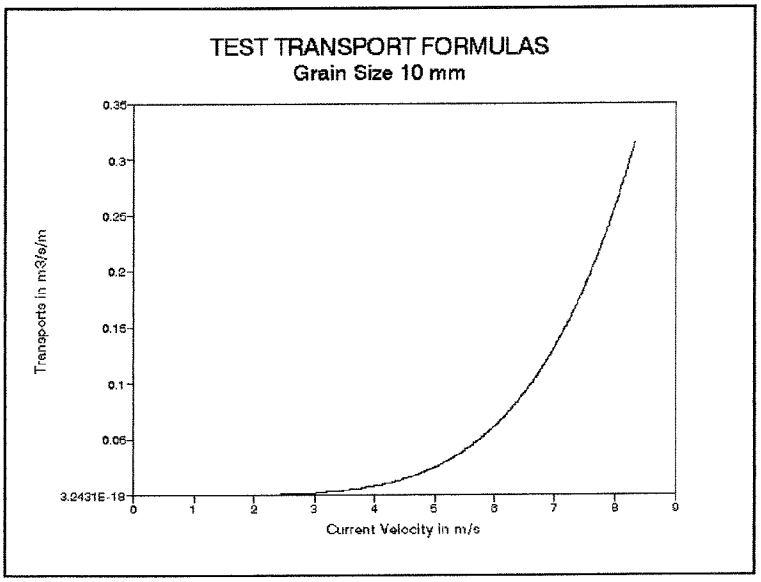


Figure D.1: Test transport formulas for D50=10 mm

This is obvious, because the transport of those small grains will not be restricted clearly by a threshold of movement as indicated by the lower transports of Paintal; as shown in the graph, the transport will take place according to the formula of Engelund Hansen.

With grains of 300 mm, this picture gets different:

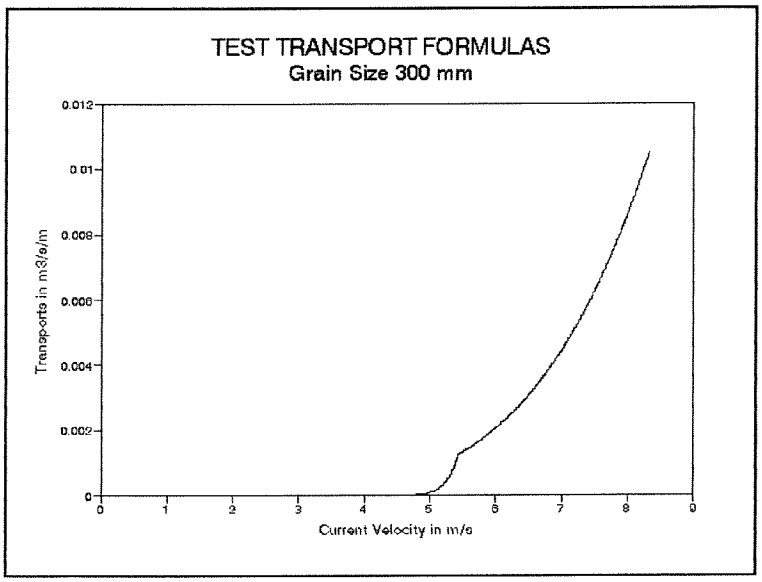


Figure D.2: Test transport formulas for D50=300 mm

It can be seen, that for the lower velocity ranges the program calculates the transports according to Paintal's formula, and that at a certain stage the velocity is high enough for the transport to be according to the fifth power relation of Engelund Hansen.

Finally, for the bigger stones of 1000 mm, the program calculates:

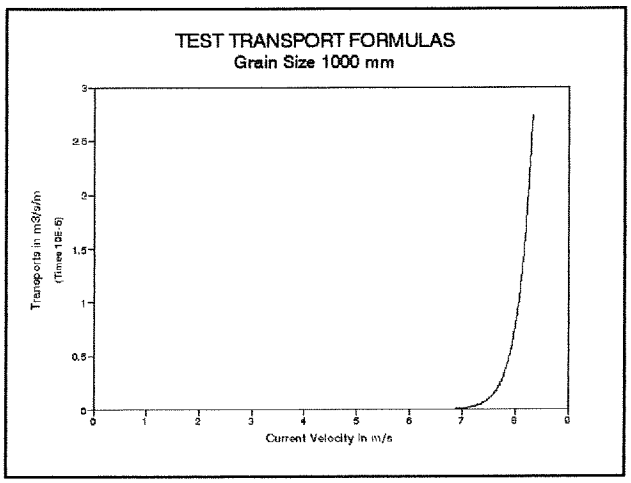


Figure D.3: Test transport formulas for D50=1000 mm

As expected, the program calculates nearly no transport for the usual range of the velocities. At a certain stage though, the 32th power relation seems to give fast increasing

transports. It should be noted though, that the scale of the graph is very small, and the velocity at which considerable transport occurs is at more than 10 m/s. According to Paintal's relation though, these transports might even be correct; anyway, in the simulation program these velocities will not occur.

From this verification can be concluded that the combination of the relations of Paintal and Engelund Hansen has been applied correctly, and that the calculated values can be useful for the transport calculations of the simulation program.

2. Verification of the Scour Hole Formula

The verification of the scour behind the bottom protection should exist of:

a. *Verification of the velocity and turbulence profile* at the end of the bottom protection;

b. *Verification of the integration of the velocity, and the calculation of Breuser's formula at slack water;*

c. *Verification of the reduction by upstream sediment supply.*

ad a.

As shown in *paragraph 6.2*, in Breuser's formula the end of the bottom protection has been divided into various sections, including a *shadow zone* (αu assumed to be zero), a *flow zone (velocity = (flow through gap / flow area at end of bottom protection), while the turbulence depends on the diminuishing of the gap profile)*. Where these zones touch, ther is a *zone of local turbulence*, in which the turbulence α is higher and the velocity u linearly increasing:

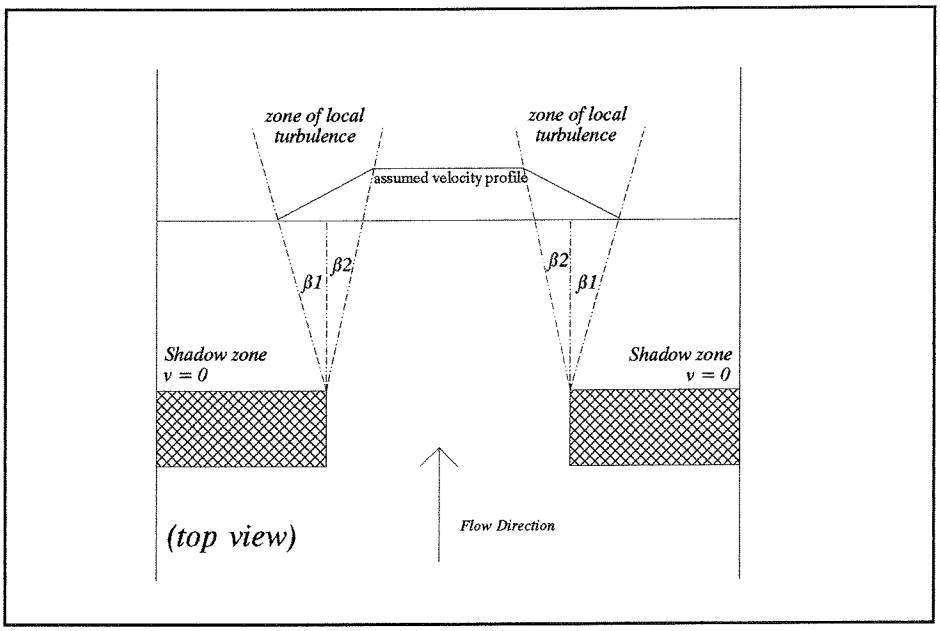
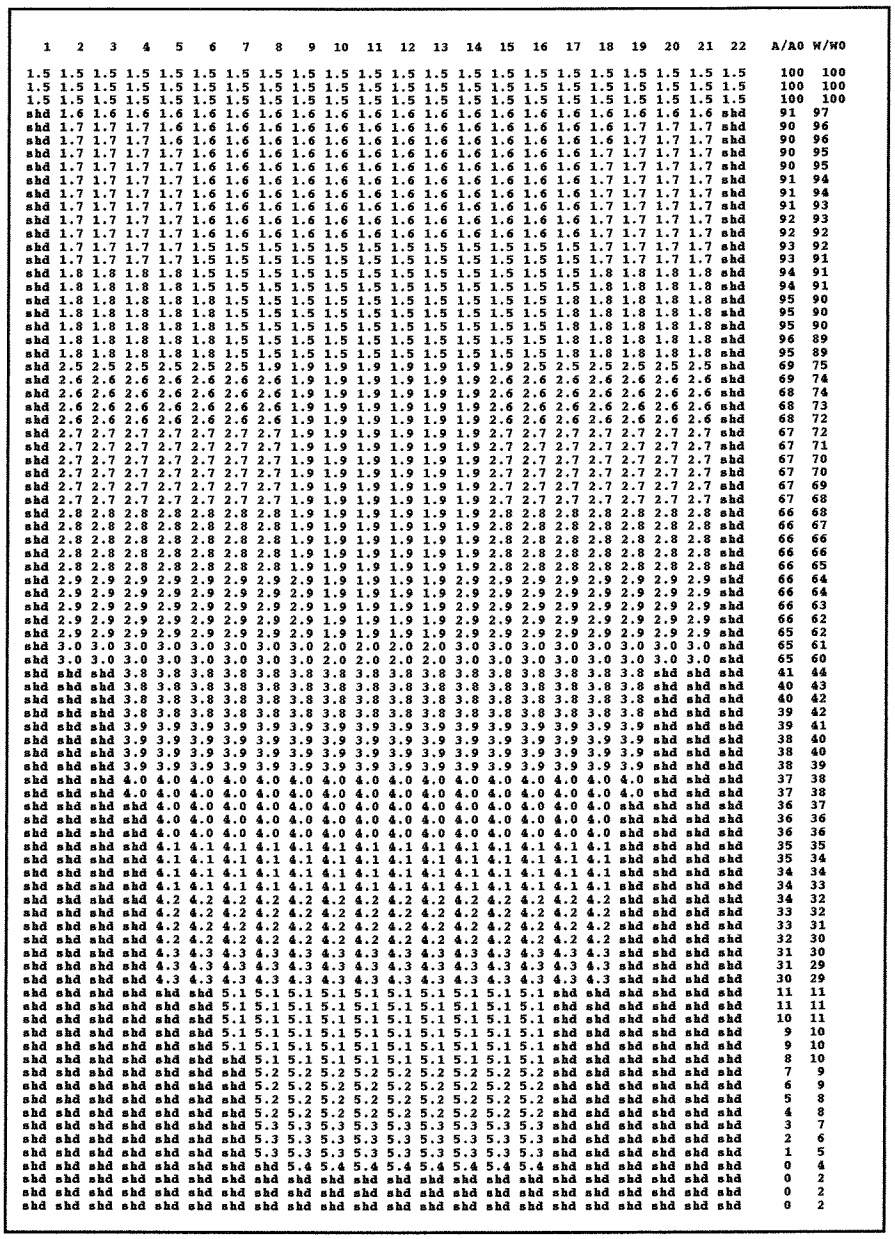


Figure D.4: Various flow- and turbulence zones

The velocity- and turbulence profiles which result from the calculations, can be made. For these profiles the product of αu is a very important factor. To do this verification, a horizontal closure is executed from both sides. At every time step, the values for α and u are written to disk. They have been presented on the next pages.

Values for α during the closure process

During the horizontal closure, the values for α have been printed in the cross section at the end of the bottom protection. In this case, the bottom protection has a total length of 600 meters; the depth h_0 is 18.6 meters. The factors A/A_0 and W/W_0 are printed as well. In this way, the turbulence factor α can be recalculated by hand.



```
 1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22     A/A0 W/W0

1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5     100  100
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5     100  100
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5     100  100
shd 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 shd      91  97
shd 1.7 1.7 1.7 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.7 1.7 1.7 shd      90  96
shd 1.7 1.7 1.7 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.7 1.7 1.7 shd      90  96
shd 1.7 1.7 1.7 1.7 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.7 1.7 1.7 1.7 shd      90  95
shd 1.7 1.7 1.7 1.7 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.7 1.7 1.7 1.7 shd      90  95
shd 1.7 1.7 1.7 1.7 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.7 1.7 1.7 1.7 shd      91  94
shd 1.7 1.7 1.7 1.7 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.7 1.7 1.7 1.7 shd      91  94
shd 1.7 1.7 1.7 1.7 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.7 1.7 1.7 1.7 shd      91  93
shd 1.7 1.7 1.7 1.7 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.7 1.7 1.7 1.7 shd      92  93
shd 1.7 1.7 1.7 1.7 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.6 1.7 1.7 1.7 1.7 shd      92  92
shd 1.7 1.7 1.7 1.7 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.7 1.7 1.7 1.7 shd      93  92
shd 1.7 1.7 1.7 1.7 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.7 1.7 1.7 1.7 shd      93  91
shd 1.8 1.8 1.8 1.8 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.8 1.8 1.8 1.8 shd      94  91
shd 1.8 1.8 1.8 1.8 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.8 1.8 1.8 1.8 shd      94  91
shd 1.8 1.8 1.8 1.8 1.8 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.8 1.8 1.8 1.8 1.8 shd      95  90
shd 1.8 1.8 1.8 1.8 1.8 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.8 1.8 1.8 1.8 1.8 shd      95  90
shd 1.8 1.8 1.8 1.8 1.8 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.8 1.8 1.8 1.8 1.8 shd      95  90
shd 1.8 1.8 1.8 1.8 1.8 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.8 1.8 1.8 1.8 1.8 shd      96  89
shd 1.8 1.8 1.8 1.8 1.8 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.8 1.8 1.8 1.8 1.8 shd      95  89
shd 2.5 2.5 2.5 2.5 2.5 2.5 1.9 1.9 1.9 1.9 1.9 1.9 1.9 1.9 2.5 2.5 2.5 2.5 2.5 2.5 shd      69  75
shd 2.6 2.6 2.6 2.6 2.6 2.6 2.6 1.9 1.9 1.9 1.9 1.9 1.9 2.6 2.6 2.6 2.6 2.6 2.6 2.6 shd      69  74
shd 2.6 2.6 2.6 2.6 2.6 2.6 2.6 1.9 1.9 1.9 1.9 1.9 1.9 2.6 2.6 2.6 2.6 2.6 2.6 2.6 shd      68  74
shd 2.6 2.6 2.6 2.6 2.6 2.6 2.6 1.9 1.9 1.9 1.9 1.9 1.9 2.6 2.6 2.6 2.6 2.6 2.6 2.6 shd      68  73
shd 2.6 2.6 2.6 2.6 2.6 2.6 2.6 1.9 1.9 1.9 1.9 1.9 1.9 2.6 2.6 2.6 2.6 2.6 2.6 2.6 shd      68  72
shd 2.7 2.7 2.7 2.7 2.7 2.7 2.7 1.9 1.9 1.9 1.9 1.9 1.9 2.7 2.7 2.7 2.7 2.7 2.7 2.7 shd      67  72
shd 2.7 2.7 2.7 2.7 2.7 2.7 2.7 1.9 1.9 1.9 1.9 1.9 1.9 2.7 2.7 2.7 2.7 2.7 2.7 2.7 shd      67  71
shd 2.7 2.7 2.7 2.7 2.7 2.7 2.7 1.9 1.9 1.9 1.9 1.9 1.9 2.7 2.7 2.7 2.7 2.7 2.7 2.7 shd      67  70
shd 2.7 2.7 2.7 2.7 2.7 2.7 2.7 1.9 1.9 1.9 1.9 1.9 1.9 2.7 2.7 2.7 2.7 2.7 2.7 2.7 shd      67  70
shd 2.7 2.7 2.7 2.7 2.7 2.7 2.7 1.9 1.9 1.9 1.9 1.9 1.9 2.7 2.7 2.7 2.7 2.7 2.7 2.7 shd      67  69
shd 2.7 2.7 2.7 2.7 2.7 2.7 2.7 1.9 1.9 1.9 1.9 1.9 1.9 2.7 2.7 2.7 2.7 2.7 2.7 2.7 shd      67  68
shd 2.8 2.8 2.8 2.8 2.8 2.8 2.8 1.9 1.9 1.9 1.9 1.9 1.9 2.8 2.8 2.8 2.8 2.8 2.8 2.8 shd      66  68
shd 2.8 2.8 2.8 2.8 2.8 2.8 2.8 1.9 1.9 1.9 1.9 1.9 1.9 2.8 2.8 2.8 2.8 2.8 2.8 2.8 shd      66  67
shd 2.8 2.8 2.8 2.8 2.8 2.8 2.8 1.9 1.9 1.9 1.9 1.9 1.9 2.8 2.8 2.8 2.8 2.8 2.8 2.8 shd      66  66
shd 2.8 2.8 2.8 2.8 2.8 2.8 2.8 1.9 1.9 1.9 1.9 1.9 1.9 2.8 2.8 2.8 2.8 2.8 2.8 2.8 shd      66  66
shd 2.8 2.8 2.8 2.8 2.8 2.8 2.8 1.9 1.9 1.9 1.9 1.9 1.9 2.8 2.8 2.8 2.8 2.8 2.8 2.8 shd      66  65
shd 2.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 1.9 1.9 1.9 1.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 shd      66  64
shd 2.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 1.9 1.9 1.9 1.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 shd      66  64
shd 2.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 1.9 1.9 1.9 1.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 shd      66  63
shd 2.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 1.9 1.9 1.9 1.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 shd      66  62
shd 2.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 1.9 1.9 1.9 1.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 2.9 shd      65  62
shd 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 2.0 2.0 2.0 2.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 shd      65  61
shd 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 2.0 2.0 2.0 2.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 shd      65  60
shd shd shd 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 shd shd shd      41  44
shd shd shd 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 shd shd shd      40  43
shd shd shd 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 shd shd shd      40  42
shd shd shd 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 shd shd shd      39  42
shd shd shd 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 shd shd shd      39  41
shd shd shd 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 shd shd shd      38  40
shd shd shd 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 shd shd shd      38  40
shd shd shd 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 3.9 shd shd shd      38  39
shd shd shd 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 shd shd shd      37  38
shd shd shd 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 shd shd shd      37  38
shd shd shd shd 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 shd shd shd shd      36  37
shd shd shd shd 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 shd shd shd shd      36  36
shd shd shd shd 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 shd shd shd shd      36  36
shd shd shd shd 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 shd shd shd shd      35  35
shd shd shd shd 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 shd shd shd shd      35  34
shd shd shd shd 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 shd shd shd shd      34  34
shd shd shd shd 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 4.1 shd shd shd shd      34  33
shd shd shd shd 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 shd shd shd shd      34  32
shd shd shd shd 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 shd shd shd shd      33  32
shd shd shd shd 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 shd shd shd shd      33  31
shd shd shd shd 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 4.2 shd shd shd shd      32  30
shd shd shd shd 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 shd shd shd shd      31  30
shd shd shd shd 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 shd shd shd shd      31  29
shd shd shd shd 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 4.3 shd shd shd shd      30  29
shd shd shd shd shd shd 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 shd shd shd shd shd shd      11  11
shd shd shd shd shd shd 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 shd shd shd shd shd shd      11  11
shd shd shd shd shd shd 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 shd shd shd shd shd shd      10  11
shd shd shd shd shd shd 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 shd shd shd shd shd shd       9  10
shd shd shd shd shd shd 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 shd shd shd shd shd shd       9  10
shd shd shd shd shd shd shd 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 5.1 shd shd shd shd shd shd       8  10
shd shd shd shd shd shd shd 5.2 5.2 5.2 5.2 5.2 5.2 5.2 5.2 5.2 shd shd shd shd shd shd       7   9
shd shd shd shd shd shd shd 5.2 5.2 5.2 5.2 5.2 5.2 5.2 5.2 5.2 shd shd shd shd shd shd       6   9
shd shd shd shd shd shd shd 5.2 5.2 5.2 5.2 5.2 5.2 5.2 5.2 5.2 shd shd shd shd shd shd       5   8
shd shd shd shd shd shd shd 5.2 5.2 5.2 5.2 5.2 5.2 5.2 5.2 5.2 shd shd shd shd shd shd       4   8
shd shd shd shd shd shd shd 5.3 5.3 5.3 5.3 5.3 5.3 5.3 5.3 5.3 shd shd shd shd shd shd       3   7
shd shd shd shd shd shd shd 5.3 5.3 5.3 5.3 5.3 5.3 5.3 5.3 5.3 shd shd shd shd shd shd       2   6
shd shd shd shd shd shd shd 5.3 5.3 5.3 5.3 5.3 5.3 5.3 5.3 5.3 shd shd shd shd shd shd       1   5
shd shd shd shd shd shd shd shd 5.4 5.4 5.4 5.4 5.4 5.4 5.4 5.4 shd shd shd shd shd shd       0   4
shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd       0   2
shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd       0   2
shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd shd       0   2
```

Calculations started; Horizontal closure started

Local Turbulence + flow zone

Closure advances

Higher values for α

Closure finished

Figure D.5: Values for turbulence for horizontal closure

It must be mentioned, that in these figures only the basin side of the construction has been considered; the data has only been written to disk during inflow. Because also during outflow the narrowing of the gap continues, sudden decreases of the width (and thus increases of α) can be noticed.

Values for the velocity u during the closure process

For the same case, the values for u have been printed in the cross section at the end of the bottom protection:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	A/A0	W/W0	u
0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	100	100	0.8
0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	100	100	0.4
0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	100	100	0.6
shd	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.1	shd	82	88	0.3
shd	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	shd	82	88	0.0
shd	0.2	0.5	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.5	0.2	shd	81	87	0.8
shd	0.2	0.5	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.5	0.2	shd	81	87	0.8
shd	0.3	0.6	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.6	0.3	shd	81	86	1.0
shd	0.3	0.7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	0.3	shd	80	86	1.2
shd	0.4	0.8	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	0.8	0.4	shd	80	85	1.3
shd	0.4	0.9	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	0.9	0.4	shd	80	84	1.5
shd	0.4	0.9	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	0.9	0.4	shd	80	84	1.7
shd	0.5	1.0	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.0	0.5	shd	80	83	1.8
shd	0.5	1.0	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.0	0.5	shd	80	82	1.9
shd	0.5	1.0	1.5	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.5	1.0	0.5	shd	81	82	2.0
shd	0.5	1.0	1.5	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.5	1.0	0.5	shd	81	81	2.0
shd	0.5	1.0	1.5	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.5	1.0	0.5	shd	81	80	2.1
shd	0.4	0.9	1.4	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.4	0.9	0.4	shd	81	80	2.0
shd	0.4	0.9	1.4	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.4	0.9	0.4	shd	81	79	2.0
shd	0.4	0.8	1.3	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.3	0.8	0.4	shd	81	78	1.9
shd	0.3	0.7	1.1	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.1	0.7	0.3	shd	81	78	1.8
shd	0.3	0.6	1.0	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.0	0.6	0.3	shd	81	77	1.6
shd	0.2	0.5	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.8	0.5	0.2	shd	81	76	1.3
shd	0.2	0.4	0.6	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.6	0.4	0.2	shd	81	76	1.0
shd	0.1	0.2	0.3	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0.2	0.1	shd	80	75	0.5
shd	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.0	0.0	0.0	shd	79	74	0.1
shd	0.0	0.0	0.1	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.1	0.0	0.0	shd	55	59	0.5
shd	0.0	0.1	0.2	0.2	0.3	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0.2	0.2	0.1	0.0	shd	54	58	0.6
shd	0.0	0.1	0.2	0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.5	0.4	0.2	0.1	0.0	shd	54	58	0.9
shd	0.0	0.1	0.3	0.4	0.6	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.6	0.4	0.3	0.1	0.0	shd	53	57	1.2
shd	shd	0.1	0.3	0.5	0.7	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.7	0.5	0.3	0.1	shd	shd	53	56	1.4
shd	shd	0.2	0.4	0.6	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.8	0.6	0.4	0.2	shd	shd	52	56	1.7
shd	shd	0.2	0.4	0.7	0.9	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	0.9	0.7	0.4	0.2	shd	shd	52	55	1.9
shd	shd	0.2	0.4	0.7	1.0	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.0	0.7	0.4	0.2	shd	shd	52	54	2.1
shd	shd	0.2	0.5	0.8	1.1	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.1	0.8	0.5	0.2	shd	shd	52	54	2.3
shd	shd	0.1	0.5	0.8	1.1	1.4	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.4	1.1	0.8	0.5	0.1	shd	shd	51	53	2.5
shd	shd	0.1	0.5	0.8	1.1	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.1	0.8	0.5	0.1	shd	shd	51	52	2.6
shd	shd	0.1	0.5	0.8	1.2	1.5	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.5	1.2	0.8	0.5	0.1	shd	shd	51	52	2.7
shd	shd	0.1	0.5	0.8	1.2	1.5	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.5	1.2	0.8	0.5	0.1	shd	shd	51	51	2.8
shd	shd	0.1	0.4	0.8	1.2	1.5	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.5	1.2	0.8	0.4	0.1	shd	shd	50	50	2.9
shd	shd	0.0	0.4	0.8	1.1	1.5	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.5	1.1	0.8	0.4	0.0	shd	shd	50	50	2.9
shd	shd	0.0	0.4	0.7	1.1	1.5	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.5	1.1	0.7	0.4	0.0	shd	shd	50	49	2.9
shd	shd	0.0	0.3	0.7	1.0	1.4	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.4	1.0	0.7	0.3	0.0	shd	shd	50	48	2.8
shd	shd	0.0	0.3	0.6	1.0	1.3	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.3	1.0	0.6	0.3	0.0	shd	shd	49	48	2.7
shd	shd	shd	0.3	0.6	0.9	1.2	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.2	0.9	0.6	0.3	shd	shd	shd	49	47	2.6
shd	shd	shd	0.2	0.5	0.8	1.1	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.1	0.8	0.5	0.2	shd	shd	shd	49	46	2.4
shd	shd	shd	0.2	0.4	0.7	0.9	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	0.9	0.7	0.4	0.2	shd	shd	shd	48	46	2.1
shd	shd	shd	0.1	0.3	0.5	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.8	0.5	0.3	0.1	shd	shd	shd	48	45	1.7
shd	shd	shd	0.1	0.2	0.4	0.5	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.5	0.4	0.2	0.1	shd	shd	shd	47	44	1.2
shd	shd	shd	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.0	0.0	0.0	shd	shd	shd	47	44	0.2
shd	shd	shd	shd	shd	0.0	0.1	0.2	0.3	0.4	0.4	0.4	0.4	0.3	0.2	0.1	0.0	shd	shd	shd	shd	shd	26	27	0.9
shd	shd	shd	shd	shd	0.1	0.2	0.4	0.5	0.6	0.6	0.6	0.6	0.5	0.4	0.2	0.1	shd	shd	shd	shd	shd	25	27	1.5
shd	shd	shd	shd	shd	0.1	0.3	0.5	0.7	0.8	0.8	0.8	0.8	0.7	0.5	0.3	0.1	shd	shd	shd	shd	shd	24	26	2.0
shd	shd	shd	shd	shd	0.1	0.3	0.5	0.8	1.0	1.0	1.0	1.0	0.8	0.5	0.3	0.1	shd	shd	shd	shd	shd	24	25	2.3
shd	shd	shd	shd	shd	0.1	0.3	0.6	0.8	1.1	1.1	1.1	1.1	0.8	0.6	0.3	0.1	shd	shd	shd	shd	shd	23	25	2.7
shd	shd	shd	shd	shd	0.1	0.4	0.6	0.9	1.1	1.2	1.2	1.1	0.9	0.6	0.4	0.1	shd	shd	shd	shd	shd	23	24	3.0
shd	shd	shd	shd	shd	0.1	0.4	0.6	0.9	1.2	1.3	1.3	1.2	0.9	0.6	0.4	0.1	shd	shd	shd	shd	shd	22	23	3.2
shd	shd	shd	shd	shd	0.1	0.4	0.7	0.9	1.2	1.3	1.3	1.2	0.9	0.7	0.4	0.1	shd	shd	shd	shd	shd	22	23	3.5
shd	shd	shd	shd	shd	0.1	0.4	0.7	1.0	1.3	1.4	1.4	1.3	1.0	0.7	0.4	0.1	shd	shd	shd	shd	shd	21	22	3.7
shd	shd	shd	shd	shd	0.0	0.3	0.7	1.0	1.3	1.4	1.4	1.3	1.0	0.7	0.3	0.0	shd	shd	shd	shd	shd	21	21	3.8
shd	shd	shd	shd	shd	0.0	0.3	0.6	1.0	1.3	1.4	1.4	1.3	1.0	0.6	0.3	0.0	shd	shd	shd	shd	shd	20	21	4.0
shd	shd	shd	shd	shd	0.0	0.3	0.6	0.9	1.2	1.4	1.4	1.2	0.9	0.6	0.3	0.0	shd	shd	shd	shd	shd	20	20	4.1
shd	shd	shd	shd	shd	0.0	0.3	0.6	0.9	1.2	1.4	1.4	1.2	0.9	0.6	0.3	0.0	shd	shd	shd	shd	shd	19	19	4.2
shd	shd	shd	shd	shd	0.0	0.3	0.6	0.9	1.2	1.4	1.4	1.2	0.9	0.6	0.3	0.0	shd	shd	shd	shd	shd	18	19	4.2
shd	shd	shd	shd	shd	shd	0.2	0.5	0.8	1.1	1.4	1.4	1.1	0.8	0.5	0.2	shd	shd	shd	shd	shd	shd	18	18	4.2
shd	shd	shd	shd	shd	shd	0.2	0.5	0.8	1.1	1.3	1.3	1.1	0.8	0.5	0.2	shd	shd	shd	shd	shd	shd	17	17	4.2
shd	shd	shd	shd	shd	shd	0.2	0.5	0.7	1.0	1.3	1.3	1.0	0.7	0.5	0.2	shd	shd	shd	shd	shd	shd	17	17	4.1
shd	shd	shd	shd	shd	shd	0.1	0.4	0.7	0.9	1.2	1.2	0.9	0.7	0.4	0.1	shd	shd	shd	shd	shd	shd	16	16	4.0
shd	shd	shd	shd	shd	shd	0.1	0.4	0.6	0.9	1.1	1.1	0.9	0.6	0.4	0.1	shd	shd	shd	shd	shd	shd	15	15	3.9
shd	shd	shd	shd	shd	shd	0.1	0.3	0.5	0.8	1.0	1.0	0.8	0.5	0.3	0.1	shd	shd	shd	shd	shd	shd	15	15	3.6
shd	shd	shd	shd	shd	shd	0.1	0.3	0.5	0.7	0.9	0.9	0.7	0.5	0.3	0.1	shd	shd	shd	shd	shd	shd	14	14	3.4
shd	shd	shd	shd	shd	shd	0.0	0.2	0.4	0.6	0.8	0.8	0.6	0.4	0.2	0.0	shd	shd	shd	shd	shd	shd	14	13	3.1
shd	shd	shd	shd	shd	shd	0.0	0.2	0.3	0.5	0.6	0.6	0.5	0.3	0.2	0.0	shd	shd	shd	shd	shd	shd	13	13	2.7
shd	shd	shd	shd	shd	shd	0.0	0.1	0.2	0.4	0.5	0.5	0.4	0.2	0.1	0.0	shd	shd	shd	shd	shd	shd	12	12	2.1
shd	shd	shd	shd	shd	shd	0.0	0.0	0.1	0.2	0.3	0.3	0.2	0.1	0.0	0.0	shd	shd	shd	shd	shd	shd	12	11	1.4
shd	shd	shd	shd	shd	shd	shd	shd	0.0	0.1	0.2	0.2	0.1	0.0	shd	shd	shd	shd	shd	shd	shd	shd	5	5	1.0
shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	0	0	0.0
shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	0	0	0.0

Initial profile; Horizontal closure started

Local turbulence zone noted in linearly increasing velocity

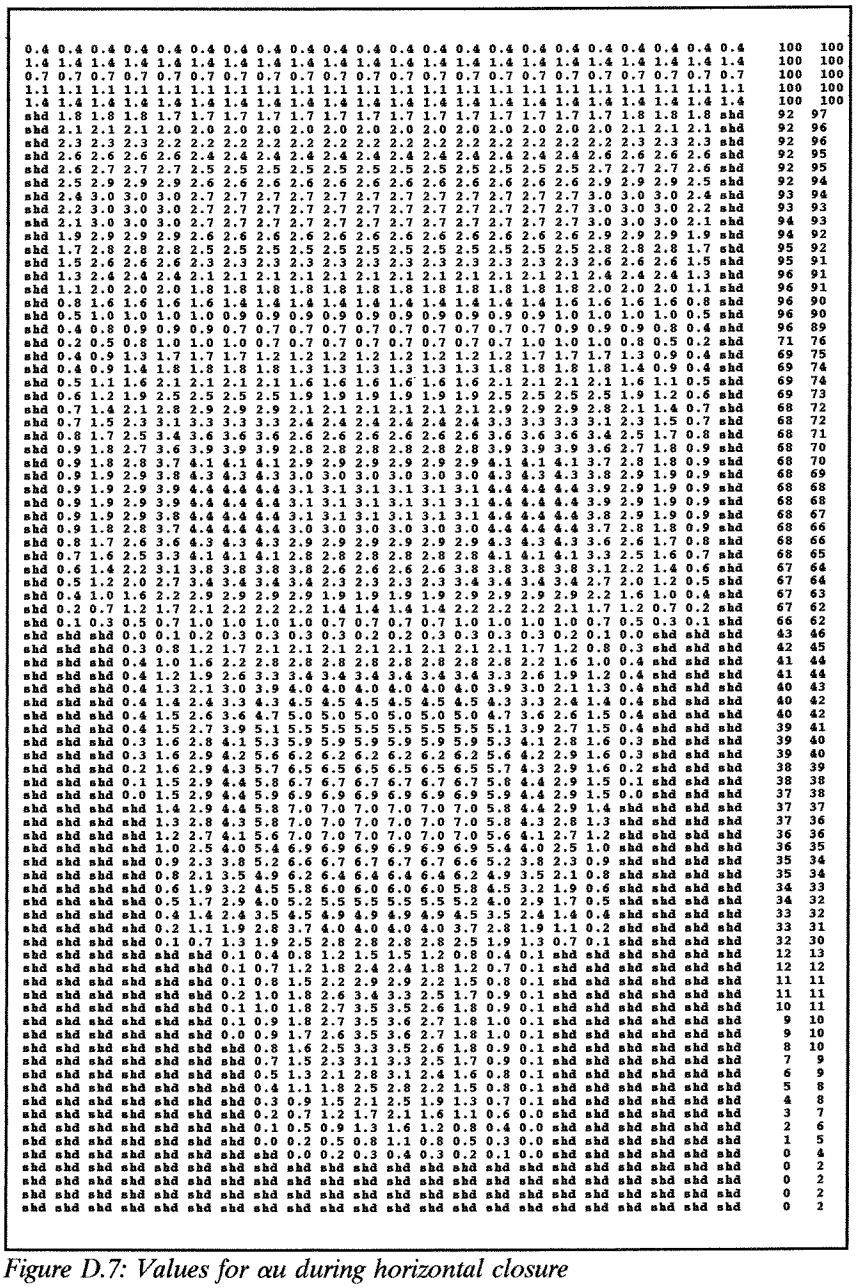
Velocity in gap differs from velocity at end of bottom protection

Closure finished

Figure D.6: Velocities during horizontal closure

Combination of calculated turbulence factor α and velocity u

Below, the resulting product of the values for α and u have been printed in the cross section at the end of the bottom protection:



Initial profile; Horizontal closure started

Local turbulence zone noted in linear increasing velocity

Closure finished

Figure D.7: Values for αu during horizontal closure

The results have also been printed in the case the same gap is closed vertically:

Initial profile; vertical closure started

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	A/A0	W/W0
1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	100	100
1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	100	100
1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	100	100
1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	100	100
1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	100	100
1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	67	100
1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	67	100
1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	66	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	65	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	64	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	64	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	62	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	63	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	32	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	32	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	31	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	31	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	31	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	31	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	31	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	31	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	31	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	31	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	31	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	31	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	31	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	32	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	32	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	32	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	32	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	32	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	32	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	32	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	32	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	32	100
2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	32	100
2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	12	100
2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	11	100
2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	11	100
2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	11	100
2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	10	100
2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	10	100
2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	10	100
2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	10	100
2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	11	100
2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	11	100
2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	11	100
2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	11	100
2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	10	100
2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	10	100
2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	10	100
2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	9	100
2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	9	100
2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	8	100
2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	8	100
2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	7	100
2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	7	100
2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	7	100
2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	6	100
shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	0	100
shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	0	100
shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	0	100
shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	0	100
shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	shd	0	100

Closure finished

Figure D.8: Turbulence factor α during vertical closure

In this case it can be seen that the scour processes can be described one-dimensionally: in every section the same α (and u) occur.

The comparison between the velocity in the gap and the velocity at the end of the bottom protection can be drawn in a graph (figure *D9*), in which the velocities are drawn as their absolute value:

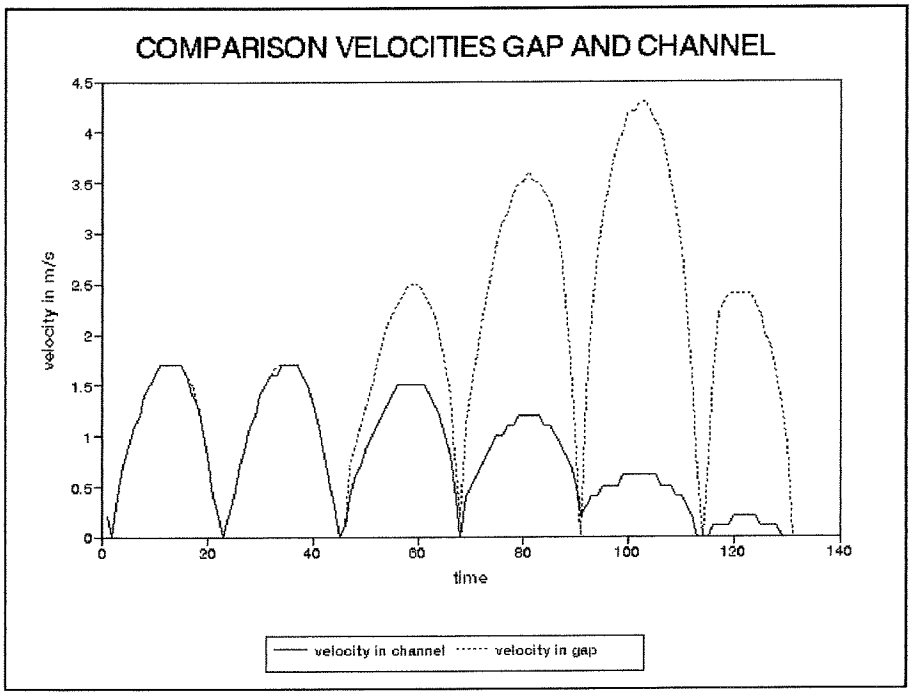


Figure D.9: Comparison velocities in gap and channel

ad b.

In order to verify the use of integrals in relation of Breusers, which has been used for the calculation of the scour holes behind the bottom protection, a similar procedure has been written. The approach of this test has been as follows:

* A certain bottom protection length has been entered;
* The values for the turbulence α and the velocity u have been taken constant (their validity has been verified *ad a*);
* The current attack takes place during a certain period of time.

After the results of the calculations have been written to disk, they can be verified directly by entering the same values in Breuser's relation.

The test has been executed for two situations:

* Bottom protection length 300 meters (effective length for the calculation 150 meters), turbulence factor α=2, velocity u=1 m/s, scouring during 5 days;

* Bottom protection length 550 meters (effective length for the calculation 275 meters), turbulence factor α=4, velocity u=2 m/s, scouring during 15 days.

The source code of the test procedure is:

```
Procedure TestBedDestruction;

var OutP: Text;
    Breusers: Real;
    TestTimeHours, CC: Integer;
    TSK: Real;

begin
   GapSurface[1]:=1;
   NumOfSnaps[1]:=5;
   Construction[1]:=Busy;
   StandLengthProt[1]:=100;
   SnapWidth[1]:=5;
   NumOfGaps:=1;
   OrigTotalGapSurface:=5000;
   GapDepthAtMSL[1]:=1000;
   StaticsOn:=True;
   Assign(OutP, 'BED2');
   ReWrite(OutP);
   BottomD50:=4;
   Chezy:=70;
   StaticAlpha:=2.5;
   StaticVelocityEndBP:=1.3;
   OrigGapDepthAtMSL[1]:=10;
   WriteLn(OutP,'   TIME ELAPSED          DEPTH in PROGRAM        DEPTH acc
                BREUSERS');
   WriteLn(OutP,'============================================================');
   for CC:=0 to 13 do
   begin
      TestTimeHours:=CC*10;
      Repeat
          TSK:=TimeSinceKent;
          BedDestruction;
      Until TimeSinceKent/3600>=TestTimeHours;
      NumOfKenterings:=5;
      Kentering;
      Breusers:=Power(StaticAlpha*StaticVelocityEndBP-CriticVelocity,1.7)*
                 Power(OrigGapDepthAtMSL[1],0.2)*Power((TSK/3600), 0.4)/
                 (10*Power(Delta,0.7));
      WriteLn(OutP, TSK,'        ',InHoleDepth[1,2],'        ',Breusers);
      AUUC34[1,2]:=0;
      InHoleDepth[1,2]:=0;
      InHoleScour[1,2]:=0;
   end;
   Close(OutP);
   halt;
end;
```

For two cases, the program's method to calculate the scour depth has been verified:

	CASE 1	*CASE 2*
Grain size (D50)	1 mm	4 mm
Roughness (Chezy)	50 √m/s	70 √m/s
Turbulence (α)	2	2.5
Velocity (u)	2 m/s	1.3 m/s
Original depth (h_0)	20 m	10 m

The calculations have been done for several durations. In the table of results below, in the right column the depth has been calculated applying the Breuser's formula directly. For case one this yields (the *elapsed time* has been expressed in hours):

TIME ELAPSED	DEPTH in PROGRAM	DEPTH acc BREUSERS
0.0000000000E+00	0.0000000000E+00	0.0000000000E+00
3.5000000000E+04	2.8023004628E+00	2.8023004628E+00
7.1000000000E+04	3.7186972299E+00	3.7186972299E+00
1.0700000000E+05	4.3816833758E+00	4.3816833757E+00
1.4300000000E+05	4.9206434341E+00	4.9206434340E+00
1.7900000000E+05	5.3830534637E+00	5.3830534636E+00
2.1500000000E+05	5.7924576169E+00	5.7924576167E+00
2.5100000000E+05	6.1625005265E+00	6.1625005263E+00
2.8700000000E+05	6.5018993079E+00	6.5018993079E+00
3.2300000000E+05	6.8166106651E+00	6.8166106652E+00
3.5900000000E+05	7.1109112808E+00	7.1109112810E+00
3.9500000000E+05	7.3879903417E+00	7.3879903420E+00
4.3100000000E+05	7.6502986410E+00	7.6502986414E+00
4.6700000000E+05	7.8997661054E+00	7.8997661059E+00

for the second case, the values are:

TIME ELAPSED	DEPTH in PROGRAM	DEPTH acc BREUSERS
0.0000000000E+00	0.0000000000E+00	0.0000000000E+00
3.5000000000E+04	9.8870999530E-01	9.8870999530E-01
7.1000000000E+04	1.3120338698E+00	1.3120338699E+00
1.0700000000E+05	1.5459491968E+00	1.5459491968E+00
1.4300000000E+05	1.7361055358E+00	1.7361055358E+00
1.7900000000E+05	1.8992534295E+00	1.8992534296E+00
2.1500000000E+05	2.0436997456E+00	2.0436997456E+00
2.5100000000E+05	2.1742585948E+00	2.1742585948E+00
2.8700000000E+05	2.2940055570E+00	2.2940055570E+00
3.2300000000E+05	2.4050422816E+00	2.4050422816E+00

3.5900000000E+05	2.5088776713E+00	2.5088776712E+00
3.9500000000E+05	2.6066369376E+00	2.6066369375E+00
4.3100000000E+05	2.6991847715E+00	2.6991847713E+00
4.6700000000E+05	2.7872020912E+00	2.7872020910E+00

From this, it can be concluded that the calculation of Breuser's integral per timestep, as has been described in chapter 5.3, does not effect the results at all. Hence, the application of Breuser's formula has been done correctly. The scour which is calculated in *case 2*, can be presented as a graph:

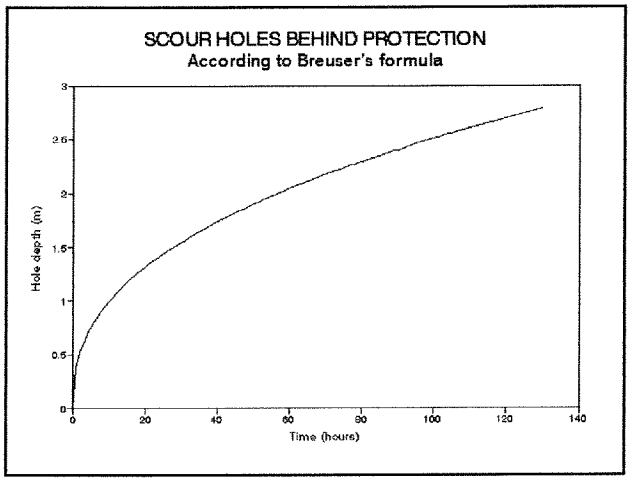


Figure D.10: Scour depth resulting from program

ad c.

As explained in chapter 6.5, the scour hole depth is reduced by an upstream sediment supply, of the sediment transport that allways occurs in the equilibrium profile of the channel. The upstream sediment transport is stored in the scour hole, of which the dimensions per meter width are calculated as:

$$I_t = b\,h_{max}^2(t) \qquad (per\ m\ width)$$

in which b is a shape factor and I_t is the quantity of material discharged until then. If the upstream material supply is T_s per meter width, the reduced quantity of material will be:

$$I_{t,red} = I_t - T_s t$$

because of this reduced scour hole volume, the reduced depth can be expressed as:

$$h_{\max,red}(t) = \sqrt{h_{\max}^2(t) - \frac{T_s t}{b}}$$

It is interesting to look to the reduction $T_s t / b$ in proportion to the square of the scour according to Breusers, h_{max}. Thereto, a horizontal closure from the right side has been executed while at one section these two values have been registered. The resulting, reduced scour depth has been printed in the first column:

RESULTING DEPTH	*DEPTH BY BREUSERS*	*(BREUSERS)²*	*FACTOR ($T_s t$)/b*	*REMARK*
0.0000000E+00	5.563543E-01	3.095301E-01	1.870808E+01	(1)
0.0000000E+00	7.341830E-01	5.390246E-01	1.870804E+01	
0.0000000E+00	8.635352E-01	7.456931E-01	1.870797E+01	
0.0000000E+00	9.689311E-01	9.388275E-01	1.870791E+01	
0.0000000E+00	1.065209E+00	1.134671E+00	1.839781E+01	
0.0000000E+00	1.153575E+00	1.330735E+00	1,821319E+01	
0.0000000E+00	1.237965E+00	1.532559E+00	1.785335E+01	
0.0000000E+00	1.319946E+00	1.742257E+00	1.743966E+01	
0.0000000E+00	1.400393E+00	1.961102E+00	1.678701E+01	
0.0000000E+00	1.482856E+00	2.198864E+00	1.566904E+01	
0.0000000E+00	1.582654E+00	2.504794E+00	1.425497E+01	
0.0000000E+00	1.720404E+00	2.959790E+00	1.252951E+01	
0.0000000E+00	1.990171E+00	3.960782E+00	1.038710E+01	
0.0000000E+00	2.735214E+00	7.481400E+00	7.912186E+00	
2.5403442E+00	3.419974E+00	1.169622E+01	5.242878E+00	(2)
3.6936541E+00	4.051217E+00	1.641236E+01	2.769282E+00	
4.1496045E+00	4.271150E+00	1.824272E+01	1.023509E+00	
4.2728194E+00	4.295910E+00	1.845484E+01	1.978605E-01	
4.2950992E+00	4.296276E+00	1.845799E+01	1.011420E-02	(3)

It can be seen, that the scour reducing process which results from the program is as expected. The following remarks have been made:

(1) When the flow profile is still big, the scour reduction is much bigger than the scour which occurs by the turbulent flow. In the program, the assumption has been made that no sedimentation takes place, and thus, there is no scour hole at all.

(2) At a certain stage, the reduction is muss less because of the reduced flow

through the gap; the resulting velocities in the channel are low. Then, the turbulent flow has the possibility to cause a scour hole;

(3) When the gap is nearly closed, the reducing sediment supply is zero. It can be seen that the resulting 'reduced scour hole' is equal to the scour hole according to Breuser's formula.

3. Verification of the Level-Flow relation

As explained in *Appendix A*, the level-flow calculation is programmed numerically, as an explicit scheme of the storage model. The verification of this calculation can be divided into three parts:

1. The gap flow, divided by the gap surface, should equal the velocity calculated with the weir formula; (verification of the weir formula)

2. The inflowing water should cause a level rise in the basin which should be equal to Q_{infl} / $A_{storage}$ (the inflow divided by the storage area of the basin) (verification of the continuity equation)

3. The velocity when the current is critical, should remain independent of the downstream water level.

ad 1.

The application of the weir formula could be checked by verifying the flow per gap, devided by the surface of that gap, which should be equal to the velocity calculated with the weir formula:

$$\frac{Q_{gap}}{A_{gap}} = \sqrt{2\,g\,\Delta H}$$

The ΔH in this calculation, should be equal to the ΔH which has been used in the explicit calculation scheme:

$$\Delta H = \frac{(h_{b1}-h_{a1})+(h_{b2}-h_{a2})}{2} = \frac{h_{b1}-h_{b2}-2\,h_{a1} - \Delta t\frac{Q}{F}}{2}$$

To do this, to the program's level-flow calculation the following code was added:

```
for GCT:=1 to NumOfGaps do
  begin
     Velocity[GCT]:=GapFlow[GCT]/GapSurface[GCT];          {Procedure's velocity}
     VeloReal[GCT]:=FlowDirection*SQRT(                  {Velocity acc. weir forumla}
            Abs(2*Gravity*(MaxLevel*2-
            CalcMinLevel*2 -
            TimeStep*LastTotalFlow/StorageSurface(OldBasinLevel))/2));
     Write(OutP, Velocity[GCT]);              {Write procedure's velocity to disk}
  end;
  WriteLn(OutP, VeloReal[GCT]);             {Write weir formula's velocity to disk}
```

The file, which is written to disk, contains the velocity in all gaps according to the left handside of the formula; the last value of each line is the velocity according to the weir formula. The results of a calculation of two gaps has been given below:

TIME IN HOURS	Q/A of GAP 1	Q/A of GAP 2	VELOCITY ON WEIR
2.0000000000E+01	1.7998569820E+00	1.7998569820E+00	1.7998570398E+00
2.0277777778E+01	1.8084541074E+00	1.8084541074E+00	1.8084544829E+00
2.0555555556E+01	1.7829949736E+00	1.7829949736E+00	1.7829950804E+00
2.0833333334E+01	1.7214173606E+00	1.7214173606E+00	1.7214170604E+00
2.1111111111E+01	1.6216632558E+00	1.6216632558E+00	1.6216631392E+00
2.1388888889E+01	1.4815387417E+00	1.4815387417E+00	1.4815384100E+00
2.1666666667E+01	1.2984123160E+00	1.2984123160E+00	1.2984121808E+00
2.1944444445E+01	1.0685138978E+00	1.0685138978E+00	1.0685132972E+00
2.2222222222E+01	7.8479759615E-01	7.8479759615E-01	7.8479663770E-01
2.2500000000E+01	4.2598389179E-01	4.2598389179E-01	4.2598238006E-01
2.2777777778E+01	-2.1364730810E-01	-2.1364730810E-01	-2.1365072092E-01
2.3333333334E+01	-9.8196138285E-01	-9.8196138285E-01	-9.8196168161E-01
2.3611111111E+01	-1.5440621957E-01	-1.5440621957E-01	-1.5440055032E-01
2.3888888889E+01	-9.1620873862E-01	-9.1620873862E-01	-9.1620894200E-01
2.4166666667E+01	-8.6547137412E-01	-8.6547137412E-01	-8.6547080176E-01
2.4444444445E+01	-1.0372330227E+00	-1.0372330227E+00	-1.0372335566E+00
2.4722222222E+01	-1.1929691510E+00	-1.1929691510E+00	-1.1929693430E+00
2.5000000000E+01	-1.3351809074E+00	-1.3351809074E+00	-1.3351815160E+00
2.5277777778E+01	-1.4618637082E+00	-1.4618637082E+00	-1.4618640050E+00
2.5555555556E+01	-1.5708204046E+00	-1.5708204046E+00	-1.5708205533E+00
2.5833333334E+01	-1.6597533033E+00	-1.6597533033E+00	-1.6597533759E+00
2.6111111111E+01	-1.7262475059E+00	-1.7262475059E+00	-1.7262478104E+00
2.6388888889E+01	-1.7677393250E+00	-1.7677393250E+00	-1.7677397225E+00
2.6666666667E+01	-1.7814660489E+00	-1.7814660489E+00	-1.7814658742E+00
2.6944444445E+01	-1.7643892964E+00	-1.7643892964E+00	-1.7643889071E+00
2.7222222222E+01	-1.7130763813E+00	-1.7130763813E+00	-1.7130763174E+00
2.7500000000E+01	-1.6234866250E+00	-1.6234866250E+00	-1.6234865121E+00
2.7777777778E+01	-1.4905562437E+00	-1.4905562437E+00	-1.4905560198E+00
2.8055555556E+01	-1.3072935699E+00	-1.3072935699E+00	-1.3072930180E+00
2.8333333334E+01	-1.0623496592E+00	-1.0623496592E+00	-1.0623493769E+00
2.8611111111E+01	-7.3072787160E-01	-7.3072787160E-01	-7.3072726718E-01
2.8888888889E+01	-1.4647337067E-01	-1.4647337067E-01	-1.4646828722E-01
2.9166666667E+01	2.6970142279E-01	2.6970142279E-01	2.6970475521E-01
3.0000000000E+01	7.8697538331E-01	7.8697538331E-01	7.8697465092E-01
3.0277777778E+01	9.6370078393E-01	9.6370078393E-01	9.6370117475E-01
3.0555555556E+01	1.1442276063E+00	1.1442276063E+00	1.1442277580E+00
3.0833333334E+01	1.3095595211E+00	1.3095595211E+00	1.3095600712E+00
3.1111111111E+01	1.4559621464E+00	1.4559621464E+00	1.4559624507E+00
3.1388888889E+01	1.5805352394E+00	1.5805352394E+00	1.5805354257E+00
3.1666666667E+01	1.6805986468E+00	1.6805986468E+00	1.6805987701E+00
3.1944444445E+01	1.7536205205E+00	1.7536205205E+00	1.7536206062E+00
3.2222222223E+01	1.7972065566E+00	1.7972065566E+00	1.7972066166E+00
3.2500000000E+01	1.8090930051E+00	1.8090930051E+00	1.8090934041E+00

This concludes, that the values calculated by the explicit scheme and the values resulting from the weir formula, agree. Small differences are due to the truncation of the computer's calculation process.

ad 2.

In order to check the application of the continuity equation in the simulations, the following can be done. First, the basin is assumed to have vertical banks (which is expressed by a constant storage function). Then a constant flow enters into the basin.

This can be done by entering a river flow, while the tidal amplitudes are zero. For the level rise it is known::

$$h_{a2} = h_{a1} + \Delta t \frac{Q}{F}$$

which is the continuity equation. This test has been done; the results are printed below:

TIME	TIMESTEP	INFLOW	STORAGE AREA	LEVEL RISE	BASIN LEVEL
0.00 h	500 s	600 m3/s	5000000 m2	0.06 m	0.0000000000E+00
0.13 h	500 s	600 m3/s	5000000 m2	0.06 m	6.0000000000E-02
0.27 h	500 s	600 m3/s	5000000 m2	0.06 m	1.2000000000E-01
0.41 h	500 s	600 m3/s	5000000 m2	0.06 m	1.8000000000E-01
0.55 h	500 s	600 m3/s	5000000 m2	0.06 m	2.4000000000E-01
0.69 h	500 s	600 m3/s	5000000 m2	0.06 m	3.0000000000E-01
0.83 h	500 s	600 m3/s	5000000 m2	0.06 m	3.6000000000E-01
0.97 h	500 s	600 m3/s	5000000 m2	0.06 m	4.2000000000E-01
1.11 h	500 s	600 m3/s	5000000 m2	0.06 m	4.8000000000E-01
1.25 h	500 s	600 m3/s	5000000 m2	0.06 m	5.4000000000E-01
1.38 h	500 s	600 m3/s	5000000 m2	0.06 m	6.0000000000E-01
1.52 h	500 s	600 m3/s	5000000 m2	0.06 m	6.6000000000E-01
1.66 h	500 s	600 m3/s	5000000 m2	0.06 m	7.2000000000E-01
1.80 h	500 s	600 m3/s	5000000 m2	0.06 m	7.8000000000E-01
1.94 h	500 s	600 m3/s	5000000 m2	0.06 m	8.4000000000E-01
2.08 h	500 s	600 m3/s	5000000 m2	0.06 m	9.0000000000E-01
2.22 h	500 s	600 m3/s	5000000 m2	0.06 m	9.6000000000E-01
2.36 h	500 s	600 m3/s	5000000 m2	0.06 m	1.0200000000E+00
2.50 h	500 s	600 m3/s	5000000 m2	0.06 m	1.0800000000E+00
2.63 h	500 s	600 m3/s	5000000 m2	0.06 m	1.1400000000E+00

The level rise in the basin seems to be calculated correctly, according to the continuity equation. Of both tests *1* and *2*, it can be seen that the application of the explicit scheme for the level-flow calculations is applicated correctly in the simultation program.

ad 3.

To verify whether within the weir formula the critical and noncritical flows are considered correctly, the following test is done: water flows through a gap, while the sill is raised. If the level differences of every time step are written to disk, as well as the velocity, a verification can be made whether this velocity is calculated well. Some results of this test, considering the weir becoming critical, are printed below:

#	MAX.LEVEL	MIN.LEVEL	CALC.MINLEVEL	VELOCITY	LEV.DIFF	WEIR FLOW
1	3.12	2.73	2.73	2.77	0.39	NORM
2	3.04	2.55	2.55	3.10	0.49	NORM
3	2.96	2.38	2.38	3.40	0.59	NORM
4	2.88	2.20	2.20	3.65	0.68	NORM
5	2.80	2.03	2.03	3.89	0.77	NORM
6	2.72	1.87	1.87	4.08	0.85	NORM
7	2.64	1.73	1.76	4.16	0.88	CRIT
8	2.56	1.59	1.71	4.08	0.85	CRIT
9	2.48	1.47	1.66	4.04	0.83	CRIT
0	2.41	1.37	1.61	3.99	0.81	CRIT
11	2.34	1.28	1.56	3.91	0.78	CRIT
12	2.27	1.22	1.52	3.86	0.76	CRIT
13	2.20	1.17	1.47	3.81	0.74	CRIT
14	2.13	1.15	1.43	3.73	0.71	CRIT
15	2.07	1.15	1.38	3.68	0.69	CRIT
16	2.00	1.16	1.34	3.63	0.67	CRIT
17	1.94	1.19	1.30	3.57	0.65	CRIT
18	1.88	1.25	1.26	3.52	0.63	CRIT
19	1.82	1.31	1.31	3.16	0.51	NORM
20	1.76	1.39	1.39	2.69	0.37	NORM
21	1.70	1.48	1.48	2.12	0.23	NORM
22	1.65	1.58	1.58	1.25	0.08	NORM

If these data are considered, it gets clear that the change from normal to critical flow has been calculated correctly. Some of the data is verified:

6. Levels: inside 2.72 m, inside 1.87 m above crest
   2/3 * Outside level: 1.81 m above crest

   Hence, flow is normal *(inside level > 2/3 outside level).*
   Current velocity $\sqrt{2gh} = 4.06$ *m/s*

7. Levels: inside 2.64 m, inside 1.72 m above crest
   2/3 * Outside level: 1.76 m above crest

   Hence, flow is critical *(inside level < 2/3 outside level).*
   Current velocity $\sqrt{2gh} = 4.16$ *m/s*

Also for the other data the border between critical and normal flow has been programmed correctly, as shown by the data.

4. Verification of the calculation of dam advancement

For the calculation of the dam advancement, the quantity of dumped material has been divided by a specific width of the trapezoidal slide. The geometrical estimations of this approach can be verified, by comparing the total volume after construction with the real volume of the constructed dam, which is known by:

$$V_{dam} = (B_t + H_d \tan \beta)\ H_d\ L$$

in which V_{dam} the volume of the completed dam, B_t the top width and H_d the height of the dam; β is the dam slope, and L the length of the completed structure (the initial width of the gap). Comparison of this value with the total volume of dumped material, proved that the method used provides for a quite correct description. For various dam dimensions and construction methods this verification has been shown in appendix *D*. In order to verify the calculation of the dam volume, the volume of the completed dam is considered, which can be calculated before the construction starts because the final dimensions of the dam are known.

In *Appendix B* has been explained how in the program the dam is built by dumping layers. The sum of these layers, at the end of the calculations, should equal the calculated dam volume which has been calculated on forehand. For various closures the *calculated dam volume* has been compared to the *real dam volume*. The results of these calculations have been printed below:

Closure Case	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Way of Construction	HOR LEFT	HOR RGT	HOR BTH	VER VESS	VER CBL	HOR LEFT	HOR RGT	HOR BTH	VER VESS	VER CBL
Production Left	2000		1000			800		1100		
Production Right		1500	1000				4000	800		
Production Vert.				500	1000				1000	1300
Dam Top Level	5	3	4	-2.5	3	1	2	3	-3	5
Dam Top Width	10	15	20	8	2	20	15	8	4	6
Material Size	1000	700	600	800	500	10	20	25	400	500
Gap Width	300	400	500	560	1000	1000	200	800	200	300
Gap Depth	13.6	10.2	8.16	13.6	4.08	4.08	20.4	5.1	20.4	20.4
Slope of Dam	0.52	0.5	0.49	0.32	0.49	0.22	0.1	0.28	0.27	0.49
Real Volume	238094	207029	259039	138950	108629	215775	450361	236343	252243	229099
Calc. Volume	238333	207500	259167	138889	108056	215932	451462	236295	252222	228246
Volume Difference	0.1%	0.23%	0.04%	0.04%	0.53%	0.07%	0.2%	0.02%	0.01%	0.37%

Of these figures, it can be concluded that the calculation of the dam volume as described in *appendix B* gives a correct description of the dam advancement.

Closure Case	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Way of Construction	HOR LEFT	HOR RGT	HOR BTH	VER VESS	VER CBL	HOR LEFT	HOR RGT	HOR BTH	VER VESS	VER CBL
Production Left	2000		1000			800		1100		
Production Right		1500	1000				4000	800		
Production Vert.				500	1000				1000	1300
Dam Top Level	5	3	4	-2.5	3	1	2	3	-3	5
Dam Top Width	10	15	20	8	2	20	15	8	4	6
Material Size	1000	700	600	800	500	10	20	25	400	500
Gap Width	300	400	500	560	1000	1000	200	800	200	300
Gap Depth	13.6	10.2	8.16	13.6	4.08	4.08	20.4	5.1	20.4	20.4
Slope of Dam	0.52	0.5	0.49	0.32	0.49	0.22	0.1	0.28	0.27	0.49
Real Volume	238094	207029	259039	138950	108629	215775	450361	236343	252243	229099
Calc. Volume	238333	207500	259167	138889	108056	215932	451462	236295	252222	228246
Volume Difference	0.1%	0.23%	0.04%	0.04%	0.53%	0.07%	0.2%	0.02%	0.01%	0.37%

Of these figures, it can be concluded that the calculation of the dam volume as described in *appendix B* gives a correct description of the dam advancement.

Appendix E

Normative Tidal Condition for Calculation of Equilibrium Profile

For the calculation of the equilibrium profile of the closure gap, the tidal prism has to be known. This is the volume of water that, during a tidal period, enters and leaves the basin and thus flows through the gap. During a longer period of time though, the tidal condition fluctuates, due to the different periods of the components of the tidal wave:

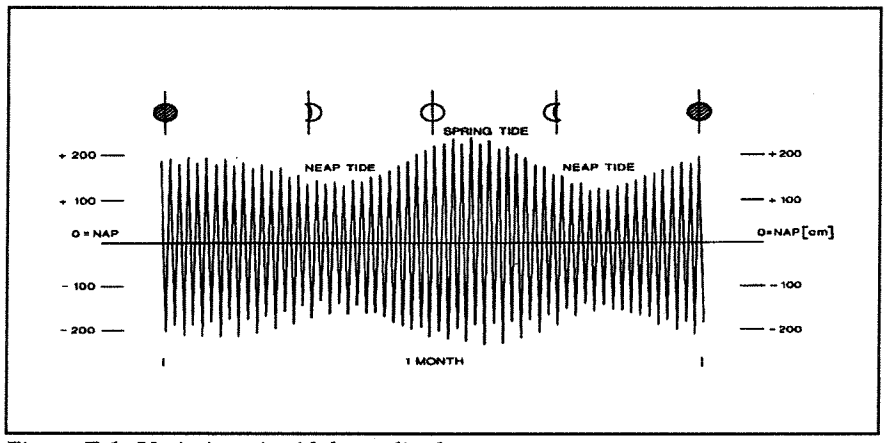


Figure E.1: Variations in tidal amplitude

For the calculation of the tidal prism, a tidal condition should be defined from the four available tidal components, which should, when occurring during a longer period of time, cause the same average transport capacity as the tide which consists of the various components.

If the sum of amplitudes would be chosen (the greatest possible amplitude), this would cause the equilibrium profile to be too big, because this amplitude only happens at spring tide; the same way, the minimum value would be an underestimation.

The *average* tidal amplitude, which would be the average of the tidal amplitudes, should not be used for this calculation. It is obvious that the equilibrium profile depends on the sand transport in the channel. These sand transports depend on the fifth power of the velocity (Engelund & Hansen); to take the average tidal amplitude, would not have the same effect as the average of the velocities that take place in reality.

In channels with enough flow profile, the velocity is linearly dependent with the level fluctuation. Like shown in figure *E.2*, the greatest velocities that will occur at a certain tidal condition, can be known by the gradient in the point of inflection:

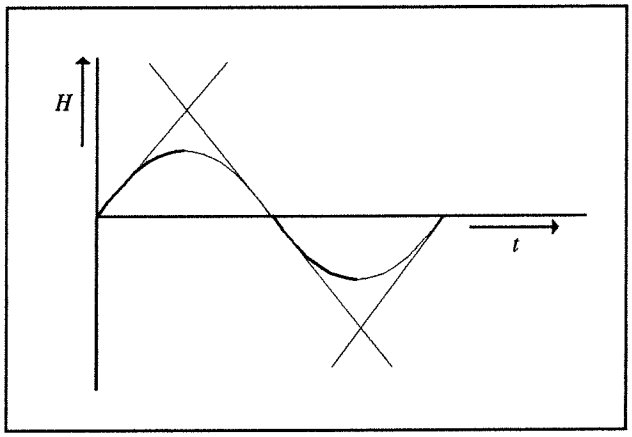


Figure E.2: Velocities caused by tide

The gradient of the tangent can be defined easily. Geometry shows, that the crossing of the tangents happens at a height of 1.5 times the amplitude of the sine. The gradient can be calculated by dividing 1.5 times the amplitude of the sine by half the period of it, and so is obtained:

$$u_{max} = \frac{3A}{0.5\,T}$$

If a combination of two sine components is considered, which have amplitudes A and B, the maximum velocities that will occur will be $1.5(A/T_1+B/T_2)$ (for *spring tide*) and $1.5(A/T_1 - B/T_2)$ (for *neap tide*). If are defined $C = 1.5A/T_1$ and $D = 1.5B/T_2$, the *maximum and minimum transports* that will occur because of these velocities, will be related to $(C+D)^5$ and $(C-D)^5$.

Aim of this all is, to define a tidal condition that will cause the same average transport capacity during a series of tidal periods, as the combination of the various components. The *average transport* that occurs, is conform the average of the fifth power of the velocities, given as:

$$u_{av}^5 = \frac{(C+D)^5 + (C-D)^5}{2}$$

If this is reformulated in C and D, the following equation is obtained for the average contribution to the transport capacity:

$$u_{av}^5 = C^5 + 5\,D^4\,C + 10\,C^3\,D^2$$

If the sum of the amplitudes would be taken as normative for the tidal condition, this value would yield:

$$u_{tot}^5 = (C + D)^5$$

Both relations give a fifth power of the velocity, when two components of the tide with amplitudes A and B are combined. The solution of the latter (u_{tot}) is known to be too high; the fifth root of the first relation (u_{av}) can give an indication of how much too high.

For a series of combinations of amplitudes, which could occur in a tidal system, both resulting maximum velocities have been calculated. In the calculations, the components *M2* and *S2* have been considered having tidal periods of *12.5* hours and *12* hours respectively. In this table, the contribution to u_{tot} (the velocity for the sum of the tidal amplitudes) and u_{av} (averaged to the 5th power of the velocity) have been indicated by N_{SUM} and N_{AVER}. It must be mentioned that some of the combinations of amplitudes will never occur in practice.

AMPLITUDE A1 (OF M2)	AMPLITUDE A2 (OF S2)	A1/T1	A2/T2	N_{SUM}	N_{AVER}	FACTOR
0.4	0.4	0.083	0.08	0.163	0.142	1.149
0.8	0.4	0.083	0.16	0.243	0.212	1.149
1.2	0.4	0.083	0.24	0.323	0.280	1.155
1.6	0.4	0.083	0.32	0.403	0.346	1.165
2.0	0.4	0.083	0.40	0.483	0.410	1.179
0.4	0.8	0.167	0.08	0.247	0.215	1.147
0.8	0.8	0.167	0.16	0.327	0.284	1.149
1.2	0.8	0.167	0.24	0.407	0.354	1.149
1.6	0.8	0.167	0.32	0.487	0.423	1.149
2.0	0.8	0.167	0.40	0.567	0.492	1.151
0.4	1.2	0.250	0.08	0.330	0.289	1.141
0.8	1.2	0.250	0.16	0.410	0.357	1.149
1.2	1.2	0.250	0.24	0.490	0.427	1.149
1.6	1.2	0.250	0.32	0.570	0.496	1.149
2.0	1.2	0.250	0.40	0.650	0.566	1.149
0.4	1.6	0.333	0.08	0.413	0.366	1.130
0.8	1.6	0.333	0.16	0.493	0.430	1.147
1.2	1.6	0.333	0.24	0.573	0.499	1.149
1.6	1.6	0.333	0.32	0.653	0.569	1.149
2.0	1.6	0.333	0.40	0.733	0.638	1.149

0.4	2.0	0.417	0.08	0.497	0.444	1.118
0.8	2.0	0.417	0.16	0.577	0.504	1.145
1.2	2.0	0.417	0.24	0.657	0.572	1.148
1.6	2.0	0.417	0.32	0.737	0.641	1.149
2.0	2.0	0.417	0.40	0.817	0.711	1.149

The factor in the right column of this table, is the quotient of N_{max} and N_{aver}.

In case the periods *T1* and *T2* are the periods of a *diurnial* and *semi-diurnial* tide, like *M2* and *M1*, the same factor is found. This is obvious, because the influence of a longer period on the velocities is equal to the influence of a decrease of the amplitude, so the factor values of the table keep their validity.

From the table it gets clear, that when the *sum of two tidal amplitudes* is applied as determining amplitude during a series of tidal cycles, the velocity is overestimated with 15 percent. Because the velocity is linearly dependant on the amplitude, the calculation should take place applying a reduction on the sum of the two amplitudes; this reduction should be *(1/1.149) = 0.87.*

In the simulation program, four tidal components have been considered. The total reducing factor that should be applied, is given by:

Combination of M2 and S2: use of 0.87 times the sum of amplitudes

Combination of K1 and O1: use of 0.87 times the sum of amplitudes

Combinations of (M2 and S2) and (K1 and O1):
$(0.87)^2$ = 0.76 times the sum of these combinations.

In this way, to obtain the tidal condition that gives a transport equal to that of the *four* components, the sum of the four amplitudes has to be reduced by *$(0.87)^2$ = 0.76.*

Appendix F

Literature

#	*Title*	*Author & Publisher*
1.	*The Closure of Tidal Basins*	J.C. Huis in 't Veld et al Delft University Press, 1987
2.	*Afsluitdammen, Regels voor het Ontwerp*	J.L.M. Konter et al [written in Dutch] Ministerie van Verkeer en Waterstaat, 1992.
3.	*Damming of Tidal Estuaries and Lowland Rivers*	University of Technology, Delft Department of Civil Engineering, april 1994
4.	*Some Considerations on Tidal Inlets*	Ministry of Public Works Section of Tidal Waters, may 1991
5.	*Morphodynamics of Tidal Inlets*	H.D. Niemeyer St. Postak. Onderwijs. CT en BT, Delft, 1990
6.	*Storm Surge Barrier Eastern Scheldt, Evaluation of Watermovement Studies*	H.E. Klatter et al Delft Hydraulics, July 1990.
7.	*Handbook Sediment Transport by Currents and Waves*	L.C. van Rijn Delft Hydraulics, June 1989.
8.	*A Monograph on Sediment Transport in Aluvial Streams*	F. Engelund, E. Hansen Nordic Hydrology 7, 1967
9.	*Anwendung der Ahnlichkeits-Mechanik und der Turbulenz-Forshung auf die Geschiebewegung*	A. Shields Preussische Versuchsanstalt fur Wasserbau und Schiffbau, Berlin, 1936
10.	*Concept of Critical Shear Stress in Loose Boundary Open Channels*	A.S. Paintal Journal of Hydraulic Research, Volume 9, No. 1, june 1970
11.	*Three-dimensional Local*	T. van der Meulen, J.J. Vinje

	Scour in Non-Cohesive Sediments	XVIth congress IAHR, Sao Paulo 1975
12.	*Conformity and Time Scale in two-dimensional local scour*	H.N.C. Breusers Delft Hydraulics Laboratory, March 1965
13.	*Local Scour caused by Vortex Streets*	J.J. Vinje Symposium on Closure of Estuarine Channels in Tidal Regions, De Ingenieur, nr. 47, 1968.
14.	*Ontgrondingen [Local Scour]*	Delft Hydraulics Laboratory [in Dutch Nota II, M1001, june 1981
15.	*Interne nota betreffende optredings- en schadecriteria.....*	P.G.J. Davis [in Dutch] Internal note Deltadienst, RWS, april 1978
16.	*Resultaten van het voor-oeveronderzoek langs de Zeeuwse stromen*	M.H. Wilderom [in Dutch] RWS, Studiedienst Vlissingen, nota nr. 75.2, januari 1979
17.	*Hydraulic Design Criteria for Rockfill Closure of Tidal Gaps*	Delft Hydraulics Laboratory Nota M 1741 part IV, july 1985.
18.	*Construction of dams by Dumping Stones in Running Water*	S.V. Izbash Moscow - Leningrad, 1932
19.	*Computation of Maximum Discharge at Overflow Rockfill Dams*	J. Knauss 13th. Congr. des Grands Barrages, New Delhi, Q.50, R9, 1979
20.	*Oosterscheldt Storm Surge Barrier, connecting breakwater stability*	J. Wouters [in Dutch] Delft Hydraulics Laboratory, Report on Model Investigation, M 1631-I, 1980

CLOSIM v 1.0

A Simulation Program on the Closure of Tidal Basins

Mark N.K. Middag
August 1994

PART II

Program Description

The program Closim v 1.0 has been written
in fulfillment of the requirements for a Master's Degree from the
University of Technology of Delft, The Netherlands
in cooperation with the Dutch Ministry of Public Works

Professor: Prof. Ir. K. d'Angremond

Supervisors:
Ing. G.A. Beaufort
Dr. Ir. J.A. Cser
Ir. F.C. van Roode
Ir. G.J. Schiereck

Preface

This paper is the second out of a series of three, which describe the Simulation Program on the Closure of Tidal Basins *CLOSIM v 1.0*. This simulation program was written in fulfillment of the requirements for a Master's thesis at the Department of Civil Engineering of the Delft University of Technology.

In this volume the simulation program source code has been discussed. The hydraulic aspects of the program have been dealt with in volume *I* of this paper; the reader is assumed to be familiar with these aspects. The third volume is a user's manual for the simulation program.

Most of the program listing printed in the next chapters of this paper has been commented in the margin. Because the calculation procedures of the program are important, these have been described more extensively.

Delft, summer 1994,

Mark Middag.

Contents

1. Introduction

In this paper, the source code of the simulation program *CLOSIM v 1.0* is presented. The structure diagram of this program is given in the next chapter. In chapter *3* the chosen programming languages are introduced briefly.

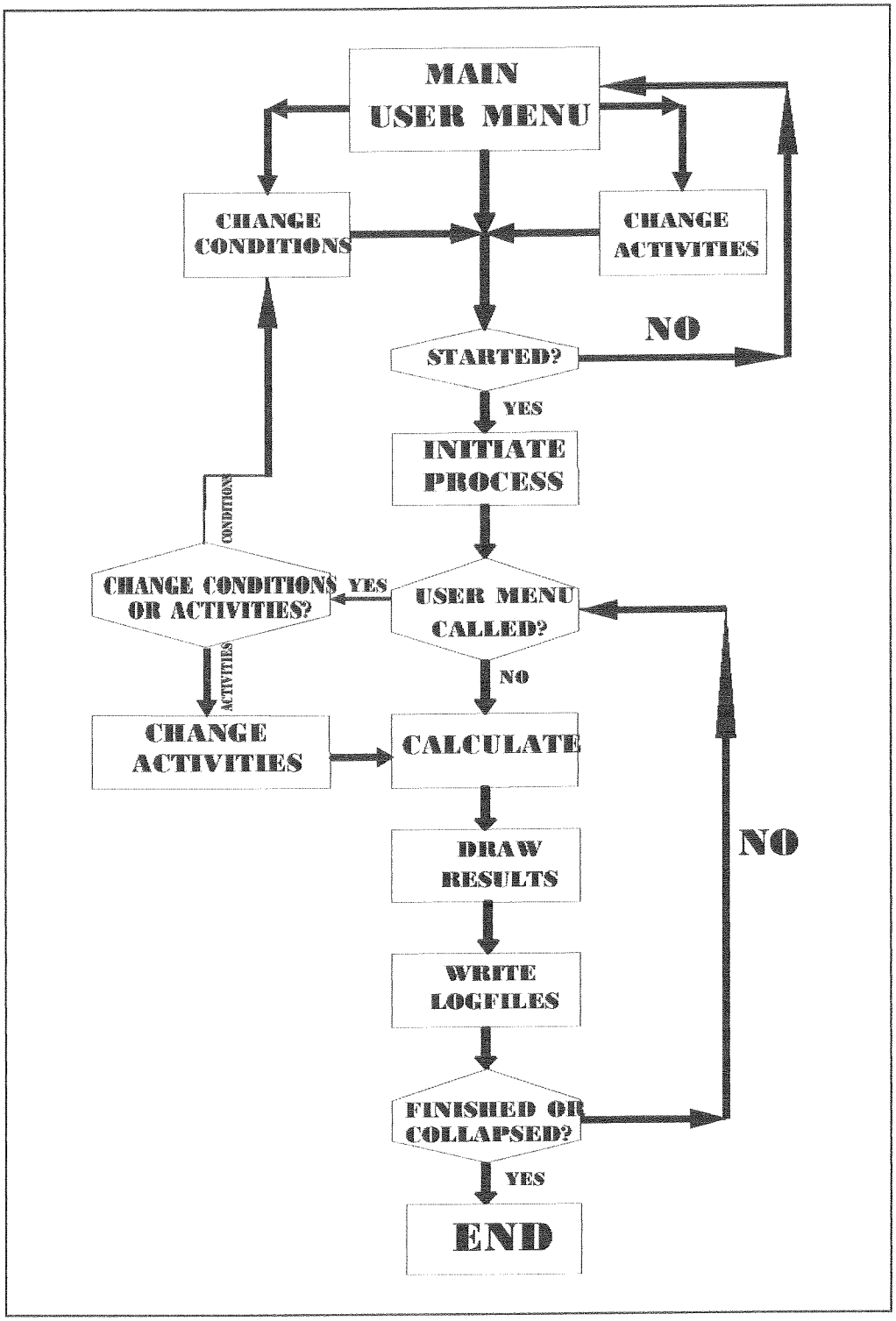
The program exists of various modules, in which the procedures and algoritms are stored. These modules have been commented in chapter *4*. Chapter *5* contains a description of all program variables and constants. Chapter *6* is a description of all source code of the program, in which the calculation procedures have been explained more extensively.

In the last chapter, some information has been given with respect to future adaptation and extension of the program. The programmer is given an instruction on how to adapt the source code, and how to compile the program; a list of required files is given as well.

2. Structure Diagram of the Simulation Program

The structure diagram of the simulation program is quite complex. Thereto, in the figure on the next page the outlines of the program structures have been illustrated; in the specific chapters of this manual, more details about the program structure are discussed.

There must be mentioned, that a number of details of the simulation program do not figure in this structure diagram. Those details will be commented in the description of the procedures in chapter *6*.



In the following, each of the cells of the shown diagram has been explained.

USER MAIN MENU

When the program starts, after an opening screen the main user menu is shown. From this menu, the user can apply all the program options: he can enter data of the closure, start the process, quit the program, or decide which information is shown on the screen or written on the disk.

CHANGE CONDITIONS

The *conditions* are referred to here as all important boundary conditions of the closure. We can, for example, think of the tidal information or the storage function of the basin. All these conditions *define* the tidal closure to be done.

CHANGE ACTIVITIES

The *activities* are those aspects, which can also be entered during the closure. They do not define the closure situation, but consider variable aspects such as for example the *productions of closure material* or the *material grain size.*

For the structure diagram though, also all activities of the program user are mentioned here (e.g. change of colors, decision of which information to store).

STARTED?

As soon as the process is started, the calculations begin.

INITIATE PROCESS

Each time a new closure process is started, the gap situation is set to equilibrium, the calculation variables are reset and the entered values are checked and corrected.

USER MENU CALLED?

Also during the running of the calculation process, the user interface can be applied to change data.

WHAT CHANGED?

When a *condition* has been changed (an important, defining variable for the closure), the process is stopped and the program returns to the main menu.

When just an *activity* is changed, the calculations can go, possibly with the entered values.

CALCULATE

In this procedure, all hydraulic calculations are done. In chapter *6*, an extensive explanation of all calculation procedures has been given.

DRAW RESULTS

After each time step, and for some data after each time slack water occurred at the closure, the results of the calculations are drawn on the screen. The various procedures needed for these drawings, have been explained in chapter *6*.

WRITE LOGFILES

If required, the calculated results are also written to specific files on the disk. The sorts of log files can be adapted in the source code of the program. For now, the program is able to write information about the *production and erosion*, the *velocities*, the *water levels* and the *gap dimensions*.

FINISHED OR COLLAPSED?

Each time step a check is made whether the dam has been finished (the gap has been closed) or the dam has collapsed. In chapter *6*, the calculation procedures for the dam collapse have been described.

If the dam has not been finished or collapsed yet, the calculations continue.

In the following chapters, the implementation of the structure diagram in the program is described.

3. Borland Turbo Pascal and Turbo Vision Graphics

3.1 The Programming Environment

The programming of the simulation program *CLOSIM* has been done in a combination of *Turbo Pascal 7.0* from *Borland Software* and *Turbo Vision Graphics*, written by Richard Andresen. While Turbo Pascal is a structured, highlevel language, in which the simulation program could be written, the Turbo Vision Graphics shell offers an environment in which the user interface of the program can be easily defined.

For the simulation program, object-oriented techniques have been applied. For a definition and explanation of what object-oriented programming is, the reader is referred to Borland's *Turbo Vision Programming Guide*. Although *Turbo Vision Graphics* is more advanced than just Borland's *Turbo Vision* itself, this guide will offer enough insight to understand the structure of the simulation program. In chapter 7 an explanation has been given of how simple aspects can be added to the simulation program, using the features of *Turbo Pascal* and *Turbo Vision Graphics*.

3.2 Turbo Pascal Unit Structure

In *Borland Turbo Pascal 7.0*, written procedures can be put apart in so-called *units*, which are procedure libraries. The Turbo Pascal commander itself also makes use of these units. For example, all procedures which have to do with graphics, are set apart in a library or unit named *GRAPH*. A user who does not use any graphic commands, saves memory because he doesn't use the unit; whenever he wants to use graphics, it is sufficient to call the unit in the first lines of his program, and the procedures in the unit

will be available.

Several separate units for Turbo Pascal, providing procedures for specialized purposes are available and are sold separately from Turbo Pascal itself. Among them units for numeric calculations, advanced graphic routines, and menu-oriented environments like *Turbo Vision*. Other libraries are program specific, and can be written by the programmer, like has been done for this program. The Turbo Pascal *Make* feature combines and compiles all procedure libraries into one executable program.

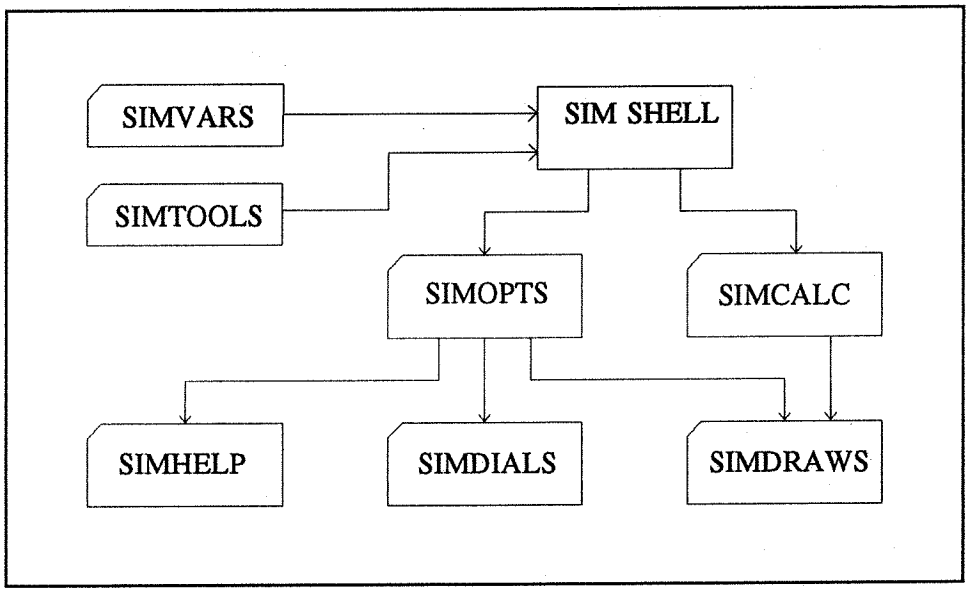
For more specific details on *Turbo Pascal* or *Turbo Vision Graphics* the reader is referred to the programmer's guide, user's guide and references from *Turbo Pascal* and *Turbo Vision*. On the programmer's source disk, the manuals of *Turbo Vision Graphics* are included.

4. Procedure Libraries of the Simulation Program

The simulation program on the Closure of Tidal Basins applies several units. Some of these units *(CRT, DOS, GRAPH)* are included in every standard version of Borland Turbo Pascal 7.0. Some others *(GAPP, GDIALOGS, GBUTTON)* are commercial procedure libraries from *Turbo Vision Graphics* which have been used especially for the user interface of the program.

All units which are indicated by *SIM* in their name *(SIMOPTS, SIMDRAWS, SIMDIALS)* have been written especially for the simulation program. Each of these units, and the procedures which these units contain, will be explained in this manual. First, a structure diagram of the simulation units of the program will be shown.

The division of the simulation program into various specific units, has mainly two functions: the large source code *(listing)* of the program is more easy to deal with; and, the program structure gets clearer which makes programming more flexible. In the figure below, the unit tree of the simulation program is shown:



in which figure the following units:

Sim Shell: This Pascal file is not a unit, but the main program, which contains the menu shell. From this shell, the program points towards all other units for the menu options, the calculation and the drawing procedures.

SimVars: This unit contains all variables, types, records and constants and is used by all other units. This way, if for example a constant is changed, the effect will be noticed through the entire program, without having to change it in all units separately.

SimTools: Various tool procedures, such as the calculation of the power function, the scale calculations for the drawings but also the calculation of the tide at a certain moment, are used by several of the other units. Therefore it is useful to have these put apart in a specific toolbox unit.

SimOpts: The shell program contains the whole set of menu options, which the user can choose. In the unit *SimOpts* the program executes the chosen option or points forward to the procedures which have to do with the option.

SimCalc: All calculations for the program, as been explained before, are executed in this procedure. This way it is easy to change the character of a calculation or formula.

SimDials: Some of the options from the menu require input from the program user.

Because the dialog boxes which are the interface between user and program are complex, but rather similar, they have been put into one unit.

SimHelp: The *help files* for the program are called in this unit.

SimDraws: All drawing procedures (length sections, cross sections, top views) are programmed in this unit

Because of the fact that the program consists of various units, it is also possible to change a complete unit for another, or to add units. This way, in a later stage the program could be extended (by adding new units with procedures), or the environment could be changed (for example, by changing the Turbo Vision shell by another shell)

5. Declaration of Variables and Constants

In Turbo Pascal, all used variables are declared in the beginning of the program or procedure. When this is done in the beginning of the main program, the specific variable is valid through all procedures and units of that program. We call such a variable a *global variable*. If a variable is only valid in a certain procedure, it is declared in the leader of the procedure itself and is called a *local variable*.

As mentioned before, all global variables of the simulation program have been declared in one unit, named *SIMVARS*. Within this unit, the variables have been divided in sections with respect to their function.

The unit *SIMVARS*, in which all variables have been commented, has been shown below.

```
Unit SimVars;                                  {GLOBAL VARIABLES FOR SIMULATION PROGRAM}
                                                      {ON THE CLOSURE OF TIDAL BASINS}
Interface

Uses Dos;                                                {Dos used only for type PATHSTR}

type RealArr = Array[1..5] of Real;          {Reals, for gap and tidal information}
     IntArr  = Array[1..3] of Integer;              {Integers, for gap information}

     RecString  = String[45];                          {String type for dialog input}
     FileString = PathStr;                      {String type for file specification}

const

{== GLOBAL CONSTANTS FOR THE SIMULATION PROGRAM ================================}

    ProgName = 'Closure of Tidal Basins';                              {Program Name}
    Ver = '1.0';                                                    {Program Version}
```

```
    WinNum: Integer    = 0;                                  {Window Counter}
    dpTV1Dialog        = 1000;              {Color pointer for Dialog Boxes}
    hcColorSel         = 1001;                  {Color selector variable 1}
    cmSetColors        = 210;                   {Color selector variable 2}
    cmDosCriticalError = 212;                                {Error command}

{== MENU CONSTANTS FOR TURBO VISION GRAPHICS ==================================}

    cmOverallWindow    = 214;          {Overall graphical information window}
    cmDataBoxWindow    = 216;                {Overall numerical data window}
    cmOptionsSave      = 220;                                {Save, menu option}
    cmOptionsLoad      = 222;                                {Load, menu option}
    cmAbout            = 238;                          {Information on program}
    cmDefault          = 256;              {Load default closure situation}
    cmInitiateProcess  = 114;                  {Initiate closure situation}
    cmStart            = 116;                   {Start calculation process}
    cmInterr           = 118;                {Interrupt calculation process}
    cmProcessToggle    = 120;          {Start or interrupt (toggle) process}
    cmToggleSound      = 122;                      {Toggle sound: ON or OFF}
    cmSelectColors     = 126;                {Color selector, menu option}
    cmDateTime         = 128;                   {Set basin date and time}
    cmBasin            = 130;                {Set basin storage function}
    cmClosure          = 132;              {Set global closure information}
    cmDepthsWidths     = 134;                {Set depths and widths of gaps}
    cmTide             = 140;                          {Set tidal conditions}
    cmRiverData        = 142;                                {Set river data}
    cmDimensions       = 144;                          {Set dam dimensions}
    cmAngles           = 146;                {Set dam side and head angle}
    cmProtection       = 150;                   {Set length of protection}
    cmGradualClose     = 152;              {Set productions and activities}
    cmDefStorm         = 154;                                {Define a storm}
    cmStartStorm       = 156;                                 {Start a storm}
    cmCaissonClose     = 158;              {Caisson closure, now unabled}
    cmCulvert          = 160;                                {Build a culvert}
    cmHelpOnTopics     = 162;                                   {Help option}

{== PERIODS FOR TIDAL COMPONENTS ==============================================}

    PeriodM2 = 44100;          {Period of M2 component of tidal wave, in m}
    PeriodS2 = 43200;          {Period of S2 component of tidal wave, in m}
    PeriodK1 = 22050;          {Period of K1 component of tidal wave, in m}
    PeriodO1 = 21600;          {Period of O1 component of tidal wave, in m}

{== VARIOUS CONSTANTS FOR SIMULATION PROGRAM ==================================}

    Left  = 0;          Horizontal = 1;                  {Answers and directions}
    Right = 1;          Vertical   = 2;
    Both  = 2;          Yes        =-1;
    Up    = 3;          No         = 0;

    OutFlow=-1;                                        {Definition FlowDirections}
    Closed=0;
    InFlow=1;

    Complete=1;                                 {Values for weir and flow situation}
    Critical=-1;
    Opened=1;

    Busy=1;                                          {Values for construction phase}
    Finished=2;
    Interrupted=0;
    Collapsed=3;
```

```
    Good=0;                                    {Qualities of Bottom Protection}
    Bad=1;

    Gravity=9.81;                                                  {Gravity}
    Delta=1.65;                              {delta (mass) for rocky material}

    e=2.7182818;                                                {Natural logarithm}

    AlphaMin=1.5;                                        {Turbulence, minimum value}
    AlphaMaxDepth=4;                  {Maximum turbulence for depth restriction}
    AlphaMaxWidth=6.7;                {Maximum turbulence for width restriction}

    FormFact=20;                                      {Form factor for Scour Holes}

    StaticsOn=False;          {Force Static Situation? (Programmer's use only)}

    ScourRed=True;        {Apply Scour Reduction or not (Programmer's use only)}

    NotBegun      = 0;                                    {Possible Closure Status}
    Start         = 1;
    StartReached  = 2;
    StartToes     = 3;
    Full          = 4;
    Reached       = 5;
    Toes          = 6;

    csTrucks      = 0;                                     {Construction by Trucks}
    csVessels     = 1;                                    {Construction by Vessels}
    csCableWay    = 2;                                   {Construction by Cableway}
    csSandDumping= 3;                                 {Construction by Sand Dumping}

    TimeStep = 1000;                                        {Time step in Seconds}

    DegToPi = 2*Pi/360;                      {Conversion factor, degrees to rads}
    PiToDeg = 1/DegToPi;                     {Conversion factor, rads to degrees}


{== JOB TIME VARIABLES ==========================================================}

var BasinYear,
    BasinMonth,
    BasinDay,
    BasinHour,
    BasinMinute: String;                          {For Job Time Administration}

    JobDayNo: Integer;                                    {Number of day in week}

    DayS: String[12];                                       {Name of day in week}


{== STORAGE SURFACE OF BASIN ====================================================}

    KomDepth,                                {Data Points X:   Level of basin, in m}
    KomSur: RealArr;                        {Data Points Y: Surface of basin, in m2}


{== VARIABLES FOR STORM =========================================================}

    StormSurge,                                 {Maximum possible storm setup, in m}
    ActStormSurge,                                       {Actual storm setup, in m}
    StormPeriod,                                    {Total duration of storm, in h}
    StormChance,                                {Probability of storm occurrance, -}
    TimeToStorm: Real;                           {Time left of storm duration, in s}
```

```
{== TIME VARIABLES FOR CALCULATIONS ==========================================}

    JobHours,                                          {Number of process hours}
    OldJobHours,                                                  {Last value}
    FictiveTime,                                {See manual volume I, chapter 5}
    YearDay,                              {Day of Year for calculation of week day}
    WX1,WY1,WDX,WDY,                                             {Draw variables}
    OWX1,OWY1,OWDX,OWDY: Real;


{== VARIOUS COUNTERS FOR PROGRAM =============================================}

    NumOfGaps,                                       {Total number of closure gaps}
    ONOG,                                             {Last number of closure gaps}
    Counter,                                                  {First global counter}
    Counter2,                                                 {Second global counter}
    SC,                                                       {Width section counter}
    GCT,                                                      {Closure gap counter}
    PSC,                                                   {Scour hole slope counter}
    Control,                                                  {Global control variable}
    DrawCounter,                                              {Counter for drawings}
    NumOfSlackWaters: Integer;                    {Number of equal levels occurred}

    Dummy: Real;                          {Real dummy variable, for programmer's use}
    DumInt: Integer;                   {Integer dummy variable, for programmer's use}

    NumOfSnaps: IntArr;                                       {Number of width sections}


{== GLOBAL VARIABLES ON FLOWS AND LEVELS =====================================}

    FlowDirection,                                        {Direction of flow in channel}
    OldFlowDirection: Integer;                 {Previous direction of flow in channel}

    MaxLevel: Real;                  {The maximum level of sea and basin level, in m}


{== GLOBAL VARIABLES FOR CONSTRUCTION AND EROSION ============================}

    GradualLeft,                                    {Production from Left Side, in m3/h}
    GradualRight,                                  {Production from Right Side, in m3/h}
    GradualUp,                                           {Vertical production, in m3/h}
    UpMaxLevel,                                    {Required top level for sill, in m}
    VerticalInAcc,                               {Inaccuracy for vertical dumping, in m}
    DamTopWidth,                                          {Top Width of dam, in m}
    DamTop,                                       {Required Top Height of dam, in m}
    DamHeight,                                        {Required Heigth of dam, in m}
    ActDamHeight,                                       {Actual Height of dam, in m}
    DSH,                                {Angle for head of horizontal construction, -}
    DSS,                                  {Side angle for section of construction, -}
    DamD50,                                                  {D50 of dam material, in mm}
    InAccVeloFact: RealArr;                        {Velocity factor for inaccuracy, -}

    InAccuracyLevel,                    {Water level according to inaccuracy, in m}
    VeloInAcc,                                {Velocity in inaccuracy, in m3/s}
    InAccMinLevel  : Real;           {Minimum level according to inaccuracy, in m}

    ClosureScheme,                    {Actual activity scheme, horizontal/vertical}
    ClosureDirection,                            {Actual direction of construction}
    OnceCableWay,                                        {Flag for use of cableway}

    YDR,                                      {Actual Level of sill or toes, in m}
    XDRL,                                                {Left toe location, in m}
    XDRR,                                               {Right toe location, in m}
    DamYL,                                                {Left top level, in m}
```

```
    DamYR,                                          {Right top level, in m}
    DumpedLeft,                          {Quantity produced on left side, in m3}
    DumpedRight,                        {Quantity produced on right side, in m3}
    DumpedUp,                             {Quantity produced vertically, in m3}
    ErodedLeft,                          {Quantity eroded from left side, in m3}
    ErodedRight,                        {Quantity eroded from right side, in m3}
    ErodedUp,                                {Quantity eroded from sill, in m3}
    oYDR,                               {Previous level of sill or toes, in m}
    oXDRL,                                  {Previous left toe location, in m}
    oXDRR,                                 {Previous right toe location, in m}
    oDamYL,                                    {Previous left top level, in m}
    oDamYR,                                   {Previous right top level, in m}
    ActualVolume,                               {Actual volume of dam, in m3}
    RequiredVolume,                           {Required volume of dam, in m3}
    TotalDumped,                         {Total dumped quantity till now, in m3}
    TotalEroded,                         {Total eroded quantity till now, in m3}
    ErosionPerMPerS,                      {Actual erosion per width, in m3/m/s}
    TotalErosion,                       {Total actual erosion quantity, in m3/dT}

    SkippedVolumeLeft,                                {Skipped volume at left side}
    SkippedVolumeRight: RealArr;                     {Skipped volume at right side}

    WayOfConstruction,                                      {Construction Method}
    ProcessRight, ProcessLeft: IntArr;                      {Construction Phases}

{== BOTTOM PROTECTION AND SCOUR HOLES ==========================================}

    ProtQuality: IntArr;                                {Quality of bottom protection}

    StandLengthProt,                          {Start length of bottom protection, in m}
    SnapWidth: RealArr;                      {Section width of bottom protection, in m}

    AUUC34,                                                {Breuser's integral for scour}
    LastProtIn,          {Last protecting section of bottom protection, basin side}
    LastProtOut,                          {idem for sea side of bottom protection}
    InHoleDepth,               {Reduced scour hole depth for sections at basin side}
    OutHoleDepth,                                    {Idem for sections at sea side}
    InHoleScour,                                  {Scour for sections at basin side}
    OutHoleScour,                                    {Idem for sections at sea side}
    oInHoleDepth,                                                {Idem, previous value}
    oOutHoleDepth: Array[1..3, 0..40] of Real;                   {Idem, previous value}

    MaxInHole,                          {Maximum scour hole depth for basin side, in m}
    MaxOutHole,                                                {Idem, for sea side, in m}
    MaxInHolePlace,                  {Location of max. scour depth, basin side, in m}
    MaxOutHolePlace: RealArr;                                  {Idem, for sea side, in m}

    U02,                    {Integral for square of velocities, for angles, in m2/s2}
    ReducPerM,                                        {Reduction on scour, in m3/m/dT}
    TimeSinceKent,                    {Time since last time levels were equal, in s}
    Beta,                                {Angle of scour hole at protection side, -}
    CriticVelocity,                  {Critical velocity for Breuser's formula,in m/s}
    Chezy,                                          {Chezy bottom roughness, in √m/s}
    BottomD50,                                       {D50 of bottom material, in mm}
    Velocity0: Real;          {Current velocity at end of bottom protection, in m/s}

    ProtSection: RealArr;                        {Length of protection sections, in m}

    InProtY,                                                {Protection settlements, in m}
    OutProtY: Array[1..3, 0..40, 0..20] of Real;

    LocalProtLength: Real;                            {Local length of protection, in m}

    ShadowLeft, ShadowRight: IntArr;          {Shadow zone length, in # of sections}
```

```
{== GAP INFORMATION ================================================================}

    GapDepthAtMSL,          OrigGapDepthAtMSL,          {Depth, Width and Surface}
    GapWidth,               OrigGapWidth,                {of gaps in m and m2 and}
    GapSurface,             OrigGapSurface,                 {their initial values}

    GapMu,                                        {contraction factor for each gap, -}
    GapFlow,                                          {Flow through each gap, in m3/s}
    LastGapFlow,                                {Previous value of gap flow, in m3/s}
    Velocity,                                     {Current velocity in gap, in m/s}
    CrestMult: RealArr;                      {Crest velocity multiplication factor, -}

    OrigTotalWidth,                               {Initial width sum of all gaps, in m}
    TotalWidth,                                    {Actual width sum of all gaps, in m}
    OrigMaxHeight,                  {Original maximum required heigth of dams, in m}
    TotalFlow,                                        {Sum of all gap flows, in m3/s}
    WidthFact,                                {Relative width factor for drawings, -}
    LevelFactY,                                    {Factor for drawing of levels, -}
    LevelDrX,                                 {X-coordinate for drawing of levels, -}
    SumOtherGapFlows,          {For calculation: sum of other gap flows, in m3/s}
    LastTotalFlow,                   {Previous sum of other gap flows, in m3/s}
    X1: Real;                                    {X-coordinate for drawing of profiles}
    StartDrawX: RealArr;                      {X-coordinates for drawing of profiles}


    ActSeaLevel,                                                {Actual sea level, in m}
    MeanSeaLevel,                                                 {Mean sea level, in m}
    BasinLevel,                                                   {Level in basin, in m}
    OldBasinLevel,                                      {Previous basin level, in m}
    SeaLevel2,                                     {Sea level for next time step, in m}
    OSeal,                                             {Previous drawn sea level, in m}
    OBasinLevel,                                     {Previous drawn basin level, in m}
    TotalGapSurface,                                    {Total of gap surfaces, in m3}
    OrigTotalGapSurface: Real;                        {Initial sum of surfaces, in m3}

    Construction,                                                {Construction situation}
    WeirSituation: IntArr;                                         {Flow critical or not}


{== TIDAL CONDITIONS AND EQUILIBRIUM PROFILE ========================================}

    TideAmp,                                  {Tidal amplitude for each component, in m}
    TidePhase0,                             {Tidal phase at  T=0  for each component, -}
    TidePeriod: RealArr;                        {Tidal period for each component, in s}

    TotalAmplitude,                        {Sum of amplitudes of tidal components, in m}
    TidalPrism,                                  {Tidal prism, volume of water, in m3}
    EquiTotalGapSurface: Real;          {Sum of gap surfaces in equilibrium, in m3}


{== RIVER AND CULVERT FLOW ==========================================================}

    RiverFlow,                                                       {River flow, in m3}
    CulvertSurface,                                               {Culvert surface, in m2}
    CulvertFlow,                                                {Flow through culvert, in m3}


{== DRAFT OF VESSELS ================================================================}

    VesselDraft: Real;                                          {Required depth for vessels}


{== BOOLEAN VARIABLES AND FLAGS =====================================================}
```

```
    Started,                    {Whether calculation processes have been started}
    OStarted,                                   {Previous value of Started}
    SoundOn,                                            {Sound on or not}
    HelpRequired,                                    {Help required or not}
    RememProfLev,                        {Whether profiles have been drawn yet}
    RememContours,                       {Whether contours have been drawn yet}
    JustSlacked,                         {Whether levels just have been equal}
    OverallOpen,                             {Whether overall window is opened}
    DataBoxOpen,                           {Whether data box window is opened}
    SomethingDrawn,                  {Whether something has been drawn lately}
    SomethingWritten,               {Whether something has been written lately}
    ReNewWindow,                         {Whether the window should be renewed}
    ProcessInited,                 {Whether the process has been initiated or not}
    Zipped,                        {Whether collapse through zipping has occurred}
    DataChanged: Boolean;                        {Whether the data has changed}

{== EXTERNAL INPUT AND OUTPUT ==========================================================}

    MouseSum: LongInt;                                  {Sum of Mouse Coordiniates}

    FTL,                                {Name of file to load / last loaded file}
    FTS: FileString;                                        {Name of file to save}

    SnapFile: Text;                                          {Channel for log files}

implementation

end.
```

6. Description of Units, Procedures and Functions

The division of this chapter is the same as the division of units in the program. Each of the mentioned units has been described in a separate paragraph.

6.1 The Main Program Shell CLOSIM

As mentioned before, the program exists in a number of procedure libraries, which are the units to be discussed in this volume. These units are compiled into a *main shell*, from which all procedures are called. This shell also contains the *user interface*, which in this case is programmed in *Turbo Vision Graphics*.

Below, the program file of the main shell has been printed. The listing has been documented with explanations on the different sections of the program.

```
{*********************************************************}
{             CLOSIM v 1.00  --  MAIN PROGRAM SHELL       }
{*********************************************************}
{ Containing the Turbo Vision Graphics User Interface for  }
{ the simulation program on the Closure of Tidal Basins.   }
{ Written in Turbo Pascal 7.0 by Mark Middag, TUDelft, 1994}
{*********************************************************}

Program Sim2;

uses CRT, DOS,                                    {Borland Turbo Pascal 7.0}
     Memory,                                            {Memory Procedures}
     MyGraph3,                                   {Advanced Graphics Library}
     GObjects, GDrivers,                        {Turbo Vision Graphics Units}
     MCursor2, GMENU6,
     GViews,   GDialogs,
     GMsgBox,  GStdDlg,
```

```
     GApp,     GColors,
     GWindow,  GBut,
     BMPDrvr,

     SIMVARS,                                              {Program Variables}
     GSIMOPTS,                                               {Menu Options}
     GSIMDIAL,                                    {Dialog Boxes and User Interface}
     SIMTOOLS,                                          {Various Programming Tools}
     SIMHELP,                                             {On Line Help Procedures}
     SIMCALC,                                  {Calculation for Closure of Tidal Basin}
     SIMDRAWS;                                     {Graphical Presentation of Results}

                                                        {The procedure declarations below}
                                                 {cause compiler to link in Bitmap Buttons}
procedure ICSAVE;   external;                      {The Bitmap Buttons appear on the left}
{$L ICSAVE.OBJ}                                                    {side of the desktop.}
procedure ICLOAD;   external;                    {Bitmap Buttons can be edited using}
{$L ICLOAD.OBJ}                                      {Window's PaintBrush and after,}
procedure ICTOGGLE; external;                          {apply the programs BMP2BMG}
{$L ICTOGGLE.OBJ}                                                  {and BINOBJ.EXE}
procedure ICWINDOW; external;
{$L ICWINDOW.OBJ}
procedure ICDATBOX; external;
{$L ICDATBOX.OBJ}
procedure ICQUIT;   external;
{$L ICQUIT.OBJ}
procedure ICSHELL;  external;
{$L ICSHELL.OBJ}
procedure ICGRAD;   external;
{$L ICGRAD.OBJ}
procedure ICSTORM;  external;
{$L ICSTORM.OBJ}

var
  OldExitProc : Pointer;                                  {Saves Exit Procedure Address}
  Graphic     : boolean;                                {True if Screen is in Graphics Mode}

procedure GExitProc; far;                {Restores screen to Text Mode if Program Halts}
begin
  ExitProc := OldExitProc;                              {Restores Exit Procedure Address}
  CloseGraph;                                             {Shut Down the Graphics System}
end;


function GSystemError(ErrorCode: integer;                          {Error Warning Procedure}
              Drive: byte): Integer; far;

const
  SRetryOrCancel:  string[30] = '~Enter~: Retry  ~Esc~: Cancel';
var
  P: Pointer;
  S: string;
  X,YOff : integer;
  SS : string;
  VPort : ViewPortType;
  SaveText : TextSettingsType;
begin
  P := Pointer(Drive + Ord('A'));
  FormatStr(S, GetCritErrorStr(ErrorCode), P);
  SS := S + '      ' + SRetryOrCancel;
  X := (GetMaxX - (Length(SS))*Charlen) div 2;

  GetViewSettings(VPort);                                    {save current viewport}
  SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);                     {set to full screen}

  GetTextSettings(SaveText);                               {save current font, style}
  SetTextStyle(font8x8,HorizDir,1);
```

```
  YOff := CalcVertTextOffset(MenuBar^.Size.y);

  SetColor(lightcyan);
  SetFillStyle(solidfill, red);                          {drawn over menu bar, so can}
                                              {easily be erased by calling MenuBar^.Draw}
  Bar3d(0, 0, GetMaxX, MenuBar^.Size.y, 0, false);
  WriteCStrXY(X, YOff, SS, white, yellow);
  SetColor(white);
  OutTextXY(Charlen,YOff, Chr($10));
  OutTextXY(GetMaxX-2*Charlen,YOff, Chr($11));

  GSystemError := FarSelectKey;                        {get retry/cancel user input}
  MenuBar^.Draw;                                                {erase error message}

  with SaveText do
    SetTextStyle(Font, Direction, CharSize);
  with VPort do
    SetViewPort(X1, Y1, X2, Y2, Clip);
end;

Type                                                {Type Declaration for various Objects}

  POverall = ^TOverall;                                {Window with Graphic Presentation}
  TOverall = object(TWinBackGround)                           {of Closure of Tidal Basin}
    Count : integer;
    Speed : integer;
    constructor Init(var Bounds: TRect);
    procedure Draw; virtual;
    procedure DrawOverall;
    procedure HandleEvent(var Event : TEvent); virtual;
  end;

  PDataBox = ^TDataBox;                               {Window with Numerical Information}
  TDataBox = object(TWinBackGround)                           {on Closure of Tidal Basin}
    Count : integer;
    Speed : integer;
    constructor Init(var Bounds: TRect);
    procedure Draw; virtual;
    procedure DrawDataBox;
    procedure HandleEvent(var Event : TEvent); virtual;
  end;

  constructor TOverall.Init(var Bounds: TRect);          {Constructor for Graphics}
  begin                                                                    {Window}
     TWinBackGround.Init(Bounds);
     EventMask := evTimerTick;
     EnAbleCommands([CmOverallWindow]);
     VColor := black;                                          {store drawing color}
  end;

  procedure TOverall.Draw;                        {Procedure which Draws the information}
  var   Glob: TRect;                           {information on the closure in the window}
  begin
    GetVPRelCoords(Glob);          {get view's outline in viewport relative coords}
    SetFillStyle(solidfill,VColor);                                {set background color}
    Bar(Glob.A.x,Glob.A.y,Glob.B.x,Glob.B.y);                        {draw background}
    SetStartDraws;

    if NumOfSlackWaters>0 then                                  {If new window is drawn,}
       JustSlacked:=True;                                    {it should be filled with the}
                                                              {information on every aspect}
    LevelDrX:=1000;

    MCur.Hide;                                                    {Hide the Mouse Cursor}
    DrawOverall;                                                      {Draw all aspects}
    MCur.Show;                                                    {Show the Mouse Cursor}
```

```
end;

procedure TOverall.DrawOverall;                          {Draw all aspects of graphical}
var                                                            {information into window}
  Radius : word;
  Glob : TRect;
  Color : integer;
begin
  GetVPRelCoords(Glob);          {get view's outline in viewport relative coords}

  WX1:=Glob.A.X;                                      {Remember coordinates of window}
  WY1:=Glob.A.Y;                                            {for display and scaling}
  WDX:=Size.X;
  WDY:=Size.Y;
  SomethingDrawn:=True;                                    {For Window Open Detection}
  MCur.Hide;                                                      {Hide Mouse Cursor}
  if Started then Overall_Draw(WX1,WY1,WDX,WDY);                        {Draw aspects}
  if JustSlacked and Started then
  begin
     JustSlacked:=False;                                        {If levels just equal,}
     Overall_SlackDraw(WX1,WY1,WDX,WDY);                         {draw rest of aspects}
  end;
  MCur.Show;                                                    {Show the Mouse Cursor}
end;

procedure TOverall.HandleEvent(var Event : TEvent);
begin
    if TopView <> PView(DeskTop^.Owner) then                     {If window is opened,}
    begin                                                         {else exit procedure}
       exit;
    end;

  (*if GetState(sfActive) then*)                             {Remove accolades for only}
    OverallOpen:=True;                                        {drawing in active window}
    if Started=True then DrawOverall;
end;

constructor TDataBox.Init(var Bounds: TRect);                 {Constructor for window}
begin                                                            {with numerical data}
   TWinBackGround.Init(Bounds);
   EventMask := evTimerTick;
   EnAbleCommands([CmDataBoxWindow]);
   VColor := black;                                                {store drawing color}
end;

procedure TDataBox.Draw;
var   Glob: TRect;
begin
  GetVPRelCoords(Glob);          {get view's outline in viewport relative coords}

  SetFillStyle(solidfill,VColor);                                 {set background color}
  Bar(Glob.A.x,Glob.A.y,Glob.B.x,Glob.B.y);                            {draw background}

 {DrawDataBox;}                {Accolades might be removed if window is extended}
end;

procedure TDataBox.DrawDataBox;                          {Fill DataBox with information}
var
  Radius : word;
  Glob : TRect;
  VPort: ViewPortType;
  Color : integer;
  TX, TY, PLX : Integer;
begin
  GetVPRelCoords(Glob);          {get view's outline in viewport relative coords}
  GetViewSettings(VPort);
  SetViewPort(Glob.A.X, Glob.A.Y,                                {Restrict new viewport}
```

```
            T(Min(GetMaxX,Glob.B.X)),
            T(Min(Glob.B.Y+20, GetMaxY-32)), ClipOn);
SetColor(LightGray);
SomethingWritten:=True;                          {For DataBox window opened-check}
begin
   TX:=10;                                                  {Relative X-coordinate}
   TY:=50;                                                  {Relative Y-coordinate}
   TextXY(TX,40,      DayS+', ');
   TextXY(TX+11*8,40, BasinDay+'/'+
          BasinMonth+'/'+BasinYear);                              {Time in basin}
   TextXY(TX,53,     'Job day #: '+
                      I(JobDayNo));                            {Number of job day}
   DataTextXY(TX,66, 'Hs (m):    ',
                      SeaLevel2);                             {Actual Sea level, m}
   DataTextXY(TX,79, 'Qt (m3/s): ',
                      TotalFlow);                           {Sum of Gap Flows, m3/s}
   DataTextXY(TX,92, 'As (m2):   ',
                      TotalGapSurface);                   {Sum of Gap Surfaces, m2}
   DataTextXY(TX,105, 'Setup (m): ',
                      ActStormSurge);                       {Actual Storm Setup, m}
   DataTextXY(TX,118, 'Dur (s):   ',
                      TimeToStorm);                       {Duration left for storm}
   TextXY(TX,108+TY,'%fil');                                    {% of gap filled, %}
   TextXY(TX,121+TY,'Qgap');                      {Flow through individual gap, m3/s}
   TextXY(TX,134+TY,'Vgap');                        {Velocity in individual gap, m/s}
   TextXY(TX,147+TY,'Agap');                                {Flow surface of gap, m2}
   TextXY(TX,160+TY,'Cfac');                           {Crest factor for velocity, -}
   TextXY(TX,173+TY,'Weir');                                    {Flow situation, -}
   TextXY(TX,186+TY,'Htop');                               {Level of dam top, m+MSL}
   TextXY(TX,199+TY,'Bgap');                                      {Width of gap, m}
   TextXY(TX,212+TY,'Hdam');                               {Actual height of dam, m}
   TextXY(TX,225+TY,'ReqV');                            {Required volume for dam, m3}
   TextXY(TX,238+TY,'Vdum');                              {Dumped volume for dam, m3}
   TextXY(TX,251+TY,'Vero');                              {Eroded volume for dam, m3}
   TextXY(TX,264+TY,'Vtot');                               {Actual volume of dam, m3}
   TextXY(TX,277+TY,'VskL');                         {Skipped volume on left side, m3}
   TextXY(TX,290+TY,'VskR');                        {Skipped volume on right side, m3}
   TextXY(TX,303+TY,'Ifac');                      {Factor for dumping irregularity, -}
   if ErosLog  then TextXY(TX   ,316+TY,'E');                   {Log file erosion on}
   if VeloLog  then TextXY(TX+16,316+TY,'V');                   {Log file velocity on}
   if LevelLog then TextXY(TX+32,316+TY,'L');                     {Log file levels on}
   if GapLog   then TextXY(TX+48,316+TY,'G');               {Log files gap diameter on}
   for GCT:=1 to NumOfGaps do
   If Construction[GCT]=Collapsed then                      {If construction collapsed}
   begin
      SetColor(LightRed);
      TextXY(TX,TY+330+GCT*16,'CONSTRUCTION '+I(GCT)+' COLLAPSED');
      SetColor(LightGray);
   end;
   for GCT:=1 to NumOfGaps do                          {Below, the presented data above}
   begin                                                          {is written numerically}
      PLX:=TX-13+GCT*50;
      DataTextXY(PLX,TY+95,'GAP ',GCT);
      if (DamTop[GCT]-OrigGapDepthAtMSL[GCT])*GapWidth[GCT] <> 0 then
         DataTextXY(PLX,TY+108,'',
                    Round(100-100*GapWidth[GCT]*(DamTop[GCT]-
                    GapDepthAtMSL[GCT])/((DamTop[GCT]-
                    OrigGapDepthAtMSL[GCT])*GapWidth[GCT])))
      else
         DataTextXY(PLX,TY+108,'', 0);
      DataTextXY(PLX,TY+121,'',Round(GapFlow[GCT]));
      DataTextXY(PLX,TY+134,'',Round(Velocity[GCT]*10)/10);
      DataTextXY(PLX,TY+147,'',Round(GapSurface[GCT]));
      DataTextXY(PLX,TY+160,'',Round(CrestMult[GCT]*100)/100);
      if WeirSituation[GCT]=Complete   then TextXY(PLX,TY+173,'CRIT');
      if WeirSituation[GCT]=Critical then TextXY(PLX,TY+173,'NORM');
      DataTextXY(PLX,TY+186,'',Round(GapDepthAtMSL[GCT]*100)/100);
```

```
          DataTextXY(PLX,TY+199,'',Round(GapWidth[GCT]*100)/100);
          DataTextXY(PLX,TY+212,'',Round(ActDamHeight[GCT]*100)/100);
          DataTextXY(PLX,TY+225,'',Round(RequiredVolume[GCT]));
          DataTextXY(PLX,TY+238,'',Round(TotalDumped[GCT]));
          DataTextXY(PLX,TY+251,'',Round(TotalEroded[GCT]));
          DataTextXY(PLX,TY+264,'',Round(ActualVolume[GCT]));
          DataTextXY(PLX,TY+277,'',Round(SkippedVolumeLeft[GCT]));
          DataTextXY(PLX,TY+290,'',Round(SkippedVolumeRight[GCT]));
          DataTextXY(PLX,TY+303,'',Round(InAccVeloFact[GCT]*100)/100);
        end;
    end;
    SetViewPort(VPort.X1, VPort.Y1, VPort.X2,          {Restore original (full)}
                            VPort.Y2, ClipOn);                        {ViewPort}
  end;

  procedure TDataBox.HandleEvent(var Event : TEvent);
  begin
      if TopView <> PView(DeskTop^.Owner) then          {Only if window is open}
      begin
         exit;
      end;

         {if GetState(sfActive) then}              {If accolades are removed, only}
      if Started=True then DrawDataBox;              {drawing takes place when the}
      DataBoxOpen:=True;                                        {window is active}

         {end;}
  end;

const

  ROverall: TStreamRec = (                                    {Constant- and Record}
     ObjType: 3001;                                        {declaration for saving of}
     VmtLink: Ofs(TypeOf(TOverall)^);                        {desktop files, Graphics}
     Load:    @TOverall.Load;
     Store:   @TOverall.Store
  );
  RDataBox: TStreamRec = (
     ObjType: 3002;
     VmtLink: Ofs(TypeOf(TDataBox)^);              {Same for numerical information}
     Load:    @TDataBox.Load;
     Store:   @TDataBox.Store
  );

  procedure RegisterLocals;                                    {Register both views}
  begin
    RegisterType(ROverall);
    RegisterType(RDataBox);
  end;

type
  TSimApp = object(TProgram)                              {Declaration of procedures}
    constructor Init;                                        {within the SIM object}
    procedure GetEvent(var Event : TEvent); virtual;
    procedure DosShell;                                                 {exit to DOS}
    procedure HandleEvent(var Event: TEvent); virtual;          {Handling of Events}
    procedure InsertOverallWin;                                {Open graphics window}
    procedure InsertDataBoxWin;                                {Open numerics window}
    procedure InitMenuBar; virtual;                               {Initiate Menu Bar}
    procedure InitMessageBar;                                  {Initiate Message Bar}
    procedure InitStatusLine; virtual;                         {Initiate Status Line}
    procedure InitToolBar;                                {Initiate Bitmap Button Bar}
    procedure PutPicture(FName: PathStr; X,Y: Integer);              {Opening Screen}
    procedure SaveDeskTop;                                     {Save Desktop to File}
    procedure LoadDeskTop(FName: String);                     {Load Desktop from File}
    procedure MainProgram;                                {Simulation program, main loop}
    procedure Default(Comment: Word);                    {Load Default Closure Situation}
```

```
    procedure LogFiles;                                    {Apply Log Files}
    procedure Load;                                {Load any Closure Situation}
    procedure Save;                              {Save actual Closure Situation}
    procedure SaveAs;                            {Save Closure Situation as....}
    procedure Quit;                                {Quit program, back to shell}
    procedure InitiateProcess;                     {Set Closure to Equilibrium}
    procedure Start;                                       {Start Calculations}
    procedure Interr(Comment: Boolean);                  {Interrupt Calculations}
    procedure ProcessToggle;                 {Start or Interrupt Calculations}
    procedure ToggleSound;                                 {Toggle Sound on/off}
    procedure SelectColors;                              {Select Color Palette}
    procedure DateTime;                            {Change basin Date and Time}
    procedure Basin;                           {Change basin surface information}
    procedure RiverData;                                   {Change river data}
    procedure Closure;                                     {Change general data}
    procedure DepthsWidths;                          {Change depths and widths}
    procedure Tide;                                  {Change tidal information}
    procedure Dimensions;                              {Change dam dimensions}
    procedure Angles;                            {Change head- and side angle}
    procedure Protection;                  {Length/quality of bottom protection}
    procedure GradualClose;                          {Change production data}
    procedure CaissonClose;                             {Not implemented yet}
    procedure StartStorm;                                  {Start storming}
    procedure DefStorm;                                    {Define a storm}
    procedure Culvert;                           {Build a culvert as spillway}
    procedure HelpOnTopics;                                    {Help option}
    destructor Done; virtual;                                  {End program}
    destructor HaltDone;                   {End program during graphics setup}
  end;

destructor TSimApp.Done;                       {Called for normal program termination}
begin
  TProgram.Done;                                             {Reset program}
  MCur.Done;                                  {releases mouse cursor memory}
  CloseGraph;                                              {Back to text mode}
  Graphic := false;
  DoneSysError;                                          {No error to give back}
  DoneEvents;                                  {No actual events to handle}
  DoneVideo;                                             {Release video memory}
  DoneMemory;                                          {Release reserved memory}
end;

destructor TSimApp.HaltDone;                     {used if program halts while trying}
                                                        {to initilize graphic mode}
begin
  DoneSysError;                                          {No error to give back}
  DoneEvents;                                  {No actual events to handle}
  DoneVideo;                                             {Release video memory}
  DoneMemory;                                          {Release reserved memory}
end;

constructor TSimApp.Init;                                    {Initiation of interface}
  procedure DoStreamRegistration;        {register objects and views for stream I/O}
  begin
    RegisterObjects;
    RegisterViews;
    RegisterDialogs;
    RegisterMenus;
    RegisterApp;
    RegisterStdDlg;
    RegisterWindows;
    RegisterLocals;
    RegisterBitMaps;
  end;
var
  GraphDriver,GraphMode,ErrorCode : integer;
begin
```

```
Graphic := false;
InitMemory;                                                    {Set memory}
InitVideo;                                                      {Set video}
InitEvents;                                                  {Set event set}
InitSysError;                                                   {Set error}
if RegisterBGIdriver(@EGAVGADriverProc) < 0 then             {Detect Initiate}
begin                                                        {graphics driver}
  HaltDone;
  Writeln('Internal EGA/VGA driver not linked.');
  Halt(1);
end;
DetectGraph(GraphDriver, GraphMode);
if not ((GraphDriver = VGA) or (GraphDriver = EGA)) then begin
  HaltDone;
  Writeln('Error - system does not support EGA or VGA graphics.');
  Halt(1);
end;

                {Optional - forces color display mode if in B&W mode. This can}
                          {cause a problem with B&W LCD laptops which can}
                     {drive an external VGA color monitor. They end up in}
                        {color mode and so Turbo Vision selects the color}
                                      {palette instead of B&W palette.}
(*SetVideoMode(smCO80);*)


if GraphDriver = VGA then GraphMode := VGAHi                 {enter graphics mode}
  else GraphMode := EGAHi;
InitGraph(GraphDriver,GraphMode,'');
ErrorCode := GraphResult;
if ErrorCode <> grOK then begin
  HaltDone;
  Writeln('Graphics Error: ',GraphErrorMsg(ErrorCode));
  Halt(1);
end
else begin                                {install exit procedure to Close graphics}
  OldExitProc := ExitProc;                                {save previous exit procedure}
  ExitProc := @GExitProc;                              {insert new exit procedure in chain}
  Graphic := true;
                                    {install graphic mode DOS critical error handler}
  SysErrorFunc := GSystemError;
                                                {improves look of dark gray and brown}
                                                  {on VGA monitors, no effect in EGA}
  ImprovePaletteColors;
end;

TSimApp.PutPicture('TEMPOR.BMP', 90, 100);                      {Load opening screen}
Repeat until KeyPressed or MousePressed;                     {Repeat until key pressed}
Repeat until MousePressed=False;                                    {and key released}

MCur.Init;                                                      {mouse cursor object}
MCur.SetSpeed(12,12);                          {how fast cursor moves, "normal" is 8,8}

TProgram.Init;                                                {Initiate User Interface}

DoStreamRegistration;                                             {Register streams}

InitMessageBar;                                                {Initiate message bar}

DoubleDelay := 6;                    {time between mouse button presses for double press}
                                                {Normal Turbo Vision uses 8 - very slow}

        {set default Viewport to just cover the DeskTop. The MainMenu,MessageBar}
           {and StatusLine temporarily reset viewport when they draw themselves.}

with DeskTop^ do SetViewPort(Origin.x, Origin.y,
         Origin.x + Size.x, Origin.y + Size.y, ClipOn);
```

```
  MCur.SetGrid(1,1,0,0);                                    {set mouse grids to off}
  MouseSnapToMenuGrid := false;
  MouseSnapToDialogGrid := false;

  InitToolBar;                                          {Initiate Bitmap Button Bar}
end;

procedure TSimApp.DosShell;              {Must override method TApplication.DosShell}
begin
                         {USE TurboVision 2.0 MEMORY Unit if compiling with TP 7.0,}
                                                  {use MEMORY ver 1.0 with TP.6.0}
  RestoreCrtMode;                                                 {back to text mode}

  DoneSysError;                                                      {No system error}
  DoneEvents;                                                            {No events}
  DoneVideo;                                                  {Release Video System}

  {$IFDEF VER60}                                                      {Compiler command}
    SetMemTop(HeapPtr);                                    {reduce reserved memory size}
  {$ELSE}                                                  {depending on Pascal version}
    DoneDosMem;
  {$ENDIF}
  Writeln('Type EXIT to return to '+ ProgName + '...');
  SwapVectors;
  Exec(GetEnv('COMSPEC'), '');
  SwapVectors;
      {$IFDEF VER60}
    SetMemTop(HeapEnd);                                          {reserve all of memory}
      {$ELSE}
    InitDosMem;
      {$ENDIF}

  InitVideo;                                             {Back to video, events, errors}
  InitEvents;                                                    {of simulation program}
  InitSysError;

  SetGraphMode(GetGraphMode);                                      {Back to graphics mode}
  ImprovePaletteColors;                                {Improve VGA gray and brown again}

  MCur.RestoreSettings;                    {Restore eventually changed mouse settings}

  Redraw;                                                  {Use Redraw here, not Draw.}
  if DosError <> 0 then
  DOSErrorMessageBox(DosError, 'Running DOS shell');                  {If error occurred}
end;

procedure HandleLogs;                                                    {Adapt Log Files}
begin
      if ErosLog  = True then                                  {Allways first write basin}
         Write(txErosLog, JobHours:2:2);                    {time to each opened log file}
      if VeloLog  = True then
         Write(txVeloLog, JobHours:2:2);
      if LevelLog = True then
      begin
          Write(txLevelLog, JobHours:2:2);                      {Write Sea and Basin Level}
          Write(txLevelLog, '  ',ActSeaLevel:2:2,'  ', BasinLevel:2:2);
      end;
      if GapLog   = True then Write(txGapLog, JobHours:2:2);

      for GCT:=1 to NumOfGaps do
      begin
                                                            {Erosion and Production}
         if ErosLog  = True then Write(txErosLog, '  ',
         TotalErosion[GCT]:2:2,' ',TotalDumped[GCT]:2:2,' ',ActualVolume[GCT]:2:2);
         if VeloLog  = True then Write(txVeloLog, '  ',
         Velocity[GCT]:2:2);                                              {Velocity in gap}
         if GapLog   = True then Write(txGapLog, '  ',
```

```
        GapWidth[GCT]:2:2,'  ', GapDepthAtMSL[GCT]:2:2);          {Gap diameters}
      end;

      if ErosLog  = True then WriteLn(txErosLog);                  {To next line}
      if VeloLog  = True then WriteLn(txVeloLog);
      if LevelLog = True then WriteLn(txLevelLog);
      if GapLog   = True then WriteLn(txGapLog);
end;

Procedure TSimApp.MainProgram;
begin
   EverySecond;                                          {Programmer's use only}
   if (DataChanged=True) and                          {If new information entered}
   (ProcessInited=False) and                           {If not set to equilibrium}
   (Started=True) then                                   {If calculations started}
   begin
       DataChanged:=False;                               {Initiate proces and set}
       RenewWindow:=True;                               {closure situation back to}
       Started:=False;                                {equilibrium; stop calculations}
       Opt_InitProcess;                                  {and load empty desktop}
       TSimApp.LoadDesktop('SIM.DSK');
   end;
   if Started and                                     {If calculations started and}
   MouseStabile then                                      {Mouse not moved then}
   begin
      Calculate;                                        {Calculate new situation}
      HandleLogs;                                              {Adapt Log Files}
   end;
end;

procedure TSimApp.GetEvent(var Event : TEvent);
const
  HelpInUse : boolean = false;
  LastPressDouble : boolean = false;
begin

  TProgram.GetEvent(Event);                             {Usual Call to Get Events}

  TSimApp.MainProgram;                                   {Execute Simulation Loop}
end;

procedure TSimApp.HandleEvent(var Event: TEvent);              {Handling of Events}

  procedure Colors;                                     {Change of Color Palette}
  var
    D: PColorDialog;
  begin
    D := New(PColorDialog, Init('',
      ColorGroup('Desktop',       DesktopColorItems(nil),
      ColorGroup('Menus',         MenuColorItems(nil),
      ColorGroup('Dialogs',  DialogColorItems(dpTV1Dialog, nil),
      ColorGroup('Windows',  WindowColorItems(wpBlueWindow, nil),
        nil))))));
    D^.HelpCtx := hcColorSel;

    if ExecuteDialog(D, Application^.GetPalette) <> cmCancel then
    begin
      ReDraw;                                {Redraw application with new palette}
    end;
  end;

  procedure DosErr;                                {Warning if any disk error occurs}
  var
    F: Text;
    Cmd : integer;
  begin
    Cmd := MessageBox(^C'Testing DOS Critical Error'+
```

```
        ^M^M^C'Remove any disk in drive A:',nil,mfWarning+mfOKCancel);
    if Cmd <> cmOK then exit;

    Assign(F, 'a:\8anyfile.7Z3');                    {Just to check whether the error}
       {$I-}                                                    {has been corrected}
    Reset(F);
    Close(F);
    Cmd := IOResult;
       {$I+}
  end;

  procedure DoAboutBox;                                        {Program Name and Version}
  begin
      MessageBox(^C'   CLOSIM v 1.0'^M^M
                      +^C'A simulation program on'^M
                      +^C'the Closure of Tidal Basins',
      nil, mfInformation+mfOKButton);
  end;

var
  R: TRect;
  PDir,FInputBox : PView;
  Cmd : integer;
begin
  if (ShiftViewPtr <> Nil) then ShiftViewPtr^.HandleEvent(Event);

  TProgram.HandleEvent(Event);                              {usual call to ancestor method}

  if Event.What = evCommand then                             {If a command has been chosen}
  begin
    if HelpRequired then Help_On(Event.Command)            {Help on topic if required}
    else
    case Event.Command of
      cmAbout: DoAboutBox;                                 {Show program Name and Version}
      cmChangeDir:                                                {Change Directory Tree}
        begin
          PDir := New(PChDirDialog, Init(cdNormal       {+ cdHelpButton},0));
          Cmd := DeskTop^.ExecView(PDir);
          Dispose(PDir, Done);
        end;
      cmSetColors: Colors;                                        {Color Palette Selection}
      cmDOSshell : DOSShell;                                             {Goto Dos Shell}
      cmDosCriticalError : DosErr;                                  {If an Error Occurs}
      cmOptionsSave : SaveDeskTop;                                {Save Desktop to File}
      cmOptionsLoad : LoadDeskTop('SIM.DSK');                    {Load Desktop from File}
      cmOverallWindow : InsertOverallWin;                        {Insert Graphics Window}
      cmDataBoxWindow  : InsertDataBoxWin;                       {Insert Numerics Window}
      cmDefault:Default(Yes);                                {Load Default Closure Situation}
      cmLogFiles: LogFiles;                                   {Change Log Files to be Saved}
      cmOpen:Load;                                              {Load any Closure Situation}
      cmSave:Save;                                            {Save Actual Closure Situation}
      cmSaveAs:SaveAs;                                        {Save Closure Situation As....}
      cmQuit:Quit;                                                  {Quit and Back to Shell}
      cmInitiateProcess:InitiateProcess;                          {Reset Closure Situation}
      cmStart:Start;                                                  {Start Calculations}
      cmInterr:Interr(True);                                       {Interrupt Calculations}
      cmProcessToggle: ProcessToggle;                           {Toggle Calculations on/off}
      cmToggleSound: ToggleSound;                                     {Toggle Sound on/off}
      cmSelectColors: SelectColors;                                 {Select Color Palette}
      cmDateTime:DateTime;                                     {Change Basin Date and Time}
      cmBasin:Basin;                                     {Change Basin Storage Information}
      cmRiverData: RiverData;                                          {Change River Data}
      cmClosure: Closure;                                            {Change General Data}
      cmDepthsWidths:DepthsWidths;                             {Change Depths and Widths}
      cmTide:Tide;                                             {Change Tidal Information}
      cmDimensions: Dimensions;                                   {Change Dam Dimensions}
      cmAngles: Angles;                                        {Change Head and Side Angle}
```

```
      cmProtection:Protection;                    {Change Information on Protection}
      cmGradualClose:GradualClose;                          {Change Productions}
      cmCaissonClose:CaissonClose;                        {Not implemented yet}
      cmStartStorm: StartStorm;                                 {Start a Storm}
      cmDefStorm: DefStorm;                                   {Define a Storm}
      cmCulvert: Culvert;                          {Build a Culvert as Spillway}
      cmHelpOnTopics: HelpOnTopics;                               {Help option}
    end;
    if DataChanged=True then                          {If data has been changed,}
    begin                                                  {Process is reset and}
       DataChanged:=False;                            {stopped, empty desktop}
       RenewWindow:=True;                               {is loaded from disk.}
       Started:=False;
       Opt_InitProcess;
       TSimApp.LoadDesktop('SIM.DSK');
    end;
  end;
end;

procedure TSimApp.InsertOverallWin;                          {Insert Graphics Window}
var
  P : PView;
  W : PWindow;
  R : TRect;

begin
if OverallOpen=False then                                       {If not opened yet}
   begin
     SetStartDraws;                                           {Reset draw counters}

     R.Assign(5*Grid-1,1*Grid,40*Grid,43*Grid);               {Reserve Window Space}

     W := New(PSubWindow, Init(R,'Overall View of Closure',wnNoNumber));

     W^.GetMaxSubViewSize(R);                                          {Insert window}
     P := New(POverall, Init(R));
     W^.Insert(P);
     DeskTop^.Insert(W);
   end;
end;

procedure TSimApp.InsertDataBoxWin;                {Insert Numerics Window on Desktop}
var
  P : PView;
  W : PWindow;
  R : TRect;

begin
if DataBoxOpen=False then                                       {If not opened yet}
   begin
     R.Assign(40*Grid+4,1*Grid,62*Grid,43*Grid);              {Reserve window space}

     W := New(PSubWindow, Init(R,'Data Field',wnNoNumber));

     W^.GetMaxSubViewSize(R);                          {Insert Data Window on desktop}
     P := New(PDataBox, Init(R));
     W^.Insert(P);
     DeskTop^.Insert(W);
   end;
end;

procedure TSimApp.InitMessageBar;               {message that covers over the MenuBar}
begin
  MessageBar := New(PGMessageBar,Init);
  Insert(MessageBar);
end;
```

```
Procedure TSimApp.Load;                                    {Load Closure Situation}

var C: Boolean;
begin

  C:=Dial_FileSelect('Closure Situation to Load',          {Select file to be loaded}
                     '*.CTB', 'DEFAULT.CTB', FTL);

  if (FTL<>'') and (C<>False) then                          {If something selected}
  begin
       Opt_Load(FTL);                                         {Load data from file}
       FTS:=FTL;                                            {Save-name is new name}
       TSimApp.Interr(False);                              {Interrupt calculations}
  end;
end;

Procedure TSimApp.SaveAs;                                {Save Actual Closure Situation}

var C: Boolean;

begin
  C:=Dial_FileSelect('Closure Situation to Save',          {Select file to save to}
                     '*.CTB', 'DEFAULT.CTB', FTS);

  if (FTS<>'') and (C<>False) then                          {If something selected}
  begin
       Opt_Save(FTS);                                                 {Save data}
  end;

end;

Procedure TSimApp.Save;                                  {Save Actual Closure Situation}
                                                             {Without choosing new name}
begin
   Opt_Save(FTS);
end;

Procedure TSimApp.LogFiles;                                          {Change log files}
begin
   if ErosLog  = True then Close(txErosLog);             {Close all open log files}
   if VeloLog  = True then Close(txVeloLog);
   if LevelLog = True then Close(txLevelLog);
   if GapLog   = True then Close(txGapLog);

   Dial_LogFiles(ErosLog, VeloLog, LevelLog, GapLog);          {Log file selection}

   if ErosLog  = True then                                    {Open new logfile for}
   begin                                                     {Erosion and Production,}
      Assign(txErosLog, 'EROSPROD.LOG');                                 {if required}
      ReWrite(txErosLog);
      WriteLn(txErosLog, 'Erosion in gap profiles with respect to process time.');
      WriteLn(txErosLog);
   end;
   if VeloLog  = True then                                    {Open new logfile for}
   begin                                                           {gap velocities,}
      Assign(txVeloLog, 'VELOCITY.LOG');                                 {if required}
      ReWrite(txVeloLog);
      WriteLn(txVeloLog, 'Velocity in gap profiles with respect to process time.');
      WriteLn(txVeloLog);
   end;
   if LevelLog  = True then                                   {Open new logfile for}
   begin                                                      {Sea and Basin Level,}
      Assign(txLevelLog, 'LEVELS.LOG');                                  {if required}
      ReWrite(txLevelLog);
      WriteLn(txLevelLog, 'Sea- and basin level with respect to process time.');
      WriteLn(txLevelLog);
   end;
```

```
   if GapLog  = True then                                    {Open new logfile for}
   begin                                                    {gap depths and widths,}
      Assign(txGapLog, 'GAPDIMEN.LOG');                                {if required}
      ReWrite(txGapLog);
      WriteLn(txGapLog, 'Depth and width of gap profiles with respect to process
time.');
      WriteLn(txGapLog);
   end;
end;

Procedure TSimApp.Default(Comment: Word);                          {Load default values}
begin
  MCur.SelectHourGlass;                                          {Change cursor symbol}
  If Comment=Yes then
  MessageBar^.ShowText('Now Loading Default Values');                        {Message bar}
  Opt_Load('DEFAULT.CTB');                                        {Load default data}
  FTS:='DEFAULT.CTB';                                            {Change name to save}
  TSimApp.Interr(False);                                           {Stop calculations}
  if Comment=Yes then
  MessageBar^.Hide;                                                    {hide menu bar}
  MCur.SelectStdCursor;                                          {Select normal cursor}
  DataChanged:=True;                                            {Process will be inited}
end;

Procedure TSimApp.Quit;                                        {Quit and back to DOS shell}
begin
   if ErosLog  = True then Close(txErosLog);                  {Close all opened log files}
   if VeloLog  = True then Close(txVeloLog);
   if LevelLog = True then Close(txLevelLog);
   if GapLog   = True then Close(txGapLog);
   Opt_Quit;
end;

Procedure TSimApp.InitiateProcess;                  {Option for Initiation of Process and}
begin                                                      {Resetting the Closure Gaps to}
   TSimApp.Interr(False);                                                  {Equilibrium}
   DataChanged:=True;
end;

Procedure TSimApp.Start;                                          {Start of calculations}
var Ers: String;
    Perm: Boolean;
begin
   StartPermission(Ers, Perm);                                  {Check for error in data}
   if Perm=False then                                             {if error, don't start}
      if Ers<>'' then
      Control := MessageBox('Impossible to start, due to an'+
          ^M'error in the information with'+
          ^M'respect to the '+Ers+'.',nil,mfWarning + mfOkButton)
      else
      begin Opt_InitProcess; Started:=False; Perm:=True; end;

   if Perm=True then                                               {If no error in data,}
   begin
      EnAbleCommands([cmOverallWindow,                             {Enable menu options}
                      cmDataBoxWindow]);
      MessageBar^.ShowText('Process Has Now Been Started');                  {Message Bar}
      Delay(500);
      Opt_Start;                                                   {Start calculations}
      MessageBar^.Hide;
      if OverallOpen=False then InsertOverallWin;                 {Open graphics window}
      if DataBoxOpen=False then InsertDataBoxWin;                 {Open numerics window}
      Started:=True;
   end
end;

Procedure TSimApp.Interr(Comment: Boolean);                      {Interrupt calculations}
```

```
begin
   Started:=False;                                          {Stop calculations}
   if Comment=True then
   MessageBar^.ShowText('Process Has Now Been Interrupted');
   DisableCommands([                                       {Disable Menu Options}
    cmInterr,
    cmOverallWindow,
    cmDataBoxWindow]);
   if Comment=True then Delay(500);                  {Only message bar if required}
   if Comment=True then MessageBar^.Hide;
end;

procedure TSimApp.ProcessToggle;                        {Toggle calculations on/off}

begin
   if Started=False then TSimApp.Start
                    else TSimApp.Interr(True);
end;

procedure TSimApp.ToggleSound;                             {Toggle sound on/off}
begin
   Opt_ToggleSound;
end;

procedure TSimApp.SelectColors;                          {Select Color Pallette}

var
  D: PColorDialog;
begin
  D := New(PColorDialog, Init('',
    ColorGroup('Desktop',       DesktopColorItems(nil),
    ColorGroup('Menus',         MenuColorItems(nil),
    ColorGroup('Dialogs/Calc',  DialogColorItems(dpGrayDialog, nil),
    ColorGroup('Editor/Puzzle', WindowColorItems(wpBlueWindow, nil),
    ColorGroup('Ascii table',   WindowColorItems(wpGrayWindow, nil),
    ColorGroup('Calendar',
      WindowColorItems(wpCyanWindow,
      ColorItem('Current day',       22, nil)),
      nil))))))));

  if ExecuteDialog(D, Application^.GetPalette) <> cmCancel then
  begin
    DoneMemory;                                        {Dispose all group buffers}
    ReDraw;                                  {Redraw application with new palette}
  end;
end;

procedure TSimApp.DateTime;                          {Change Basin Date and Time}
begin
   Opt_DateTime;
end;

procedure TSimApp.Basin;                              {Change Basin Surface Data}
begin
   Opt_Basin;
end;

procedure TSimApp.RiverData;                          {Change data on river flow}
begin
   Opt_RiverData;
end;

procedure TSimApp.Closure;                  {Change general information on closure}
begin
   Started:=False;
   Opt_Closure;
   DataChanged:=True;
```

```
end;

Procedure TSimApp.DepthsWidths;                    {Change depths and widths of gaps}
begin
   Opt_DepthsWidths;
end;

Procedure TSimApp.Tide;                                  {Change information on the}
begin                                                            {tidal components}
   Opt_Tide;
end;

Procedure TSimApp.Dimensions;                            {Change information on the}
begin                                                              {dam dimensions}
   Opt_Dimensions;
end;

Procedure TSimApp.Angles;
begin
   Opt_Angles;                                          {Change Head and Side Angle}
   CalculateAngles(WayOfConstruction, DSS, DSH);             {Verify Maximum Angles}
end;

Procedure TSimApp.Protection;                        {Change length and quality of}
begin                                                        {the bottom protection}
   Opt_Protection;
end;

Procedure TSimApp.StartStorm;                                          {Start a Storm}
begin
   StormChance:=1;
end;

Procedure TSimApp.DefStorm;                                            {Define a Storm}
begin
   Opt_DefStorm;
end;

procedure TSimApp.CaissonClose;                  {Caisson Closures not implemented}
begin                                                {yet, therefore a warning box}
   Warn_CaissonClosure;
end;

Procedure TSimApp.GradualClose;            {Change of Gradual Closure Productions}
var OldDSS: RealArr;
    SkippedVolume: RealArr;
    DamH: Real;
    AnySkipped:Boolean;

begin
   AnySkipped:=False;
   for GCT:=1 to NumOfGaps do
   if (SkippedVolumeLeft[GCT]>0) or (SkippedVolumeRight[GCT]>0) then
   AnySkipped:=True;
   if AnySkipped then                                {If still material to be skipped,}
   Warn_SkippedNoChange                               {information can not be changed.}
   else
   begin
      Opt_GradualClose;                                                {Ask for productions}
      for GCT:=1 to 3 do OldDSS[GCT]:=DSS[GCT];
      CalculateAngles(WayOfConstruction, DSS, DSH);                         {Verify angles}
      for GCT:=1 to 3 do
         if (DumpedLeft[GCT]<>0) or (DumpedRight[GCT]<>0) then
         if (OldDSS[GCT]<>0) then
         begin
            DamH:=DamTop[GCT]-GapDepthAtMSL[GCT];                  {If angle changed,}
            SkippedVolume[GCT]:=DamTopWidth[GCT]*                 {then skip material}
```

```
            ((0.5*DamH*Tan(DSS[GCT])/Tan(OldDSS[GCT])*
            (DamH/Tan(OldDSS[GCT]))-
            (DamH*DamH/Tan(OldDSS[GCT]))));
          end;
   end;
end;

procedure TSimApp.Culvert;                                    {Construct Spillway}
begin
   Opt_Culvert;
end;

procedure TSimApp.HelpOnTopics;                                    {Set Help flag}
begin
   if HelpRequired=True then HelpRequired:=False
                        else begin HelpRequired:=True; Warn_HelpOn; end;
end;

procedure TSimApp.InitMenuBar;                          {Turbo Vision Graphics Menu Bar}
var
  R: TRect;
begin
  MenuBarHeight :=      {15}23;                    {programmer's choice, Menu Bar
Heigth}
  GetExtent(R);                                                  {All menu items below}
  MenuBar := New(PGMenuBar, Init(R, NewMenu(
    NewSubMenu('~I~nfo ', hcNoContext, NewMenu(
      NewItem('~A~bout Simulator ', '', 0, cmAbout, hcNoContext,
      nil)),
    NewSubMenu('~F~ile', hcNoContext, NewMenu(
      NewItem('~O~pen...', 'F3', kbF3, cmOpen, hcOpen,
      NewItem('~S~ave...', 'F2', kbF2, cmSave, hcSave,
      NewItem('Save ~A~s', 'Alt-F2', kbAltF2, cmSaveAs, hcSaveAs,
      NewLine(
      NewItem('~C~hange dir...', '', kbNoKey, cmChangeDir, hcChangeDir,
      NewItem('~D~OS shell', '', kbNoKey, cmDosShell, hcDosShell,
      NewItem('E~x~it', 'Alt+X', kbAltX, cmQuit, hcExit,
      nil))))))),
    NewSubMenu('~P~rogram ', hcNoContext, NewMenu(
      NewItem('~D~efault reset ', 'F9',kbF9 , cmDefault, hcNoContext,
      NewItem('Init ~P~rocess','' , 0, cmInitiateProcess, hcNoContext,
      NewItem('~S~tart Process', 'F5',kbF5 , cmStart, hcNoContext,
      NewItem('~I~nterrupt Process', 'F6' ,kbF6 , cmInterr, hcNoContext,
      NewLine(
      NewItem('~T~oggle Sound', '' ,0 , cmToggleSound, hcNoContext,
      NewItem('Select ~C~olors', '',0, cmSelectColors, hcNoContext,
      NewItem('~L~og Files', '', 0, cmLogFiles, hcNoContext,
      nil)))))))))),
    NewSubMenu('Desk~T~op', hcNoContext, NewMenu(
      NewItem('~S~ave desktop', '', kbNoKey, cmOptionsSave, hcNoContext,
      NewItem('~L~oad desktop', '', kbNoKey, cmOptionsLoad, hcNoContext,
      nil))),
    NewSubMenu('~W~indows', hcNoContext, NewMenu(
      NewItem('~O~verall View', '', kbNoKey, cmOverallWindow , hcNoContext,
      NewItem('~D~ataBox Window', '', kbNoKey, cmDataBoxWindow, hcNoContext,
      NewItem('~C~lose', 'Alt+F3', kbAltF3, cmClose, hcClose,
      nil)))),
    NewSubMenu('~B~asin ', hcNoContext, NewMenu(
      NewItem('Date & ~T~ime', '', 0 , cmDateTime, hcNoContext,
      NewItem('~B~asin data', '', 0 , cmBasin, hcNoContext,
      NewItem('C~l~osure', '', 0, cmClosure, hcNoContext,
      NewItem('~D~epths and Widths', '', 0 , cmDepthsWidths, hcNoContext,
      NewItem('Tidal ~C~onditions', '', 0 , cmTide, hcNoContext,
      NewItem('~R~iver data','',0,cmRiverData, hcNoContext,
      NewSubMenu('~S~torm ', hcNoContext, NewMenu(
        NewItem('~D~efine Storm', '', 0, cmDefStorm, hcNoContext,
        NewItem('~S~tart Storm', '', 0, cmStartStorm, hcNoContext,
```

```
      nil))),
      nil)))))))),

    NewSubMenu('~D~am ', hcNoContext, NewMenu(
      NewItem('Dam ~d~imensions', '', 0, cmDimensions, hcNoContext,
      NewItem('Dam ~a~ngles', '', 0, cmAngles, hcNoContext,
      nil))),

    NewSubMenu('~S~trategy ', hcNoContext, NewMenu(
      NewItem('~B~ottom Protection', '',0 , cmProtection, hcNoContext,
      NewItem('~G~radual Closure', 'F8', kbF8, cmGradualClose, hcNoContext,
      NewItem('C~a~isson Closure', '',0 , cmCaissonClose, hcNoContext,
      NewItem('~C~onstruct Culvert', '',0, cmCulvert, hcNoContext,
      nil))))),

    NewSubMenu('~H~elp ', hcNoContext, NewMenu(
      NewItem('~H~elp on options', 'F7',kbF7, cmHelpOnTopics, hcNoContext,

    nil)),nil)))))))
  )))));
end;

procedure TSimApp.InitStatusLine;                    {Status line, below on screen}
  function  HiddenStatusKeys(Next : PStatusItem) : PStatusItem;
  begin
    HiddenStatusKeys :=                                  {Status keys which are}
        NewStatusKey('', kbF10, cmMenu,                 {not shown on status line}
        NewStatusKey('', kbAltF3, cmClose,
        NewStatusKey('', kbF8, cmGradualClose,
        Next)));
  end;
var
  R: TRect;
begin
  GetExtent(R);
  R.A.Y := R.B.Y - 9       {Boxheight};             {this gives a 10 pixel tall
StatusLine}
StatusLine :=   New(PGStatusLine, Init(R,
    NewStatusDef(0, $FFFF,                                {Status line commands}
      NewStatusKey('~F1~ Help', kbF1, cmHelp,
      NewStatusKey('~F2~ Save', kbF2, cmSave,
      NewStatusKey('~F3~ Open', kbF3, cmOpen,
      NewStatusKey('~F4~ Save As', kbF4, cmSaveAs,
      NewStatusKey('~F6~ Toggle Process', kbF6, cmProcessToggle,
      NewStatusKey('~Alt-X~ Exit', kbAltX, cmQuit,
      HiddenStatusKeys(nil)))))),
       nil)));

StatusLine^.VFont := font8x8;
                     {use for 10 pixel tall StatusLine - default font is font8x8}
end;

procedure TSimApp.InitToolBar;
                              {See Bitmap documentation of Turbo Vision Graphics.}
const
  BWidth = 32;                                              {Width of Buttons}
  BHeight = 32;                                            {Height of Buttons}
var
  PBar : PToolBar;
  PBut : PIconButton;
  R : TRect;
  NumOfButtons: Integer;
begin
  NumOfButtons:=9;                                          {Number of Buttons}
  R.Assign(0, 0, 5 +BWidth, 2+Boxheight + NumOfButtons *BHeight);
  PBar := New(PToolBar, Init(R, ''));
```

```
                {Buttons have ofSelectable set by default.  If so, the Selected}
            {button will have a dotted line drawn around it. Setting bfGrabFocus}
              {in the Opts field of the constructor will cause a button to Select}
          {itself when clicked with mouse. So the dotted line will be on the last}
                                                                  {clicked button.}
                 {If the dotted line is not wished, clear the ofSelectable flag}
               {in the button's Options field after construction. bfGrabFocus is}
                                          {not needed in this case but doesn't hurt.}
             {Note that buttons will respond to HotKeys, if they have been set.}

R.A.x := 3;  R.A.y := 14;                                          {SAVE BUTTON}
PBut := New(PIconButton, Init(R, '',cmSaveAs,
        bfGrabFocus+tbDrawFrame+tbAutoSize, @ICSAVE));
PBut^.Options := PBut^.Options and not ofSelectable;
PBar^.Insert(PBut);

Inc(R.A.y, BHeight);                                               {LOAD BUTTON}
PBut := New(PIconButton, Init(R, '',cmOpen,
        bfGrabFocus+tbDrawFrame+tbAutoSize, @ICLOAD));
PBut^.Options := PBut^.Options and not ofSelectable;
PBar^.Insert(PBut);

Inc(R.A.y, BHeight);                                             {TOGGLE BUTTON}
PBut := New(PIconButton, Init(R, '',cmProcessToggle,
        bfGrabFocus+tbDrawFrame+tbAutoSize, @ICTOGGLE));
PBut^.Options := PBut^.Options and not ofSelectable;
PBar^.Insert(PBut);

Inc(R.A.y, BHeight);                                      {GRAPHICS WINDOW BUTTON}
PBut := New(PIconButton, Init(R, '',cmOverallWindow,
        bfGrabFocus+tbDrawFrame+tbAutoSize, @ICWINDOW));
PBut^.Options := PBut^.Options and not ofSelectable;
PBar^.Insert(PBut);

Inc(R.A.y, BHeight);                                      {NUMERICS WINDOW BUTTON}
PBut := New(PIconButton, Init(R, '',cmDataBoxWindow,
        bfGrabFocus+tbDrawFrame+tbAutoSize, @ICDATBOX));
PBut^.Options := PBut^.Options and not ofSelectable;
PBar^.Insert(PBut);

Inc(R.A.y, BHeight);                                      {GRADUAL CLOSURE BUTTON}
PBut := New(PIconButton, Init(R, '', cmGradualClose,
        bfGrabFocus+tbDrawFrame+tbAutoSize, @ICGRAD));
PBut^.Options := PBut^.Options and not ofSelectable;
PBar^.Insert(PBut);

Inc(R.A.y, BHeight);                                      {BUTTON FOR STORM START}
PBut := New(PIconButton, Init(R, '~N~',cmStartStorm,
        bfGrabFocus+tbDrawFrame+tbAutoSize, @ICSTORM));
PBut^.Options := PBut^.Options and not ofSelectable;
PBar^.Insert(PBut);

Inc(R.A.y, BHeight);                                      {GO TO DOS SHELL BUTTON}
PBut := New(PIconButton, Init(R, '', cmDosShell,
        bfGrabFocus+tbDrawFrame+tbAutoSize, @ICSHELL));
PBut^.Options := PBut^.Options and not ofSelectable;
PBar^.Insert(PBut);

Inc(R.A.y, BHeight);                                               {QUIT BUTTON}
PBut := New(PIconButton, Init(R, '', cmQuit,
        bfGrabFocus+tbDrawFrame+tbAutoSize, @ICQUIT));
PBut^.Options := PBut^.Options and not ofSelectable;
PBar^.Insert(PBut);

PBar^.SelectNext(false);

DeskTop^.Insert(PBar);
```

```
end;

procedure TSimApp.PutPicture(FName: PathStr; X,Y: Integer);        {Opening Screen}
var
  R : TRect;
  BitPtr : PBitMap;
  Cmd : integer;
  FInputBox : PFileDialog;
  InFile : file;
  Result : word;
  Buf : array[0..Sizeof(TBitMapInfoHeader)-1] of byte;
  TotalBytes : LONGint;      {!!!}
  ErrStr : string;
begin
  BitPtr := nil;
  Inc(WinNum);
  R.A.x := 100; R.A.y := 100;
  if FName <> '' then begin
    Assign(InFile, FName);
    Reset(InFile,1);                                          {reads 1 byte blocks}

                                                          {read just the InfoHeader}
    BlockRead(InFile, Buf, Sizeof(TBitMapInfoHeader), Result);

                          {remember - the Infoheader is in Buf, not yet in BitPtr^.}
    ErrStr := BMPFormatOKStr(PBitMap(@Buf), FName);
    If ErrStr = '' then begin
      BitPtr := AllocateBMPmem(PBitMap(@Buf));      {allocate mem,use special call}
      if BitPtr <> nil then begin
        TotalBytes := GetBitImageSize(PBitMap(@Buf));

        Reset(InFile,1);               {start again at beginning of file, read all}
        BlockRead(InFile, BitPtr^, TotalBytes, Result);

        WinToTVColor(BitPtr);
        MCur.Hide;                                            {Hide Mouse Cursor}
        PutBitMap(X,Y, BitPtr, 0, NormalPut);
        MCur.Show;                                            {Show Mouse Cursor}
      end;
    end
    else
      Cmd := MessageBox(ErrStr, nil, mfError+mfOKButton);

    System.Close(InFile);

    if BitPtr <> nil then FreeMem(BitPtr, TotalBytes);
  end;
end;

procedure TSimApp.SaveDeskTop;                          {Save desktop to file SIM.DSK}
const
  FName = 'SIM.DSK';
var
  SaveFile : TBufStream;
  FStatus,Cmd : integer;
  Pal : PString;
begin
    SaveFile.Init(FName, stCreate, 1048);                    {create a save file}
    Pal := PString(GetPalette);                            {get pointer to palette}
    SaveFile.WriteStr(Pal);                                          {save palette}
    SaveFile.Put(DeskTop);                                           {save DeskTop}
    SaveFile.Flush;
    FStatus := SaveFile.Status;
    SaveFile.Done;                                                 {flushes buffer}
    if FStatus <> stOK then
      if FStatus = stPutError then
        Cmd := MessageBox('Put of unregistered object.',nil,
```

```
        mfError + mfOkButton)
      else if SaveFile.ErrorInfo <> 0 then
           DOSErrorMessageBox(SaveFile.ErrorInfo, FName)
           else
             Cmd := MessageBox('Error saving file.',nil,
             mfError + mfOkButton);
end;

procedure TSimApp.LoadDeskTop(FName: String);              {Load desktop from file}
  procedure CloseView(P: PView); far;
  begin
    Message(P, evCommand, cmClose, nil);
  end;
  procedure ReadFile(var S : TBufStream);
  var
    Pal : PString;
  begin
    if Desktop^.Valid(cmClose) then
    begin
      Pal := S.ReadStr;
      if Pal <> nil then
      begin
        GetPalette^ := Pal^;
        DisposeStr(Pal);
      end;
      Delete(DeskTop);
      Dispose(DeskTop,Done);
      DeskTop := PDeskTop(ValidView(PDeskTop(S.Get)));

      Insert(DeskTop);
    end;
  end;
var
  SaveFile : TBufStream;
  FStatus,Cmd : integer;
begin
  SaveFile.Init(FName, stOpenRead, 1048);
  if (SaveFile.Status = stOK) then begin                          {found file}
    ReadFile(SaveFile);
    FStatus := SaveFile.Status;
    SaveFile.Done;                                              {flushes buffer}
    if FStatus <> stOK then
      if FStatus = stGetError then
        Cmd := MessageBox('Get of unregistered object.',nil,
        mfError + mfOkButton)
      else if SaveFile.ErrorInfo <> 0 then
           DOSErrorMessageBox(SaveFile.ErrorInfo, FName)
           else
             Cmd := MessageBox('Error reading file.',nil,
             mfError + mfOkButton);
  end;
end;

Procedure InitProgram;                          {Initiations for Simulation Program}
begin
  SoundOn       := True;                                    {Sound Flag set to On}
  HelpRequired  := False;                                      {No help required}
  TimeToStorm   := 0;                                        {No storm active}
  ErosLog  := False;                                        {No log files at all}
  VeloLog  := False;
  LevelLog := False;
  GapLog   := False;
end;

var                                                                {Main Program}
  SimApp: TSimApp;
```

```
begin
  SimApp.Init;                          {Initiate Simulation Program User Interface}
  InitProgram;                                    {Initiate Simulation Program}
  SimApp.Default(No);                                   {Load Default Situation}
  SimApp.Run;                                                   {Run Program}
  SimApp.Done;                                        {Shut Down User Interface}
end.
```

6.2 The Calculation Unit SIMCALC

All procedures which regard to the closure calculations in the simulation program, have been written in the unit *SIMCALC.TPU*. The same way as the other units, this unit is called from the beginning of the main program, after which all its procedures can be used. The calculation unit can easily be extended with extra procedures, which could be added in future versions of the program.

In this chapter, all functions and procedures of the calculation unit have been explained. Of each of these subroutines, a brief explanation of tasks and structure is given. The variables which are effected or required by the procedure are listed. After this, a *Program Structure Diagram* clarifies the most important program steps. At last, a commented listing is given.

6.2.1 Procedure Calculate (Main calculation procedure)

As explained earlier (see chapter *2*), the simulation program exists mainly of a *menu loop*, which is gone through constantly. From this menu loop, all other procedures are called: calculations are made, the results are written on the screen, windows are opened and closed, parameters can be changed.

If the *process* has been *started*, the calculation procedure *Calculate* is called each time the program goes through this menu loop. In the main flow chart of the program (chapter *2*), this has been shown. The advantage of calling all calculation procedures from this procedure *Calculate* is, that the numeric processes for each time step are governed from just one place in the program. This makes it possible to control them, to change the location in the program where the calculations take place, but also to be able to extend the procedure with other calculations.

When reading this chapter with regard to the program structure of the calculation loop, the reader is assumed to be acquainted with the information on this subject in the first volume of this description, where each component of the main calculation procedure has been treated separately.

In the *Program Structure Diagram* below, the calculations which are made in this calculation procedure are shown. The variables which are required or adapted are mentioned in the description of each of the specific procedures.

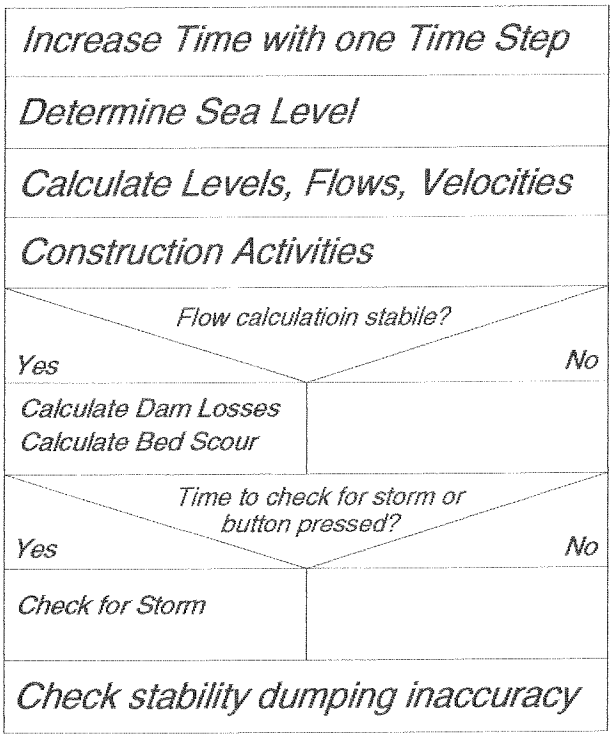
The procedure *Calculate* is called by the procedure:

procedure	*TSimApp.MainProgram*	(Main loop of program shell)

The procedure calls the following functions or procedures:

function	*SeaLevel*	(Calculation of actual sea level)
procedure	*Calcul_Komberg*	(Calculation of levels, flows, velocities)
procedure	*DamDestruction*	(Calculation of dam erosion)
procedure	*BedDestruction*	(Calculation of scour holes)
procedure	*Activities*	(Calculation of construction activities)
procedure	*AdvanceTime*	(Advancing the job time)
procedure	*StormCheck*	(Checking for a storm)
procedure	*ZipCheck*	(Checking the crest stability)

The structure diagram for the procedure *Calculate* is:



The procedures, which are needed to do these calculations will be the main subject of this chapter. As can be seen, before each calculation step the calculation time is increased. After that, for the new process time the sea level is calculated - this occurs in a procedure which has been implemented in the unit *SIMTOOLS*. With the new sea level, the new flows, basin level and the current velocities in each of the gaps are calculated. The closure activities are done (which result in a dam advance, if the dumping exceeds the erosion). In the first (job) hours of the process, the dam losses and bed scour are not calculated, because the hydraulic calculation is not stabile yet. This stabilization time is very short though, and the user of the program will not notice this delay.

Once every day, the probability of a storm to occur is checked. If there is a storm, a counter is started which indicates the time the storm will last. Another check which is made is, whether for vertical dumping an inaccuracy in the construction process (when the dam is lower on one location) could lead to instability. After all these calculations have been made, the program returns to the menu loop.

Below, the Pascal 7.0 listing of this procedure has been given. The comments for each line are written between accolades.

```
Procedure Calculate;                    {MAIN PROCEDURE FOR CALCULATION OF PROCESSES}

var SecEtm: Real;                     {Timestep truncated number of seconds in one day}

begin

   ActSeaLevel:=                              {Calculate actual sealevel with respect}
      SeaLevel(MeanSeaLevel,                 {to the relevant parameters. Function for}
               TideAmp,                            {sea level determination has been}
               TidePeriod,                      {implemented in the unit named SIMTOOLS}
               TidePhase0,
               JobHours);

   Calcul_Komberg;                                {Calculate levels-flows-velocities}

   If JobHours>3 then                            {the first three hours the level-flow}
      if (ONOG=NumOfGaps) then                   {relation needs time to settle. After}
      begin                                        {this, the erosions are calculated}
         DamDestruction;                                    {of the dam material}
         BedDestruction;                                  {and of the bed material}
      end;

   if (ONOG=NumOfGaps)                                                  {error trap}
   then Activities;                                          {execute human activities}

   AdvanceTime(T(TimeStep));                              {advance the time one timestep}

   SecEtm:=Int(3600*24/TimeStep)*TimeStep;
   if ((JobHours*3600/SecEtm)
   =Int(JobHours*3600/SecEtm))                                    {if storm check time}
   or (StormChance=1)                                          {or storm button chosen}
   and (TimeToStorm=0) then                                 {and no storm happening yet}
   begin StormCheck; end;                                {then look whether it is time to storm}


   ZipCheck;                      {check whether the dumping inaccuracy caused collapse}

End;
```

The procedures for the calculation of the closure situation which are - directly or indirectly - called from this procedure, will be explained in the following paragraphs.

6.2.2 Advancing the Process Time

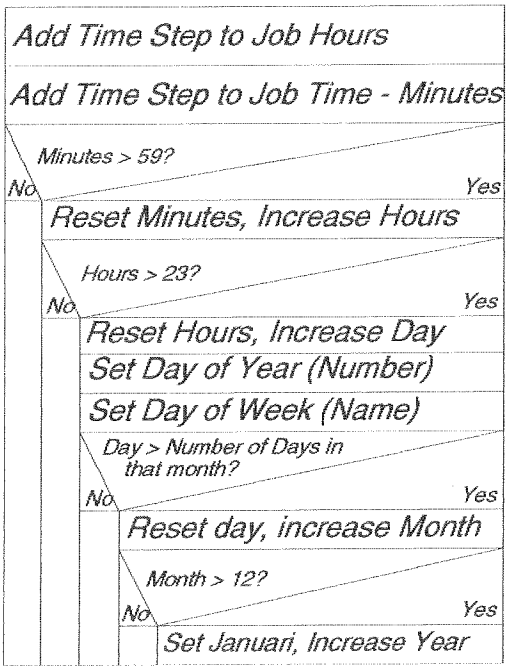
The closure process has its own *basin time and date*, which are useful for the user of the program. For the internal calculation, the number of process hours is calculated, which is important for the calculation of the various time-dependent factors (e.g. *erosion* and *construction activities*).

The procedure *AdvanceTime* requires a *time step*, which is given in seconds. It also calculates with the string variables *BasinYear, BasinMonth, BasinDay, BasinHour* and *BasinMinutes*. The calculations take place with the similar *integer* values which have the names *BYear, BMonth*, etcetera.

The results of the procedure are: *YearDay* (an integer day counter between 1 and 365), *WeekDay* (an integer day counter between 1 and 7); *YobDayNo*, which is a counter for the number of days the construction of the dam is going on; and *DayS*, which is the name of the actual week day (*Monday, Tuesday, etc.*).

The procedure is called by the main procedure *Calculate* and does not call other procedures itself.

In the following *Program Structure Diagram*, the calculations for the job time are shown .



The part of the program in which this structure is included, can be seen in the Pascal listing of the procedure *AdvanceTime* below. Some of the operations are done in strings, because the job time will always be printed to the screen as a string:

```
Procedure AdvanceTime(Seconds: Integer)                    {ADMINISTRATION OF JOB TIME}

var Minutes,                                          {number of minutes of time advance}
    byear,                                                    {integer for basin hour}
    bmonth,                                                 {integer for basin month}
    bday,                                                     {integer for basin day}
    bhour,                                                   {integer for basin hour}
    bminute,                                               {integer for basin minute}
    MaxDays,                                          {maximum number of days in a month}
    code,                                                          {control variable}
    WeekDay: integer;                                    {number of day in week (1-7)}

begin
   Minutes:=T(Seconds/60);
   JobHours:=JobHours+Seconds/3600;                    {CALCULATE number of process hours}
   val(BasinYear,BYear,code);
   val(BasinMonth,BMonth,code);                       {CONVERT time strings into integers}
   val(BasinDay,BDay,code);
   val(BasinHour,BHour,code);
   val(BasinMinute,BMinute,code);
   BMinute:=BMinute+Minutes;                              {ADD number of minutes to time}
   if BMinute>59 then
   begin
      repeat BMinute:=BMinute-60 until BMinute<60;          {ADMINISTRATION of time}
      BHour:=BHour+1;
      if BHour>24 then
      begin
         BHour:=BHour-24;
         BDay:=BDay+1;
         JobDayNo:=JobDayNo+1;
         YearDay:=0;
         for Counter:=1 to BMonth-1 do                       {CALCULATE DAY OF YEAR}
         begin
            if Counter in [1,3,5,7,8,10,12]                 {NUMBER OF DAYS in month}
               then YearDay:=YearDay+31
               else YearDay:=YearDay+30;
            if Counter=2 then
            if BYear/4 = Int(BYear/4)                      {FEBRUARI is special case}
               then YearDay:=YearDay+29
               else Yearday:=Yearday+28;
         end;

         YearDay:=YearDay+BDay;

         WeekDay:=
             BYear-1900 +                                    {CALCULATE day of week}
             Trunc((BYear-1900)/4 + YearDay) + 1;

         WeekDay:=t((WeekDay/7-Trunc(WeekDay/7))*7+1);

         if WeekDay=1 then DayS:='Sunday';                          {ADD NAME OF DAY}
         if WeekDay=2 then DayS:='Monday';
         if WeekDay=3 then DayS:='Tuesday';
         if WeekDay=4 then DayS:='Wednesday';
         if WeekDay=5 then DayS:='Thursday';
         if WeekDay=6 then DayS:='Friday';
         if WeekDay=7 then DayS:='Saturday';
         if BMonth in [1,3,5,7,8,10,12]                  {CALCULATE month length}
```

```
            then MaxDays:=31
            else MaxDays:=30;
         if BMonth=2 then if BYear/4 = Int(BYear/4) then MaxDays:=29
                                                    else MaxDays:=28;
         if BDay>MaxDays then
         begin
            BDay:=1;
            BMonth:=BMonth+1;
            if BMonth>12 then
           begin
               BMonth:=1;
               BYear:=BYear+1;
            end;
         end;
      end;
   end;                                          {CONVERT time back into strings}
   str(BMinute, BasinMinute);
   if length(BasinMinute)=1 then BasinMinute:='0'+BasinMinute;
   str(BHour, BasinHour);

   if length(BasinHour)=1 then BasinHour:='0'+BasinHour;
   str(BDay, BasinDay);
   if length(BasinDay)=1 then BasinDay:='0'+BasinDay;
   str(BMonth, BasinMonth);
   if length(BasinMonth)=1 then BasinMonth:='0'+BasinMonth;
   str(BYear, BasinYear);
end;
```

6.2.3 Calculation of Levels, Flows, and Velocities

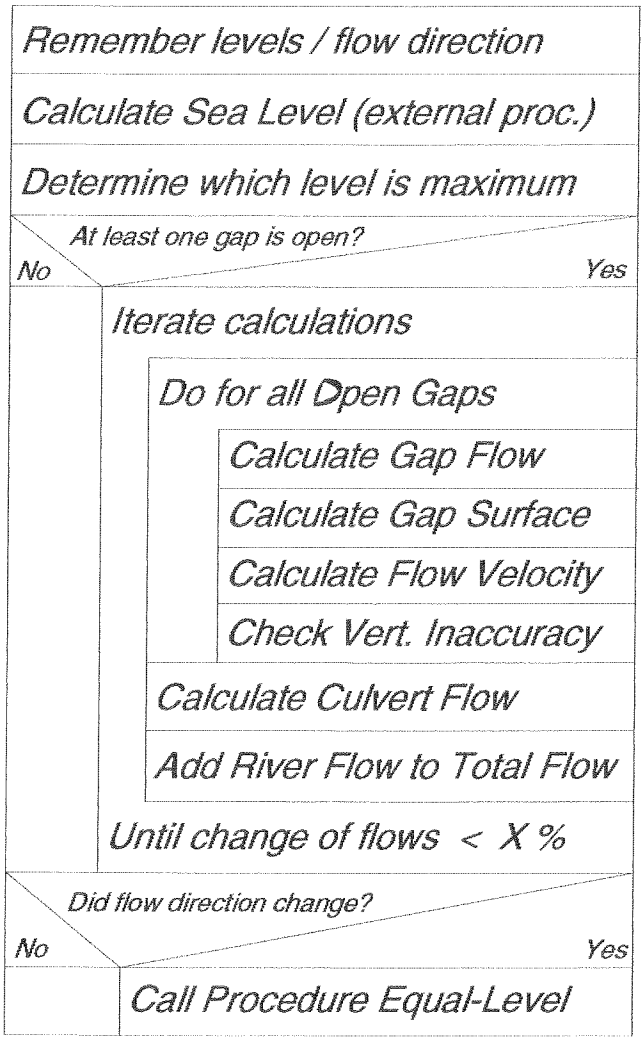
After the sea level has been calculated (with the function *SeaLevel* from the unit *SIMTOOLS*), the storage schematisation which has been explained in volume I of this paper determines the basin level and the gap flows and velocities for the new situation.

For the calculations, the flow and level variables of the last time step are used *(BasinLevel, SeaLevel* and *GapFlow*), as well as the variable *GapSurface*. The procedure calculates the values for the next time step. It also determines whether the variable *WeirSituation* is *critical* or *normal*, if the construction is *finished* or not, and if the variable *FlowDirection* is *InFlow* or *OutFlow*. The variables *GapDepthAtMSL (the depth of the gap at Mean Sea Level)* and *GapWidth* result in a new *GapSurface*.

If needed, the stability of a vertical dumping is verified, using the variable *VerticalInAcc*. If instability occurs, the flag *Zipped* is set *True*.

The procedure is called to by the procedure *Calculate*.When the variable *FlowDirection* changes, the procedure *SlackWater* is called. Other procedures and functions called are *SeaLevel* (for the calculation of the actual sea level), *Power* (to calculate the power relations) and *StorageSurface* (for the calculation of the storage surface of the basin).

Below, the structure of this procedure is given:



The hydraulic backgrounds of this calculation have been explained in chapter 6 of volume I of this paper. Below, the program listing of this procedure is shown:

```
procedure Calcul_Komberg;                {CALCULATION OF LEVELS, FLOWS AND VELOCITIES}

var A, B, C, C2,                        {elements of A-B-C formulas for root calculation}
    D,  D2,                                          {determinants of A-B-C formulas]}
    LD, LD2,                           {level difference and increased level difference}
    MaxLevel,                                        {one of the two levels is higher}
    MinLevel,                                          {one of the two levels is lower
}
    CalcMinLevel,                            {restricted minimum in case of critical flow}
    CalcSeaLevel,                  {average of actual sealevel and that of next timestep}
    OverTop: Real;                                   {height of water above the sill}

begin

    OldBasinLevel:=BasinLevel;                                {remember last basin level}

    SeaLevel2:=SeaLevel(MeanSeaLevel,                  {calculate sealevel with respect}
                        TideAmp,                         {to several actual conditions}
                        TidePeriod,
                        TidePhase0,
                        JobHours*3600+TimeStep);

    CalcSeaLevel:=(ActSeaLevel + SeaLevel2)/2;          {calculation with average of}
                                                       {actual sealevel and sealevel of}
                                                  {next timestep, see volume I chapter 6}

    if OldBasinLevel>=CalcSeaLevel then                    {Look which level is maximum}
       begin
          FlowDirection:=Outflow;                    {in case water flowing out of the
basin}
          MaxLevel:=OldBasinLevel;
          MinLevel:=CalcSeaLevel;
       end
    else
       begin
          FlowDirection:=InFlow;                     {in case water flowing into the
basin}
          MaxLevel:=CalcSeaLevel;
          MinLevel:=OldBasinLevel;
       end;

  repeat                                                          {NUMERIC ITERATION}

    for Counter:=1 to NumOfGaps do
    LastGapFlow[Counter]:=                                 {start value for iteration}
    (LastGapFlow[Counter]+GapFlow[Counter])/2;
    TotalFlow:=0;                                                  {reset variables}
    TotalGapSurface:=0;

    for GCT:=1 to NumOfGaps do

    begin

       GapDepthAtMSL[GCT]:=YDR[GCT];             {depth acording to MSL is sill level}

       InaccuracyLevel:=                                {at one place though, the sill is}
       GapDepthAtMSL[GCT]-                  {lower with respect to dumping inaccuracies}
       VerticalInAcc[GCT];

       if (MinLevel-InAccuracyLevel)<                          {in case of critical flow}
```

```
                                                        {at dumping inaccuricy}
(2/3)*(MaxLevel-InAccuracyLevel) then
   InAccMinLevel:=
   (2/3)*(MaxLevel-InAccuracyLevel)+InAccuracyLevel

else InAccMinLevel:=MinLevel;                          {if flow is not critical}

if (MinLevel-GapDepthAtMSL[GCT])<
(2/3)*(MaxLevel-GapDepthAtMSL[GCT]) then          {in case of critical flow}
                                                   {in the whole closure gap}
begin
  CalcMinLevel:=                                 {then, the level difference}
  (2/3)*(MaxLevel-GapDepthAtMSL[GCT])               {does only depend on the}
  +GapDepthAtMSL[GCT];                              {upstream energy height}
  WeirSituation[GCT]:=Normal;
  OverTop:=CalcMinLevel-GapDepthAtMSL[GCT];
  if OverTop>0 then                                   {in case of high dam,}
  if (OverTop/(Delta*DamD50[GCT]/1000)<4) and        {velocities are higher}
     (OverTop/(Delta*DamD50[GCT]/1000)>0) and
     (2.4-0.35*OverTop/(Delta*DamD50[GCT]/1000)>0)
     then CrestMult[GCT]:=
        SQRT(2.4-0.35*OverTop/
        (Delta*DamD50[GCT]/1000))
     else CrestMult[GCT]:=1
  else CrestMult[GCT]:=1;
end
else                                          {if the flow is not critical,}
begin                                      {the dam is never regarded as high}
  CrestMult[GCT]:=1;
  CalcMinLevel:=MinLevel;
  WeirSituation[GCT]:=InNormal;
end;

if CalcMinLevel-GapDepthAtMSL[GCT]<0 then             {Gap Vertically Closed}
begin
   CalcMinLevel:=GapDepthAtMSL[GCT];
end;

GapSurface[GCT]:=                                    {calculate flow surface}
Abs(CalcMinLevel-                                    {which is width * depth}
GapDepthAtMSL[GCT])*GapWidth[GCT];

if GapSurface[GCT]<0.5 then GapSurface[GCT]:=0;                  {error trap}

If GapSurface[GCT]=0 then
   WeirSituation[GCT] := Closed;

TotalGapSurface:=                      {for calculation of total flow surface}
   TotalGapSurface+GapSurface[GCT];

if WeirSituation[GCT] <> Closed then

begin
                          {=============================================}
                          { see chapter 6 of program description,  volume I}
                          {=============================================}


   A  := 1/Power(GapMu[GCT]*(GapSurface[GCT]), 2);
   B  := Gravity * TimeStep / StorageSurface(OldBasinLevel);
   LD := MaxLevel - CalcMinLevel;
   SumOtherGapFlows:=0;
   if (NumOfSlackWaters>1) then
   for Counter:=1 to NumOfGaps do
      if Counter<>GCT then SumOtherGapFlows:=
          SumOtherGapFlows+LastGapFlow[Counter];
   C  := - gravity * 2 * LD + B * SumOtherGapFlows;
```

```
          D  := Power(abs(B),2) + abs(4 * A * C);
          GapFlow[GCT] := FlowDirection * (-B + SQRT(D)) / (2*A);

          If GapSurface[GCT]<>0
             then Velocity[GCT]:=CrestMult[GCT]*GapFlow[GCT]/GapSurface[GCT]
             else Velocity[GCT]:=0;

          if (ClosureScheme[GCT]=Vertical) and          {for the dumping inaccuracy,}
             (InAccuracyLevel<MinLevel) then            {the calculation is repeated}

          begin
             LD2:= MaxLevel - InAccMinLevel;
             C2 := - gravity * 2 * LD2+ B * (SumOtherGapFlows+RiverFlow);
             D2 := Power(abs(B),2) + Abs(4 * A * C2);
             VeloInAcc      := (Abs(-B + SQRT(D2)) / (2*A))/GapSurface[GCT];
             InAccVeloFact[GCT] := VeloInAcc/SQRT(Delta*Gravity*DamD50[GCT]);
             if (InAccVeloFact[GCT]>1.4) then
             begin Zipped:=True; Construction[GCT]:=Collapsed; end;
          end;
        end
        else
        begin
           GapFlow[GCT]:=0;                    {if the gap is closed, these are zero}
           Velocity[GCT]:=0;
        end;
        TotalFlow:=TotalFlow+GapFlow[GCT];
    end;

    TotalGapSurface:=                                  {a culvert flow is calculated}
    TotalGapSurface+CulvertSurface;

    if CulvertSurface <> 0 then                              {if a culvert is present}

    begin
       A  := 1/Power(1*CulvertSurface, 2);
       B  := Gravity * TimeStep / StorageSurface(OldBasinLevel);
       LD := MaxLevel - MinLevel;
       C  := - gravity * 2 * LD + B * (TotalFlow+RiverFlow);
       D  := Power(abs(B),2) + abs(4 * A * C);
       CulvertFlow := FlowDirection * (-B + SQRT(D)) / (2*A)
    end
    else CulvertFlow:=0;

    TotalFlow:=TotalFlow+CulvertFlow;                 {the total flow is calculated}
  until                {the iteration is repeated till certain exactness is reached}

         (Abs(LastGapFlow[1]-GapFlow[1])<0.01)
     and ((Abs(LastGapFlow[2]-GapFlow[2])<0.01) or (NumOfGaps<2))
     and ((Abs(LastGapFlow[3]-GapFlow[2])<0.01) or (NumOfGaps<3));

  BasinLevel:=OldBasinLevel+                          {the flow causes the basin level}
  TimeStep * (TotalFlow+RiverFlow)/                      {to change. Also a river flow}
  StorageSurface(OldBasinLevel);                                {is taken into account}

  If TotalGapSurface<>0                                                    {error trap}
     then Velocity0:=                                 {calculate velocity at end of}
          TotalFlow/OrigTotalGapSurface                        {the bottom protection}
     else Velocity0:=0;

  if (FlowDirection <> OldFlowDirection) then              {if the flow direction has}
  begin                                  {changed, call the equal-level procedure}
     SlackWater;
     OldFlowDirection:=FlowDirection;
  end;
end;
```

The procedure *SlackWater* (also referred to as *EquiLevel*), is explained this section *6.2.6* of this chapter.

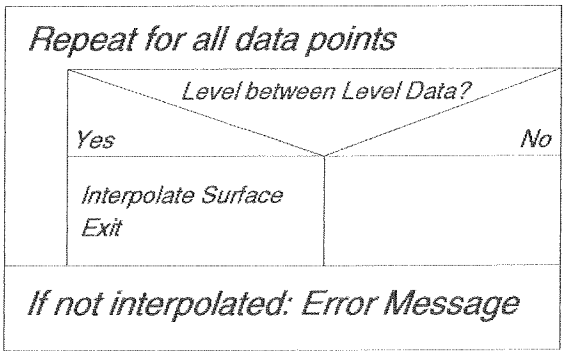
A function is used to determine the storage surface of the basin. In the following paragraph this function has been explained.

6.2.4 Determination of the Storage Surface of the Basin

As explained in chapter *6* of volume I, the storage surface of the basin is not a constant, but varies with the water level in the basin. The user can enter five combinations (*data points*) of a level and a basin surface. The actual level, which is entered in this function, leads via a linear interpolation to a storage surface of the basin. If no interpolation is done, this means that the entered level is not between the data points for the basin level. Then, an error message is given and the user should adapt the basin storage information.

The variables used are the arrays *KomDepth* and *KomSur,* and the actual level which is entered with the header and further named as *Level.* The function returns the calculated value *StorageSurface.* The function is called by the procedure *CalculKomberg,* and calls the warning dialog box *Dial_WarnStorSurf.*

The structure of this interpolation is given in the figure below:



In Pascal, this algorithm reads:

```
Function StorageSurface(Level: Real): Real;              {CALCULATE STORAGE SURFACE}

var ks: Real;                                  {temporal variable for storage surface}
    k: Integer;                                                  {data point counter}

begin
   For k:=1 to 5 do                                                  {for ALL data points}
   if (Level>KomDepth[k-1])
    and (Level<KomDepth[k]) then                          {IF level between data levels}
   begin
      ks:=KomSur[k-1]+
          (Level-KomDepth[k-1])*
          (KomSur[k]-KomSur[k-1])/
          (KomDepth[k]-KomDepth[k-1]);              {INTERPOLATE surface between data}
```

```
    end;
    StorageSurface:=ks;                                {RETURN VALUE for basin surface}
    If StorageSurface=0 then Dial_WarnStorSurf;{error if level not between data points}
end;
```

6.2.5 Scour Holes behind a bottom protection

Behind a bottom protection, scour occurs, as has been explained in chapter 5 of volume I. This scour is calculated according to Breuser's formula. During one tidal period an integral is accumulated, which depends on the velocity and the turbulence. Hereto, the different zones behind the dam (*flow zone, turbulence zone, shadow zone*) are determined; after that, for all locations behind the bottom protection the integral is calculated.

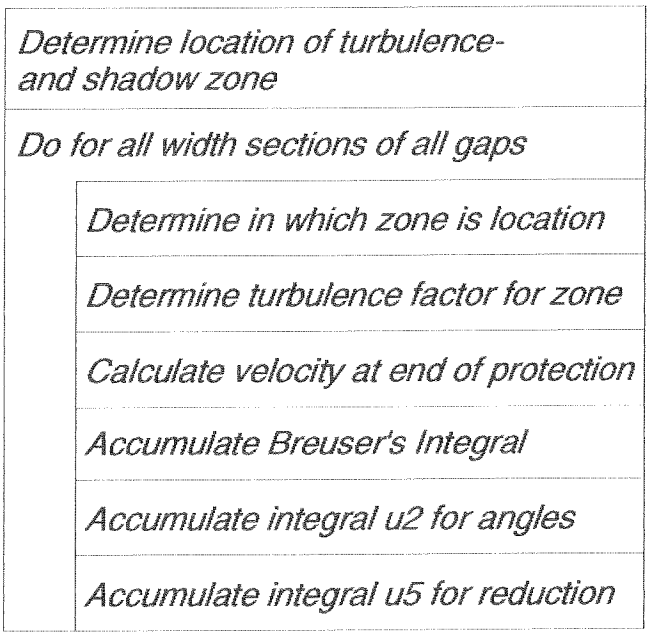
The actual scour is calculated in the procedure *EquiLevel*, which will be treated in the next paragraph. This procedure is called only once in a tidal period, while the accumulation of the integral takes place in every time step. At the same time as Breuser's integral, also integrals for u^2 (for the *scour angle*) and for u^5 (for the *scour reducing transport*) are calculated in this procedure.

The procedure deals with a lot of variables. First, the time since the last time slack water occured, stored in a variable called *TimeSinceKent*, is increased. The critical velocity for the actual bottom material is calculated. The length of the bottom protection (StandLengthProt) , the number of sections of the protection *(NumOfSnaps),* the original total gapsurface *(OrigGapSurface)*, the actual gapsurface *(GapSurface),* the *FlowDirection* and the turbulence for H/L=10 *(Alpha10)* are needed

The most important values which are returned, are: *AUUC34* (the scouring integral of the Breusers formula), *U02* (the square of the velocity, needed for the calculation of the angle of erosion), and *ReducPerM*, the integral for the reduction of the scour holes by the upstream sediment supply.

The procedure is called from the main procedure *Calculate,* and does not call other procedures itself.

Below, the structure of the calculation of the integrals for each location has been shown:



The Pascal listing of this calculation has been shown below. In the program, the procedure which is called from *Calculate* is named *BedDestruction*. This procedure though, does only point to *Calcul_ScouringIntegral,* and has only been written to clarify the program structure.

```
procedure Calcul_ScouringIntegral;   {CALCULATES BREUSER'S INTEGRAL FOR SCOUR HOLES}

var Alpha10,                                   {turbulence factor for Lbp/Hw = 10}
    Alpha,                                  {turbulence factor for actual situation}
    VelocityEndBP,                            {velocity at end of bottom protection}
    VelocityDif,                      {difference between acting and critical velocity}
    RealShadowLeft,                                 {shadow zone  left, in meters}
    RealShadRight,                                  {shadow zone right, in meters}
    SnapPlace,                                {width section counter, in meters}
    RealShearLeft,                            {turbulence zone left,  in meters}
    RealShearRight,                           {turbulence zone right, in meters}
    SnapShearLeft,                      {turbulence zone  left, in width sections}
    SnapShearRight,                     {turbulence zone right, in width sections}
    PSI,                          {Shields parameter to calculate reducing transports}
    M: Real;                             {Constant from sediment transport formula}

begin
   {FOR CALCULATION OF SCOUR HOLE DEPTH}
   TimeSinceKent:=TimeSinceKent+TimeStep;                    {advance time since last}
                                                               {time levels were equal}
   CriticVelocity:=                                        {calculate critical velocity}
       Chezy*SQRT(0.04*Delta*BottomD50/1000);

   for GCT:=1 to NumOfGaps do                                  {for all gaps, if they}
   if  (Construction[GCT]<>Collapsed)                        {are have not collapsed}
   and (Construction[GCT]<>Finished)                             {nor been finished}
```

```
and (StandLengthProt[GCT]<>0) then              {if the bottom has been protected}

begin
  RealShadowLeft:=XDRL[GCT]-(1/7)*StandLengthProt[GCT];          {shadow zone}
  if RealShadowLeft<0 then RealShadowLeft:=0;                  {not left from gap}
  ShadowLeft[GCT]:=T(RealShadowLeft/SnapWidth[GCT]);            {truncated border}
  if ShadowLeft[GCT]=0 then ShadowLeft[GCT]:=-1;

  RealShadRight:=XDRR[GCT]+
                 (1/7)*StandLengthProt[GCT];                 {same for right side}

  if RealShadRight>OrigGapWidth[GCT] then
     RealShadRight:=OrigGapWidth[GCT];
  ShadowRight[GCT]:=T(RealShadRight/SnapWidth[GCT]);
  if ShadowRight[GCT]=NumOfSnaps[GCT] then
     ShadowRight[GCT]:=NumOfSnaps[GCT]+1;

  for SC:=0 to NumOfSnaps[GCT] do                          {for each width section}
  Begin
     SnapPlace:=SC*SnapWidth[GCT];                      {determine location of section}

     Alpha10:=AlphaMin+(                                 {calculate turbulence factor}

     (OrigTotalGapSurface-TotalGapSurface)/OrigTotalGapSurface)
                            *(AlphaMaxDepth-AlphaMin);

     RealShearLeft:=                                                {shear zone border}
                    (XDRL[GCT]+(1/12)*StandLengthProt[GCT]);
     RealShearRight:=(XDRR[GCT]-(1/12)*StandLengthProt[GCT]);

     if XDRL[GCT]=0  then RealShearLeft:=0;                  {cannot be left from gap}
     if XDRR[GCT]=OrigGapWidth[GCT] then RealShearRight:=OrigGapWidth[GCT];

     SnapShearLeft  :=T(RealShearLeft/SnapWidth[GCT]);          {same, but now in}
     SnapShearRight :=T(RealShearRight/SnapWidth[GCT]);          {section widths}

     if (SC<=SnapShearLeft)  and (XDRL[GCT]<>0) or
        (SC>=SnapShearRight) and (XDRR[GCT]<>OrigGapWidth[GCT])

           then Alpha10:=Alpha10+(                        {in turbulence zone, the}
                                                           {turbulence factor is}
     (OrigTotalWidth-TotalWidth)/OrigTotalWidth)                {locally higher}
                            *(AlphaMaxWidth-AlphaMin);

     if FlowDirection=OutFlow              {calculation is done with local length}
                                                          {of the bottom protection}

        then LocalProtLength:=LastProtOut[GCT,SC]*ProtSection[GCT]
        else LocalProtLength:=LastProtIn[GCT,SC]*ProtSection[GCT];

     Alpha:=1.5+(1.57*Alpha10 - 2.35)*                  {calculate alpha with the}
            Power(e,-(0.045*LocalProtLength/          {actual protection and depth}
            abs(OrigGapDepthAtMSL[GCT])));

     if ((RealShadRight-RealShadowLeft)*                  {calculate velocity at}
        (ActSeaLevel-OrigGapDepthAtMSL[GCT]))<>0           {end of the protection}
        then VelocityEndBP:=GapFlow[GCT]/
        ((RealShadRight-RealShadowLeft)*                   {if the gap is closed,}
        (ActSeaLevel-OrigGapDepthAtMSL[GCT]))              {this velocity is zero}
        else VelocityEndBP:=0;

     if (SnapPlace>=RealShadowLeft) and                      {else, this velocity}
        (SnapPlace<=XDRL[GCT]) and                             {is the gap flow}
        (XDRL[GCT]-RealShadowLeft<>0) then                   {divided by the area}
             VelocityEndBP:=VelocityEndBP *               {between the turbulence}
             (SnapPlace-RealShadowLeft)/                                 {zones}
             (XDRL[GCT]-RealShadowLeft);
```

```
        if (SnapPlace>=XDRR[GCT]) and                          {idem for the other}
           (SnapPlace<=RealShadRight) and                        {turbulence zone}
           (XDRR[GCT]-RealShadRight<>0) then
                VelocityEndBP:=VelocityEndBP *
                (SnapPlace-RealShadRight)/
                (XDRR[GCT]-RealShadRight);

        if (SnapPlace<RealShadowLeft) or                {in the shadow zone, the}
        (SnapPlace>RealShadRight) then                 {velocity at the end of the}
        VelocityEndBP:=0;                            {protection is assumed to be zero}

        if GapSurface[GCT]<>0 then
        begin
           VelocityDif:=                                          {calculate the}
              abs(Alpha*VelocityEndBP) - CriticVelocity;         {scouring velocity}
           if (VelocityDif<0) then VelocityDif:=0;              {and accumulate the}
           VelocityDif:=VelocityDif*Sgn(VelocityEndBP);          {scouring integral}
           If StaticsOn then VelocityDif:=(1.4*1.8-0.4);              {according to}
           AUUC34[GCT, SC]:=AUUC34[GCT, SC]+                     {Breuser's formula}
             Power(abs(VelocityDif), 1.7) * TimeStep;
        end;
     end;

     {FOR CALCULATION OF BETA}
     if (Construction[GCT]<>Finished)                {accumulate velocity integral}
     and (Construction[GCT]<>Collapsed) then                    {for calculation of}
        U02:=U02+(VelocityEndBP*VelocityEndBP*TimeStep);              {scour angle}

      {FOR CALCULATION OF TRANSPORT REDUCTION}
      if (Construction[GCT]<>Finished) and            {if scouring still relevant}
      (Construction[GCT]<>Collapsed) then
      begin
         PSI:=Velocity0*Velocity0/                    {calculate transport at end}
         (Chezy*Chezy*Delta*(BottomD50/1000));           {of bottom protection as}
         if PSI>0.06 then                                 {reduction of the scour}
         begin                                          {process bejomd protection}
            M:=0.084/(Power(Chezy,3)*
            SQRT(Gravity)*SQR(Delta)*
            (BottomD50/1000));
            ReducPerM:=ReducPerM+M*Power(Abs(Velocity[GCT]),5);
         end
       end;
   end;
end;

procedure BedDestruction;                          {CALCULATION OF SCOURING HOLES}

begin
                    {As the calculation of scouring holes is only done once a tide,}
                          {the procedure for loss which is continuously called only}
                          {accumulates the scouring integral according to Breusers.}

    Calcul_ScouringIntegral;
end;
```

6.2.6 Calculations when Slack Water occurs

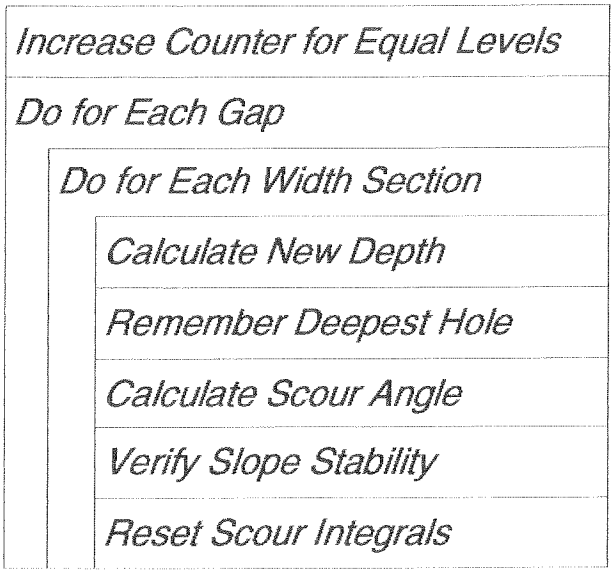
When basin level and sea level are equal (this is called *slack water*), several calculations and drawings are performed. As said in the previous paragraph, the calculation of the scour hole depth is done twice a tidal period, in this procedure. The depths are calculated according to Breuser's formula; the angles are calculated, and a check for stability of the scour hole slopes is made. After this, the integrals (which are accumulated during every time step) are reset to zero.

The variables required are: *TimeSinceKent* (the time since the last slack water occurred), the actual depth *GapDepthAtMSL*, the gap surface *GapSurface*; the integrals *AUUC34* and *U02* are needed, as well as *ReducPerM* and the *FlowDirection*.

All integrals (*TimeSinceKent, AUUC34, U02* and *ReducPerM*) are reset to zero after the calculations. The depths behind the bottom protection *(InHoleDepth, OutHoleDepth)* and the amount of scour *(InHoleScour, OutHoleScour)* are calculated, as well as their extremes *(MaxInHole, MaxOutHole)*. The flag *JustSlacked* is set *True*.

The procedure is called from the main procedure *Calculate* and only calls the procedure *StabilizeScourHoles* to calculate the stability of the scour hole slopes.

Below, the structure diagram for these calculations has been given.


Increase Counter for Equal Levels
Do for Each Gap
Do for Each Width Section
Calculate New Depth
Remember Deepest Hole
Calculate Scour Angle
Verify Slope Stability
Reset Scour Integrals

This structure diagram for the *slack water calculations* is written in Pascal as:

```
procedure SlackWater;                                  {This procedure is called each time}
                                                      {the inside and outside levels are equal}

var P,                                                                  {part of scour formula}
    Scf1,                                                               {part of scour formula}
    Scf2: Real;                                                         {part of scour formula}
    SC  : Integer;                                                      {width section counter}

begin

  NumOfSlackWaters:=NumOfSlackWaters+1;              {number of equal levels is counted}

  if (TimeSinceKent-TimeStep) *                       {none of these should equal zero!}
     GapDepthAtMSL[GCT] *
     GapSurface[GCT] <> 0 then
  if NumOfSlackWaters>1 then                        {not the first time, because then the}
                                                      {flow calculation is not correct yet}
  begin
     for GCT:=1 to NumOfGaps do                                          {for all gaps}
      if  (Construction[GCT]<>Finished)
      and (Construction[GCT]<>Collapsed) then
      begin
      for SC:=0 to NumOfSnaps[GCT] do                          {for all width sections}
       begin
         Scf1:=0;
         Scf2:=0;
         P:=(AUUC34[GCT, SC]/(TimeSinceKent-TimeStep))              {scour formula, see}
            *Power(abs(OrigGapDepthAtMSL[GCT]),0.2)                {volume I, chapter 5}
            /(10*Power(Delta, 0.7));
         if P<>0 then
         begin
            if (FlowDirection=Inflow)                        {if water flowing into basin}
            or (StaticsOn) then                  {StaticsOn is for programmer's use only}
            begin
               FictiveTime{in hours}:=                    {calculate fictive process time}
               Power(InHoleScour[GCT, SC]/               {see scour description in manual}
                     P,1/0.4);
               InHoleScour[GCT, SC]:=
               P*Power((FictiveTime*3600+
               TimeSinceKent-TimeStep)/3600, 0.4);
               Scf1:=SQR(InHoleScour[GCT, SC]) -                          {scour reduction}
                     Abs(ReducPerM)*SnapWidth[GCT]/FormFact;
               if Scf1>=0 then InHoleDepth[GCT, SC]:=Sqrt(SCF1);
               InHoleDepth[GCT, SC]:=InHoleScour[GCT, SC];

                                  {InHoleScour is the unreduced scour depth, while}
                                          {InHoleDepth is the reduced scour depth}

            end
            else
            begin
               FictiveTime{in hours}:=                      {if water flows out of basin}
               Power(OutHoleScour[GCT, SC]/               {the same calculation is done}
                     P,1/0.4);
               OutHoleScour[GCT, SC]:=
               P*Power((FictiveTime*3600+
               TimeSinceKent-TimeStep)/3600, 0.4);
               Scf2:=SQR(OutHoleScour[GCT, SC]) -
                     Abs(ReducPerM)*SnapWidth[GCT]/FormFact;
               if Scf2>=0 then OutHoleDepth[GCT, SC]:=Sqrt(SCF2);
               OutHoleDepth[GCT, SC]:=OutHoleScour[GCT, SC];
```

```
            end;

            if InHoleDepth[GCT, SC]>MaxInHole[GCT] then
            begin
               MaxInHole[GCT]:=                          {remember place and depth of}
                    InHoleDepth[GCT, SC];                        {deepest scour hole}
               MaxInHolePlace[GCT]:=SC;                              {at basin side}
            end;
            if OutHoleDepth[GCT, SC]>MaxOutHole[GCT] then
            begin
               MaxOutHole[GCT]:=                         {remember place and depth of}
                    OutHoleDepth[GCT, SC];                       {deepest scour hole}
               MaxOutHolePlace[GCT]:=SC;                 {at sea side (for drawings)}
            end;
          end;
        end;
      end;
      Calcul_ScoAngle;                    {call procedure for scour angle calculation}
  end;

  TimeSinceKent:=TimeStep;              {reset counter for time since last SlackWater}

  ReducPerM:=0;                        {reset reduction counter and Breuser's integral}
  for GCT:=1 to NumOfGaps do
  for Counter:=0 to NumOfSnaps[GCT] do AUUC34[GCT, Counter]:=0;

  if Beta<>0 then begin                                                  {error trap}
     StabilizeScourHoles;                          {verify stability of new scour holes}
     JustSlacked:=True;                                          {set flag for drawing}
  end;
end;
```

6.2.7 Construction Activities

The construction activities, which are entered in dumped volume from the left side, right side or upside, are converted into a certain dam advance *dX* or *dH* in a rather complex procedure *Activities*, which is the subject of this paragraph. The procedure is called from the main procedure *Calculate*. The only procedure called to, is *MultiBeep* in case the construction has been finished.

In order to keep the structure diagrams managable, the process has been split up in four sections:

* Verification before activities and calculation erosion
* Horizontal construction
* Vertical construction
* Verification after activities

For each of these sections, a *PSD* and a Pascal listing will be given.

Verification before activities and calculation of erosion

Before the construction activities are implemented, three checks are made:

* If a gap is collapsed, all construction activities for that gap are stopped.
* If the required top of the sill has been reached, as indicated by the user of the program, the vertical construction is stopped.
* If the top of the sill (or the location where both toes meet in the case of horizontal construction) is below water level, erosion is calculated.

This part of the procedure requires the following variables:

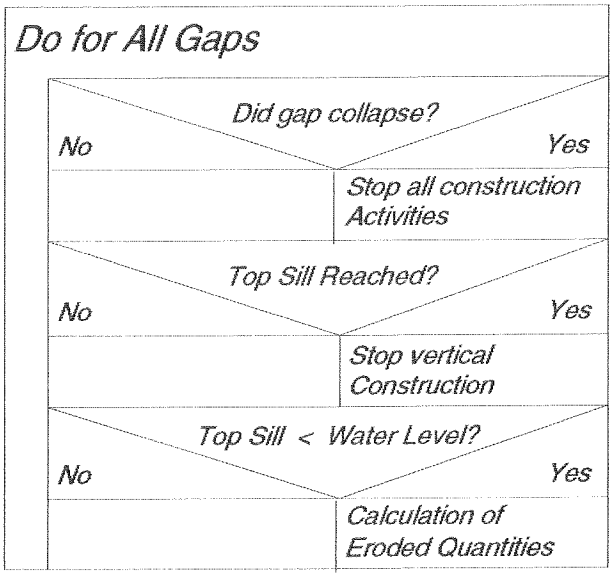
Construction	State of construction (*Finished, Busy* or *Collapsed*)
UpMaxLevel	The maximum sill height
GradualUp	The vertical production
DumpedLeft	The total volume dumped from the left side
DumpedRight	The total volume dumped from the right side

It calculates the following variables:

ErosionLeft	Total erosion at the left slope

ErosionRight	Total erosion at the right slope
ErosionUp	Total erosion at the horizontal sill
TotalErosion	The total erosion at all sides

In the structure diagram below, these checks are shown.



Vertical construction

When a volume has been dumped or eroded vertically, this volume is converted into a dam advance *dH* (which can be negative in case the erosion exceeds the production). The vertical construction can either happen by cableway or by dumping barges. After the advance *dH* has been calculated, a check is made whether the dam has been finished.

In case there has also been horizontal construction, after vertical construction the location of the toes changes. When the toes touched before, after vertical construction they do not.

For the implementation of the structure diagram beneath, the following variables are required:

GradualUp	The production dumped vertically
ErosionUp	The erosion of the sill, calculated above
UpMaxLevel	The maximum height of the sill
WayOfConstruction	The actual way of construction *(csTrucks, csVessels, csCableWay, csSandDump)*

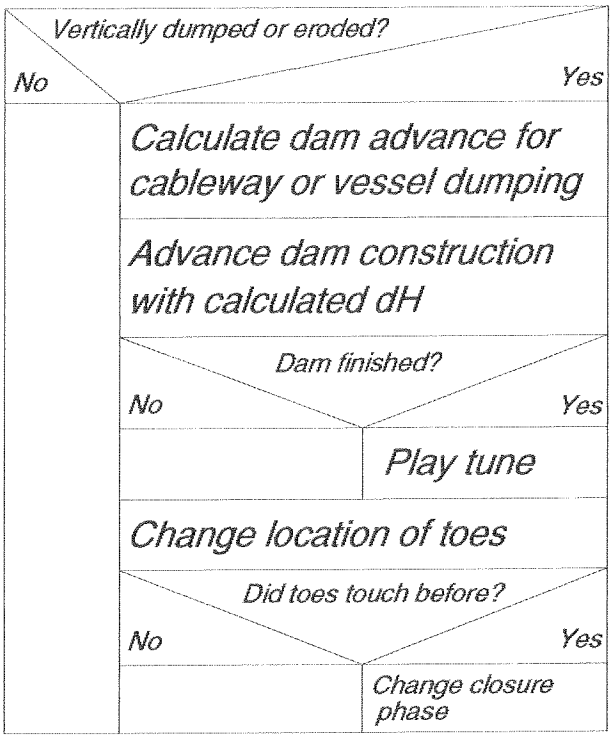
DumpedLeft	The total volume dumped from the left side
DumpedRight	The total volume dumped from the right side
DamTop	The required height of the dam
ProcessLeft	The state of the construction at the left side

The following variables are changed in this procedure:

YDR	The actual construction top level
OnceCableWay	Whether a cable way has been used yet

ProcessRight	The state of the construction calculation at the right side *(Start, StartToes, Full, Toes, Reached)*
ActDamHeight	The actual height of the construction
Construction	The construction phase *(Busy, Finished* or *Collapsed)*
DumpedUp	Total volume dumped vertically
ErodedUp	Total volume eroded from sill top
XDRL, XDRR	X-coordinates of toes from left and right side

Beneath, the structure diagram for the vertical construction has been shown.



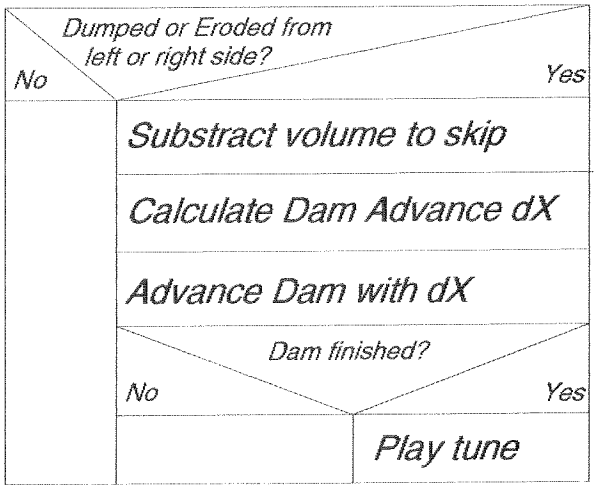
* *Horizontal construction*

In case of horizontal construction, the structure of the procedure is more or less the same. In this case though, the dam advance is expressed in *dX*. The variables which are required for the calculations are:

GradualRight	The production at the right side
ErosionRight	The total erosion from the right side

The following variables can be adapted by the procedure:

Construction	The construction phase *(Busy, Finished* or *Collapsed)*
SkippedVolumeRight	Volume correction for the right side
XDRL, XDRR	X-coordinates of toes at left and right side
DamTop	Required dam top level
DamYR	Right side top of dam head
YDR	Top level of the sill
ProcessRight	Calculation process for the right side
DumpedRight	Total volume dumped from the right side
ErodedRight	Total volume eroded from the right side



* *Verification after construction activities*

After the construction activities have been expressed in a dam advance, a check is made:

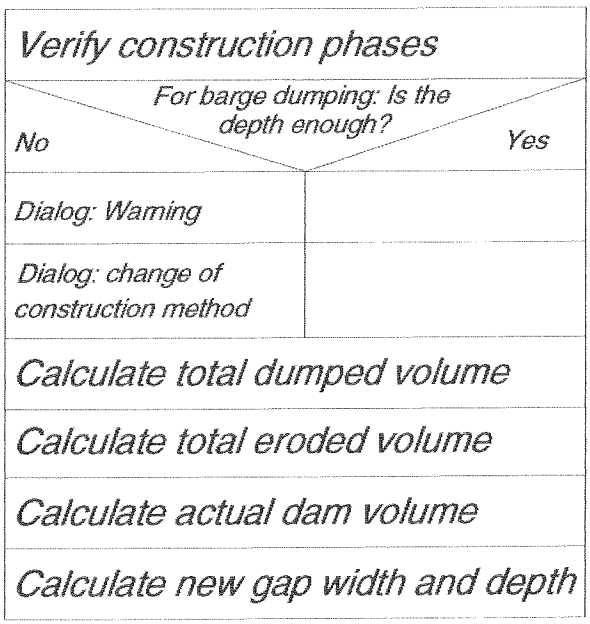
In the case of dumping by barges, the depth should be enough; if not, the user is warned and should select another construction method.

The totals of dumped and eroded volume are calculated - their difference is the actual dam volume. The new width and depth of the gap are calculated for the hydraulic calculations in the next time step. For these calculations, the following variables are necessary:

OrigGapWidth	Original width of gap
DamTop	Required dam top level
ActSeaLevel	Actual Sea Level
VesselDraft	Draft of dumping vessels
ClosureScheme	Closure scheme *(Horizontal, Vertical, Both)*
ClosureDirection	Closure direction *(Left, Right, Both)*

These variables can be adapted by the procedure:

WayOfConstruction	Actual way of construction
SkippedVolumeRight	Volume correction for the right side
SkippedVolumeLeft	Volume correction for the left side
ProcessLeft	Construction phase for the left side
ProcessRight	Construction phase for the right side
DamYL, DamYR	X-coordinates of top of dam heads
XDRL, XDRR	X-coordinates of toes
TotalDumped	Total dumped volume
TotalEroded	Total eroded volume
ActualVolume	Actual volume of dam
GapWidth	Actual width of flow gap
Construction	Construction phase *(Busy, Finished, Collapsed)*



Pascal Listing of procedure Activities

Below, the four sections of the procedure *Activities* are given in Pascal.:

```
procedure Activities;

var Dt,                                          {width between toes of dam heads}
    Bb,                                            {averaged width of dam section}
    A, B, C, D                                       {variables for A-B-C formula}
    dH,                                            {advance of sill top level}
    dX,                                                {advance of dam head}
    OXDRL,                                  {old location of  left toe, relative}
    OXDRR,                                  {old location of right toe, relative}
    OYDR,                                                {old sill top level}
    GapLeft,                                          {location of left toe}
    GapRight,                                        {location of right toe}
    ErosionLengthLeft,                      {length of erosion area,  left slope}
    ErosionLengthRight,                     {length of erosion area, right slope}
    ErosionLengthUp,                          {length of erosion area, sill top}
    TotalErosionLength,                          {Total length of erosion areas}
    ErosionLeft,                           {eroded volume taken from  left slope}
    ErosionRight,                          {eroded volume taken from right slope}
    ErosionUp,                               {eroded volume taken from sill top}
    AddFill,                                   {added volume to construction area}
    SubFill,                        {eroded volume, taken from construction area}
    TotFill: Real;                      {total volume, added or eroded from area}
    LastProcessLeft,                          {last construction phase,  left side}
    LastProcessRight,                         {last construction phase, right side}
    LastProcessUp: Integer;                      {last construction phase, on sill}
begin
```

```
{=========================================================}
{PROGRAM SECTION BELONGING TO FIRST STRUCTURE DIAGRAM}
{=========================================================}

  for GCT:=1 to NumOfGaps do
  begin
      if Construction[GCT]=Collapsed          {if construction collapsed, then stop}
         then NoActivities;                            {all construction activities}
      if  (YDR[GCT]>=UpMaxLevel[GCT])                       {if top of sill reached}
      and (GradualUp[GCT]<>0) then                        {and if vertically dumping}
      begin
         GradualUp[GCT]:=0;                            {then stop vertical activities}
      end;
      if Construction[GCT]=Collapsed          {if construction collapsed, then stop}
         then NoActivities;                            {all construction activities}

                                                              {Erosion of Gap Material}

      if (YDR[GCT]<MaxLevel) then             {if sill top lower than maximum level}
                                                  {then calculate eroded quantities}
      begin
         TotalErosionLength:=0;
         if DumpedLeft[GCT]<>0   then
             ErosionLengthLeft:=(DamYL[GCT]-YDR[GCT])/Sin(DSH[GCT])
                                 else ErosionLengthLeft:=0;
         if DumpedRight[GCT]<>0  then
             ErosionLengthRight:=(DamYR[GCT]-YDR[GCT])/Sin(DSH[GCT])
                                 else ErosionLengthRight:=0;
         ErosionLengthUp:=XDRR[GCT]-XDRL[GCT];
         TotalErosionLength:=ErosionLengthLeft+ErosionLengthUp+ErosionLengthRight;
         TotalErosion[GCT] := TotalErosionLength*ErosionPerMPerS[GCT];
         ErosionLeft  := TotalErosion[GCT]*(ErosionLengthLeft /TotalErosionLength);
         ErosionRight := TotalErosion[GCT]*(ErosionLengthRight/TotalErosionLength);
         ErosionUp    := TotalErosion[GCT]*(ErosionLengthUp   /TotalErosionLength);
      end
      else
      begin
         ErosionLeft:=0;                  {if both water levels are lower than the}
         ErosionRight:=0;                  {deepest construction activity, then the}
         ErosionUp:=0;                                  {erosion at all sides is zero}
         TotalErosion[GCT]:=0;
      end;


{=========================================================}
{PROGRAM SECTION BELONGING TO SECOND STRUCTURE DIAGRAM}
{=========================================================}

      if ((GradualUp[GCT]<>0) or (ErosionUp<>0))              {if on sill something}
      and (Construction[GCT]<>Finished)                        {is dumped or eroded}
      and (YDR[GCT]<UpMaxLevel[GCT]) then                {and sill not normald yet}
      begin                                                        {Vertical Closure}
        AddFill:=(GradualUp[GCT]/3600)*TimeStep;                      {volume dumped}
        SubFill:=ErosionUp*TimeStep;                                  {volume eroded}
        TotFill:=AddFill-SubFill;                                      {total volume}
        OYDR:=YDR[GCT];
        Dt:=XDRR[GCT]-XDRL[GCT]; if Dt<0 then Dt:=0;
        Bb:=DamTopWidth[GCT]+2*(DamTop[GCT]-YDR[GCT])/Tan(DSS[GCT]);
        ActDamHeight[GCT]:=YDR[GCT]-OrigGapDepthAtMSL[GCT];
        if WayOfConstruction[GCT]=csCableWay then                    {remember whether}
                                                          {cableway has been used}

        begin                                                {if cableway construction:}
           Bb:=DamTopWidth[GCT]+2*ActDamHeight[GCT]/Tan(DSS[GCT]);
           OnceCableWay[GCT]:=Yes;
        end;
```

```
                                                    {calculate dam advance from volume.}
                                                      {See manual volume I, Appendix B}
        A:=-Dt/Tan(DSS[GCT]);
        if (DumpedLeft[GCT]<>0) then A:=A+0.5*Bb/Tan(DSH[GCT]);
        if (DumpedRight[GCT]<>0) then A:=A+0.5*Bb/Tan(DSH[GCT]);
        B:=Dt*Bb;
        C:=-TotFill;
        D:=B*B-4*A*C;
        if D>=0 then
        if A<>0 then dH:=(-B+SQRT(D))/(2*A)
                else begin dH:=0; end;
        if Sgn(dH)<>Sgn(TotFill) then dH:=0;
        YDR[GCT]:=YDR[GCT]+dH;
        if YDR[GCT]>=DamTop[GCT] then
        begin
           Construction[GCT]:=Finished;              {in case dam has been finished}
           MultiBeep;                                                  {play tune}
           if (YDR[GCT]-OYDR)<>0 then if YDR[GCT]-OYDR<>0 then
           begin
              AddFill:=AddFill*(DamTop[GCT]-OYDR)/(YDR[GCT]-OYDR);
              SubFill:=SubFill*(DamTop[GCT]-OYDR)/(YDR[GCT]-OYDR);
              TotFill:=TotFill*(DamTop[GCT]-OYDR)/(YDR[GCT]-OYDR);
           end;
           YDR[GCT]:=DamTop[GCT];
        end;
                                                    {vertical dumping also effects}
        if (Construction[GCT]<>Finished) then                       {locations toes}
        begin
          if  (DumpedLeft[GCT]<>0) then XDRL[GCT]:=XDRL[GCT]-(dH/Tan(DSH[GCT]));
          if (DumpedRight[GCT]<>0) then XDRR[GCT]:=XDRR[GCT]+(dH/Tan(DSH[GCT]));
        end;
        DumpedUp[GCT]:=DumpedUp[GCT]+AddFill;                   {add dumped volume}
        ErodedUp[GCT]:=ErodedUp[GCT]+SubFill;                {substract eroded volume}

                                                  {if toes originally touched,}
        if TotFill>0 then                      {after vertical dumping they will not}
        begin
           if ProcessLeft[GCT] =Toes      then ProcessLeft[GCT] :=Full;
           if ProcessLeft[GCT] =StartToes then ProcessLeft[GCT] :=Start;
           if ProcessLeft[GCT] =Reached   then ProcessLeft[GCT] :=Full;
           if ProcessRight[GCT]=Toes      then ProcessRight[GCT]:=Full;
           if ProcessRight[GCT]=StartToes then ProcessRight[GCT]:=Start;
           if ProcessRight[GCT]=Reached   then ProcessRight[GCT]:=Full;
        end;
    end;


{=====================================================}
{PROGRAM SECTION BELONGING TO THIRD STRUCTURE DIAGRAM}
{=====================================================}

    if ((GradualRight[GCT]<>0) or (ErosionRight<>0))
    and (Construction[GCT]<>Finished) then
    begin
                                                {Horizontal Closure from RIGHT Side}
       AddFill:=(GradualRight[GCT]/3600)*TimeStep;          {volume to be skipped is}
       AddFill:=AddFill-SkippedVolumeRight[GCT];          {extracted, process waits}
       if AddFill<0 then begin
                            SkippedVolumeRight[GCT]:=Abs(AddFill);
                            AddFill:=0;
                         end
                         else SkippedVolumeRight[GCT]:=0;
       SubFill:=ErosionRight*TimeStep;                          {calculate sum of}
       TotFill:=AddFill-SubFill;                              {constructing volumes}
       OXDRR:=XDRR[GCT]; OYDR:=YDR[GCT];
       Bb:= DamTopWidth[GCT]+
         (DamTop[GCT]-DamYR[GCT])/Tan(DSS[GCT])+
```

```
  (DamTop[GCT]-  YDR[GCT])/Tan(DSS[GCT]);        {A-B-C formula, see manual}
 if Bb<>0 then C:=-TotFill/Bb;
 if (ProcessRight[GCT]=Full)
 or (ProcessRight[GCT]=StartReached) then A:=0;
 if ProcessRight[GCT]=Start        then
    A:= 0.5*(Sin(DSH[GCT])*Cos(DSH[GCT]) + SQR(Sin(DSH[GCT]))*Tan(DSH[GCT]));
 if ProcessRight[GCT]=StartToes    then
    A:= 0.5*(Sin(DSH[GCT])*Cos(DSH[GCT]) + SQR(Sin(DSH[GCT]))*Tan(DSH[GCT]))
-0.25*Tan(DSH[GCT]);
 if ProcessRight[GCT]=Reached      then

A:=-0.5*Sqr(Sin(DSH[GCT]))*Tan(DSH[GCT])-0.5*(Sin(DSH[GCT])*Cos(DSH[GCT]));
 if ProcessRight[GCT]=Toes         then A:=-0.25*Tan(DSH[GCT]);
B:=DamYR[GCT]-YDR[GCT];
 if B<0 then B:=0;
D:=B*B-4*A*C;
 if D<0 then begin D:=0; if a<>0 then C:=B*B/(4*A) end;
 if (A=0) and (B<>0) then dX:=-C/B
                     else if a<>0 then dX:=(-B+SQRT(D))/(2*A);

 if ProcessRight[GCT]=Start then        {application dam advance, see manual}
begin
   DamYR[GCT]:=DamYR[GCT]+dX*Tan(DSH[GCT]);
   if DamYR[GCT]>DamTop[GCT] then DamYR[GCT]:=DamTop[GCT];
   XDRR[GCT]:=XDRR[GCT]-dX;
end;
if ProcessRight[GCT]=StartToes then
begin
  DamYR[GCT]:=DamYR[GCT]+dX*Tan(DSH[GCT]);
  if DamYR[GCT]>DamTop[GCT] then DamYR[GCT]:=DamTop[GCT];
  YDR[GCT]:=YDR[GCT]+0.5*dX*Tan(DSH[GCT]);
  XDRR[GCT]:=XDRR[GCT]-0.5*dX;
  XDRL[GCT]:=XDRR[GCT];
end;
if ProcessRight[GCT]=StartReached then
begin
  DamYR[GCT]:=DamYR[GCT]+dX*Tan(DSH[GCT]);
  if DamYR[GCT]>DamTop[GCT] then DamYR[GCT]:=DamTop[GCT]
  else YDR[GCT]:=YDR[GCT]+dX*Sin(DSH[GCT])*Cos(DSH[GCT]);
end;
if ProcessRight[GCT]=Full then
begin
  XDRR[GCT]:=XDRR[GCT]-dX;
end;
if ProcessRight[GCT]=Reached then
begin
  YDR[GCT]:=YDR[GCT]+dX*Sin(DSH[GCT])*Cos(DSH[GCT]);
  DamYL[GCT]:=YDR[GCT];
  if DamYL[GCT]>DamTop[GCT] then DamYL[GCT]:=DamTop[GCT];
end;
if ProcessRight[GCT]=Toes then
begin
  YDR[GCT]:=YDR[GCT]+0.5*dX*Tan(DSH[GCT]);
  XDRR[GCT]:=XDRR[GCT]-0.5*dX;
end;

if YDR[GCT]>=DamTop[GCT] then              {if location where toes touch is}
                              {higher than dam top, construction is finished}
begin
   Construction[GCT]:=Finished;
   MultiBeep;
   if YDR[GCT]-OYDR<>0 then
    begin
       AddFill:=AddFill*(DamTop[GCT]-OYDR)/(YDR[GCT]-OYDR);
       SubFill:=SubFill*(DamTop[GCT]-OYDR)/(YDR[GCT]-OYDR);
       TotFill:=TotFill*(DamTop[GCT]-OYDR)/(YDR[GCT]-OYDR);
    end;
```

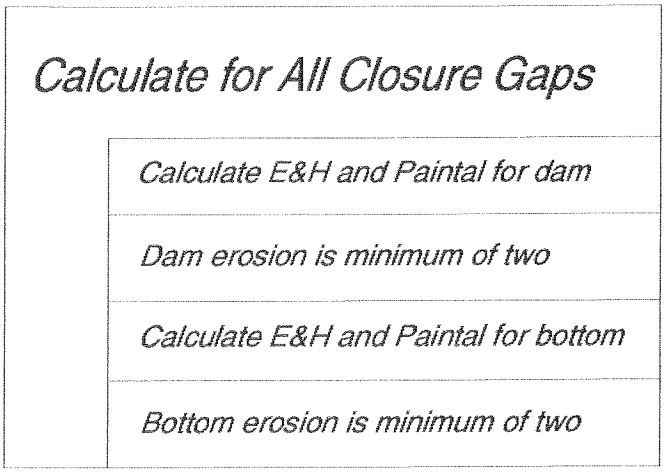
```
        YDR[GCT]:=DamTop[GCT];
      end;

      DumpedRight[GCT]:=DumpedRight[GCT]+AddFill;
      ErodedRight[GCT]:=ErodedRight[GCT]+SubFill;

    end;

                    {The program section for the dumping and eroding of material}
                         {from the left side is identical to the section for the}
                       {dumping and eroding of material from the right side, and}
                      {for that reason this section has not been reproduced here}


{=====================================================}
{PROGRAM SECTION BELONGING TO FOURTH STRUCTURE DIAGRAM}
{=====================================================}

                                              {Verify actual construction situation}

                                          {stop skipping if no volume left to be skipped}

    if SkippedVolumeRight[GCT]<10 then SkippedVolumeRight[GCT]:=0;
    if SkippedVolumeLeft[GCT]<10  then SkippedVolumeLeft[GCT]:=0;

    if Construction[GCT]<>Finished then
    begin
                            {side levels of dam can never be higher than sill top!}
      if DamYL[GCT]<YDR[GCT] then DamYL[GCT]:=YDR[GCT];
      if DamYR[GCT]<YDR[GCT] then DamYR[GCT]:=YDR[GCT];

       LastProcessLeft:=ProcessLeft[GCT];                          {remember last}
      LastProcessRight:=ProcessRight[GCT];                      {construction phase}


                        {in the lines below, the administration of construction}
                      {phases is handled. See manual volume I, appendix B for a}
                    {detailed description of the possible phases of construction}

      if (XDRL[GCT]<(DamTop[GCT]-YDR[GCT])/Tan(DSH[GCT])) then
      begin
        if LastProcessLeft =Reached then ProcessLeft[GCT]:=StartReached;
        if LastProcessLeft =Toes    then ProcessLeft[GCT]:=StartToes;
      end
      else
      if DamYL[GCT]=DamTop[GCT] then
      begin
        if ProcessLeft[GCT]=Start        then ProcessLeft[GCT]:=Full;
        if ProcessLeft[GCT]=StartReached then ProcessLeft[GCT]:=Reached;
        if ProcessLeft[GCT]=StartToes    then ProcessLeft[GCT]:=Toes;
      end;

      if (XDRR[GCT]>(OrigGapWidth[GCT]-(DamTop[GCT]-YDR[GCT])/Tan(DSH[GCT])))
      then
      begin
        if LastProcessRight=Reached then ProcessRight[GCT]:=StartReached;
        if LastProcessRight=Toes    then ProcessRight[GCT]:=StartToes;
      end
      else
      if DamYR[GCT]=DamTop[GCT] then
      begin
        if ProcessRight[GCT]=Start        then ProcessRight[GCT]:=Full;
        if ProcessRight[GCT]=StartReached then ProcessRight[GCT]:=Reached;
        if ProcessRight[GCT]=StartToes    then ProcessRight[GCT]:=Toes;
      end;
```

```
if (XDRR[GCT]<=0) then
begin
   XDRR[GCT]:=0;
   XDRL[GCT]:=0;
   ProcessLeft[GCT]:=StartToes;
   if ProcessRight[GCT]=Full      then ProcessRight[GCT]:=Reached;
   if ProcessRight[GCT]=Start     then ProcessRight[GCT]:=StartReached;
   if ProcessRight[GCT]=Toes      then ProcessRight[GCT]:=Reached;
   if ProcessRight[GCT]=StartToes then ProcessRight[GCT]:=StartReached;
end
else
begin
   If ProcessRight[GCT]=Reached      then ProcessRight[GCT]:=Full;
   If ProcessRight[GCT]=StartReached then ProcessRight[GCT]:=Start;
end;

if (XDRL[GCT]>=OrigGapWidth[GCT]) then
begin
   XDRR[GCT]:=OrigGapWidth[GCT];
   XDRL[GCT]:=OrigGapWidth[GCT];
   ProcessRight[GCT]:= StartToes;
   if ProcessLeft[GCT]=Full      then ProcessLeft[GCT]:=Reached;
   if ProcessLeft[GCT]=Start     then ProcessLeft[GCT]:=StartReached;
   if ProcessLeft[GCT]=Toes      then ProcessLeft[GCT]:=Reached;
   if ProcessLeft[GCT]=StartToes then ProcessLeft[GCT]:=StartReached;
end
else
begin
   if ProcessLeft[GCT]=Reached      then ProcessLeft[GCT]:=Full;
   if ProcessLeft[GCT]=StartReached then ProcessLeft[GCT]:=Start;
end;

if YDR[GCT]>=DamTop[GCT] then                        {last check for finished}
begin                                                          {construction}
  Construction[GCT]:=Finished;
  Multibeep;
  if (YDR[GCT]-OYDR)<>0 then if YDR[GCT]-OYDR<>0 then
  begin
     AddFill:=AddFill*(DamTop[GCT]-OYDR)/(YDR[GCT]-OYDR);
     SubFill:=SubFill*(DamTop[GCT]-OYDR)/(YDR[GCT]-OYDR);
     TotFill:=TotFill*(DamTop[GCT]-OYDR)/(YDR[GCT]-OYDR);
  end;
  YDR[GCT]:=DamTop[GCT];
end;

if (ActSeaLevel-YDR[GCT]<VesselDraft)          {if too shallow for barges}
and (WayOfConstruction[GCT]=csVessels)                {choose other method}
then begin
   Warn_VesselDraft;
   NoActivities;
   case NumOfGaps of                                      {show dialog boxes}
   1: begin
      Dial_Dimensions1(DamTopWidth, DamTop, WayOfConstruction);
      Dial_GradualClose1(GradualLeft, GradualRight,
                         GradualUp, UpMaxLevel, DamD50);
      end;
   2: begin
      Dial_Dimensions2(DamTopWidth, DamTop, WayOfConstruction);
      Dial_GradualClose2(GradualLeft, GradualRight,
                         GradualUp, UpMaxLevel, DamD50);
      end;
   3: begin
      Dial_Dimensions3(DamTopWidth, DamTop, WayOfConstruction);
      Dial_GradualClose3(GradualLeft, GradualRight,
                         GradualUp, UpMaxLevel, DamD50);
      end;
   end;
```

```
        end;

        if (XDRL[GCT]>=XDRR[GCT]) and (ProcessRight[GCT]<>Reached)
                                  and (ProcessLeft[GCT] <>Reached) then
        begin
           if ProcessRight[GCT]=Start then ProcessRight[GCT]:=StartToes;
           if ProcessRight[GCT]=Full  then ProcessRight[GCT]:=Toes;
           if ProcessLeft[GCT] =Start then ProcessLeft[GCT] :=StartToes;
           if ProcessLeft[GCT] =Full  then ProcessLeft[GCT] :=Toes;
           XDRL[GCT]:=XDRR[GCT];
        end;

                                                {add constructed and eroded volumes}

        TotalDumped[GCT]:=DumpedLeft[GCT]+DumpedRight[GCT]+DumpedUp[GCT];
        TotalEroded[GCT]:=ErodedLeft[GCT]+ErodedRight[GCT]+ErodedUp[GCT];
        ActualVolume[GCT]:=TotalDumped[GCT]-TotalEroded[GCT];
    end;
  end;

  for GCT:=1 to NumOfGaps do
  begin
                                                        {determine location  left toe}
     GapLeft :=XDRL[GCT]-0.5*(DamYL[GCT]-YDR[GCT])/Tan(DSH[GCT]);
     if GapLeft<0 then GapLeft:=0;
                                                        {determine location right toe}
     GapRight:=XDRR[GCT]+0.5*(DamYR[GCT]-YDR[GCT])/Tan(DSH[GCT]);
     if GapRight>OrigGapWidth[GCT] then GapRight:=OrigGapWidth[GCT];

     GapWidth[GCT]:=GapRight-GapLeft;       {gap width is difference toe locations}

     if GapWidth[GCT]<0.01 then GapWidth[GCT]:=0;

                                                    {determine actual closure method}
     ClosureScheme[GCT]:=Vertical; ClosureDirection[GCT]:=Up;

     if (DumpedLeft[GCT]>0) or (DumpedLeft[GCT]>0) then
     ClosureScheme[GCT]:=Horizontal;

     if (DumpedLeft[GCT]>0)  then ClosureDirection[GCT]:=Left;
     if (DumpedRight[GCT]>0) then ClosureDirection[GCT]:=Right;
     if (DumpedLeft[GCT]>0) and (DumpedRight[GCT]>0) then
                                  ClosureDirection[GCT]:=Both;
  end;
end;
```

6.2.8 Erosion of Dam Material

The backgrounds on the erosion of dam material have been explained in volume *I* of this paper. Two transport formulas are used, depending on the shear stress parameter of Shields; in practice this means, that always the least of the two is chosen. The erosion of bottom material is done in the same formula, but then another erosion length and eroding material are used. The structure of this algoritm is:



which in Pascal is:

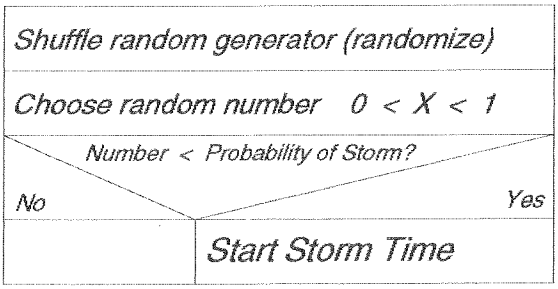
```
procedure DamDestruction;                                    {CALCULATION OF
LOSS OF DAM MATERIAL}

var ME, MP,
    ErosE, ErosP: Real;
    BL, SQBL, SQEPM: Real;

begin
   for GCT:=1 to NumOfGaps do
   begin
                                                                {Calculate
Erosion Dam Material}
      ME:=0.084/(Power(Chezy,3)*SQRT(Gravity)*SQR(Delta)*(DamD50[GCT]/1000));
      MP:=6.56e+18*2600*Power(gravity, 1.5);
      ErosE:=ME*Power(Abs(Velocity[GCT]*CrestMult[GCT]),5);
      begin
       ErosP:=MP*SQR(SQR(SQR(SQR(SQR(
           Abs(Velocity[GCT]*CrestMult[GCT]/
           (Chezy*Power(Delta,0.484375)*Power(DamD50[GCT]/1000,0.4531)))))));
       ErosionPerMPerS[GCT]:=Min(ErosE, ErosP);
      end

                                                    {Calculate Erosion Bottom
Material At Dam Location}
```

```
      ME:=0.084/(Power(Chezy,3)*SQRT(Gravity)*SQR(Delta)*(BottomD50/1000));
      ErosE:=ME*Power(Abs(Velocity[GCT]*CrestMult[GCT]),5);
      begin
        ErosP:=MP*SQR(SQR(SQR(SQR(SQR(
             Abs(Velocity[GCT]*CrestMult[GCT]/
            (Chezy*Power(Delta,0.484375)*Power(BottomD50/1000,0.4531)))))));
        BotErosPerMPerS[GCT]:=Min(ErosE, ErosP);
      end
    end;
end;
```

6.2.9 Checking whether a storm occurs

During the program, a storm can occur. The way a storm is schematized in the program, has been explained in chapter *3.1.4* of volume I. Here it is sufficient to mention that a storm setup is dealt with in procedure *SeaLevel* (unit *SIMTOOLS)*, and that the momentaneous velocity increase is calculated in procedure *CalcKomb* (unit *SIMCALC)*. The procedure described below only checks whether a storm has to begin, and if that is so, sets the storm time counter.

The procedure *StormCheck* is called by the main procedure *Calculate* and does not interact with other procedures itself.



The procedure requires the variables *StormChance* (the probability for storm) and *StormPeriod* (the length of the storm, in seconds). It adapts the variable *TimeToStorm* (and makes the time to storm equal to the storm period).

Which, in Pascal, has been written as:

```
Procedure StormCheck;                          {CHECK WHETHER A STORM OCCURS}

var RN: Real;                                                 {Random variable}

begin
   Randomize;                                                 {Shuffle randomizor}
   RN:=Random(10000)/10000;                  {Determine random number smaller than 1}
   if RN<StormChance then                     {if the random number smaller than the}
   begin                                                {actual storm chance, then}
      TimeToStorm:=StormPeriod*3600;          {start storm by setting the counter}
   end;
end;
```

6.2.10 Checking a vertical dumping inaccuracy

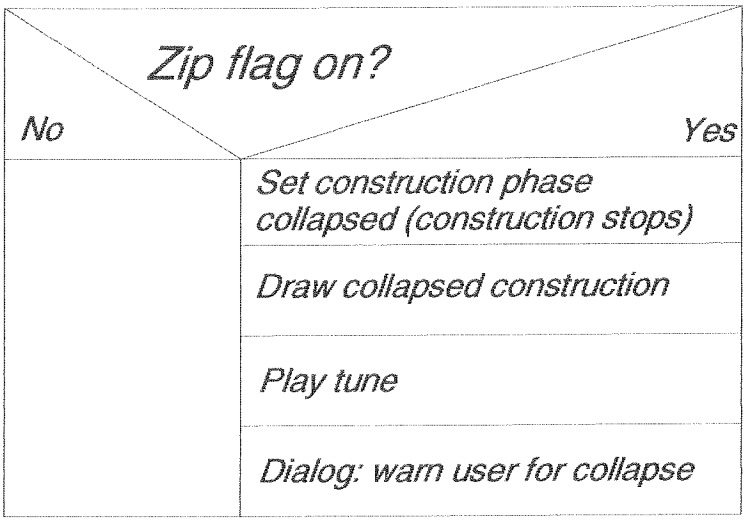
Vertical construction can never be so accurate, that at each moment the sill rise is the same for the whole dam width. A dumping inaccuracy - a location where the dam is lower - can lead to collapse of the dam though, as has been explained in chapter *5.6* of volume I of this paper.

During the calculation of levels, flows and velocities, a check is made whether the stones in a deeper section of the sill are stabile. If they are not, the so-called *zip-flag* is put on. In the calculation loop is looked at this zip flag, and if it has been put on, the construction collapses.

In case an instability occurs, the procedure calls to *MultiBeep* (to produce a collapse tune), to *WarnZipped* (to show a message box) and to *NoActivities* (to stop all construction activities). The procedure itself is called from the main procedure *Calculate*.

The procedure requires and resets the boolean variable *Zipped*.

The structure of this check is as follows:



In Pascal this check has been written as:

```
Procedure ZipCheck;          {CHECK WHETHER A DUMPING INACCURACY HAS CAUSED COLLAPSE}

begin
   if (Zipped=True) then                                    {if a deeper stone has moved}
   begin
      MultiBeep; MultiBeep; MultiBeep;                          {then play warning tune}
      Warn_Zipped;                                                  {insert dialog box}
      for GCT:=1 to NumOfGaps do
      begin
         NoActivities;                                {stop all construction activities}
         Zipped:=False;                                             {reset warning flag}
      end;
   end;
end;
```

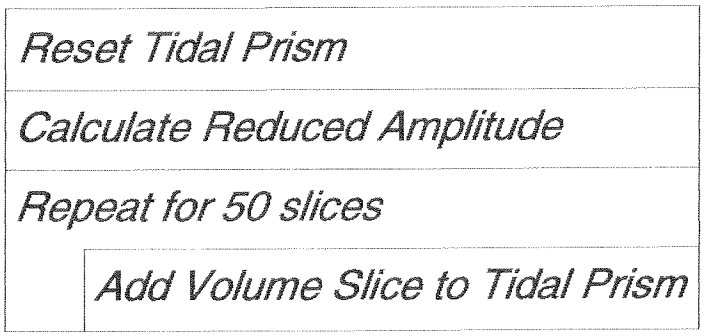
6.2.11 Calculation of the Equilibrium Profile

Some calculation procedures are not called from within the procedure *Calculate*, simply because their calculations are not executed in every time step. Three calculations are made when the process is started: the equilibrium profile is calculated and the actual profiles are increased or enlarged according to this equilibrium profile; in another procedure some maximum values are stored. In this paragraph, the procedure for the calculation of the equilibrium profile has been treated.

For several reasons, we wish to know the equilibrium profile; this is the channel section that exists if nature is not intervened by construction activities. To calculate this equilibrium profile, the tidal prism (the volume of water that every tidal period enters and leaves the basin) should be known. In the structure diagram below is shown how this volume is calculated. The equilibrium profile is linearly dependent on this tidal prism (see volume I, Chapter *3.2.2*).

The variables which are required for the calculation of tidal prism and equilibrium volume, are the array of the four tidal amplitudes *TideAmp*, and the variable *MeanSeaLevel*. The calculated value is given back in the variable *EquiTotalGapSurface*.

The procedure calls to the function *StorageSurface* in order to calculate the storage surface of the basin.



In Pascal, the calculation of the tidal prism is as below:

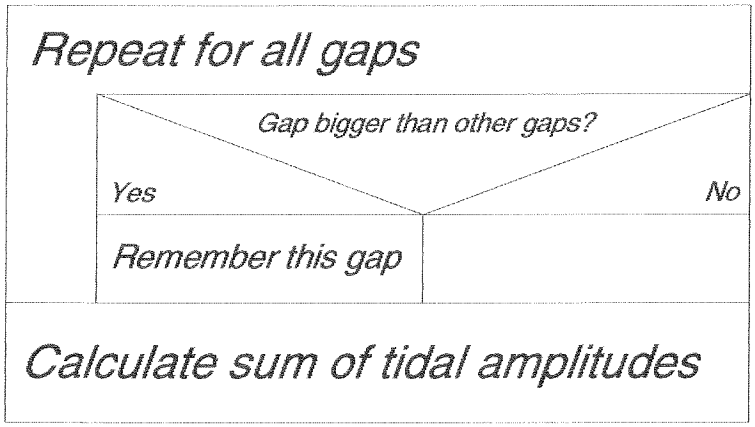
```
procedure Determine_EquilibriumProfile;          {DETERMINE EQUILIBRIUM GAP PROFILE}

var CalcAmplitude,                                      {reduced sum of amplitudes}
    Slide: Real;                                  {thickness of a slice of water}
```

```
begin
  TidalPrism:=0;                                          {RESET accumulator}
  CalcAmplitude:=0.7*(TideAmp[1]
                      +TideAmp[2]                  {calculation is executed using a}
                      +TideAmp[3]                        {REDUCED SUM of amplitudes}
                      +TideAmp[4]);                 {reader is referred to manuals}

  For Counter:=1 to 50 do                                 {DIVIDE tidal volume}
  begin                                                   {         into fifty}
      Slide:=2*CalcAmplitude/50;                          { calculation layers}

      TidalPrism:=TidalPrism+          {INCREASE tidal prism with volume of slide}

      StorageSurface(MeanSeaLevel-(25+Counter)*Slide)*Slide;
  end;
  EquiTotalGapSurface:=TidalPrism/16500;          {CALCULATE equilibrium profile}

                                    {Reader is referred to volume I, section 3.2.2}
end;
```

6.2.12 Determination and storage of maximum values

In several cases, the program needs to know maximum values. The scour hole with the maximum depth is drawn; the maximum depth is needed for the profile drawings; the maximum possible tidal amplitude is needed for the scaling of the level drawing. As an example, the structure diagram is given for the determination of the maximum gap depth. In this procedure, also the sum of the tidal amplitudes is calculated.

Of course, which variables are effected by this algorithm depends on where it is used. In the example below, the variables *GapDepthAtMSL* and *DamTop* are used to calculate the maximum required dam height, *OrigMaxHeight*. The total tidal amplitude possible, is calculated from the array of specific tidal amplitudes *TideAmp* and is stored in the real variable *TotalAmplitude*. The procedure only calls the function *Min* to determine which of two variables is minimum. The structure diagram is:



In Pascal the algorithm shown above occurs in the procedure *DetermineMaxima*:

```
Procedure DetermineMaxima;                                  {CALCULATE MAXIMUM VALUES}

begin
  for GCT:=1 to NumOfGaps do                              {Calculate Maximum Dam Height}
  begin
                              {minimum is calculated because the values are negative}
      OrigMaxHeight:=Min(OrigMaxHeight, GapDepthAtMSL[GCT]-DamTop[GCT]);
  end;

  TotalAmplitude:=0;                          {Calculate Maximum Tidal Amplitude Possible}
  For Counter:=1 to 4 do
     TotalAmplitude:=TotalAmplitude+TideAmp[Counter];
end;
```

6.2.13 Calculation of the Slope Angle

In this procedure, the angle of the side and head slopes of the construction to be built has been approximated. For this angle no clear expressions exist. As explained in volume one of this paper, the angle of deposure of material mainly depends on the way of construction of the dam (dumping by trucks produces quite another angle than the dumping by pipe lines) and on the size of the material.

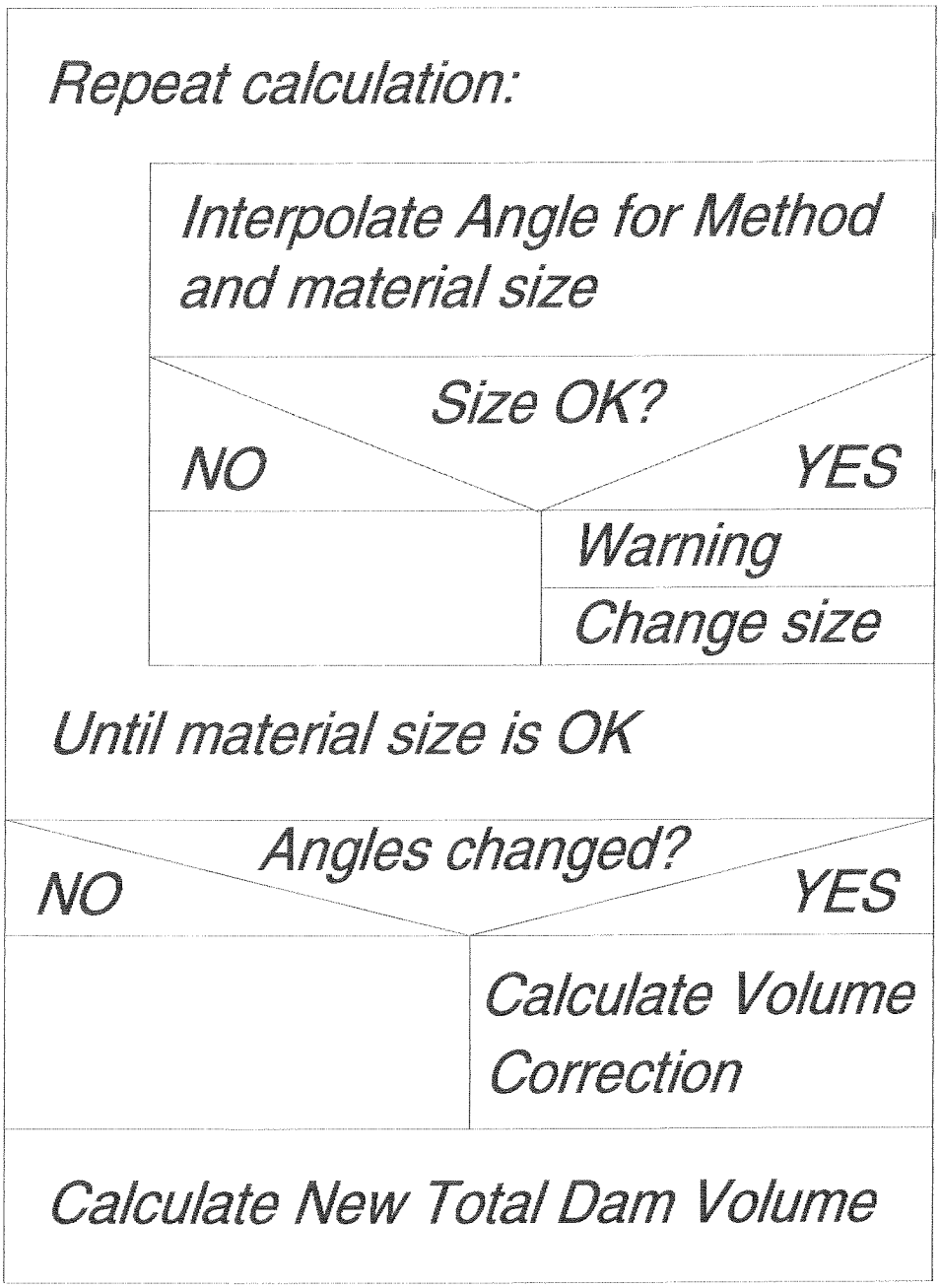
To obtain an angle of deposure of the material, from experience some combinations of way of construction, material size and slope angle have been entered in the program; to calculate the actual angle, an interpolation between those values is made.

After the new slope angles have been calculated, the new slope will allways be steeper than the old slope. The volume between those two slopes has to be calculated and corrected in the calculations of the dam volume.

The procedure uses the variable *WayOfConstruction* (which can be *csTrucks, csVessels, csCableWay* or *csSandDump)* and the size of the construction material *DamD50*. The arrays for the slope angles, *DSS* and *DSH* are adapted, as well as the correction volumes *SkippedVolumeLeft* and *SkippedVolumeRight,* for which *DamTopWidth* and *DamHeight* are used. Other required variables are *ClosureScheme (Horizontal, Vertical* or *Both)* and *TotalDumped* (the total dumped volume).

The procedure calls to *Warn_Material* (to warn for wrongly chosen material sizes) and *Opt_GradualClose* (to choose a new material size) and after the changes have been made, it calls *CalculateRequiredVolume* to calculate the new dam volume.

The structure diagram of this calculation is:



Below, the Pascal source file for this calculation is given.

```
procedure CalculateAngles(WayOfConstruction: IntArr; Var DSS, DSH: RealArr);
```

```
var OldDSS: RealArr;
    CoTanAngle,
    W, H, T0, T1: Real;

const TooBig=1; TooSmall=2; SizeOK=3;

begin
   for GCT:=1 to NumOfGaps do
   repeat
      OldDSS[GCT]:=DSS[GCT];
      MaterialSize:=SizeOK;
      Case WayOfConstruction[GCT] of
      csTrucks: begin                          {Calculate angles in case of truck
dumping}

         if DamD50[GCT]<0.5 then MaterialSize:=TooSmall;
         if (DamD50[GCT]>=0.5) and (DamD50[GCT]<=50) then
         CoTanAngle:=5-(5-2)*(DamD50[GCT]-0.5)/(50-0.5);
         if (DamD50[GCT]>=50) and (DamD50[GCT]<=2000) then
         CoTanAngle:=2-(2-1.5)*(DamD50[GCT]-50)/(2000-50);
      end;
      csVessels: begin                        {Calculate angles in case of vessel
dumping}

         if DamD50[GCT]<0.5 then MaterialSize:=TooSmall;
         if (DamD50[GCT]>=0.5) and (DamD50[GCT]<=50) then
         CoTanAngle:=10-(10-4)*(DamD50[GCT]-0.5)/(50-0.5);
         if (DamD50[GCT]>=50) and (DamD50[GCT]<=2000) then
         CoTanAngle:=4-(4-1.5)*(DamD50[GCT]-50)/(2000-50);
      end;
      csCableWay: begin                    {Calculate angles in case of cable way
dumping}

         if DamD50[GCT]<100 then MaterialSize:=TooSmall;
         CoTanAngle:=2-(2-1.5)*(DamD50[GCT]-50)/(2000-50);
      end;
      csSandDump: begin                    {Calculate angles in case of pipe line
dumping}
         if DamD50[GCT]<0.15 then begin MaterialSize:=TooSmall; end;
         if DamD50[GCT]>50  then MaterialSize:=TooBig;
         if (DamD50[GCT]>=0.15) and (DamD50[GCT]<=0.25) then
         CoTanAngle:=40-(40-25)*(DamD50[GCT]-0.15)/(0.25-0.15);
         if (DamD50[GCT]>=0.25) and (DamD50[GCT]<=0.5) then
         CoTanAngle:=25-(25-10)*(DamD50[GCT]-0.25)/(0.5-0.25);
         if (DamD50[GCT]>=0.5) and (DamD50[GCT]<=50) then
         CoTanAngle:=10-(10-4)*(DamD50[GCT]-0.5)/(50-0.5);
      end;
     end;
     if DamD50[GCT]>2000 then MaterialSize:=TooBig;
     if (MaterialSize<>SizeOK) and (Adapting=False) then
                  begin                          {Verify combination of material and}
                                                                {way of construction}
                     Warn_Material(MaterialSize,GCT);
                     Opt_GradualClose;
                  end

        else
        if WayOfConstruction[GCT]=1 then
           begin DSS[GCT]:=Min(ArcTan(1/CoTanAngle), DSS[GCT]);
           DSH[GCT]:=Min(ArcTan(1/CoTanAngle), DSH[GCT]);
        end
        else
        begin
           DSS[GCT]:=ArcTan(1/CoTanAngle);
           DSH[GCT]:=DSS[GCT];
        end;
```

```
   until (MaterialSize=SizeOK) or (Adapting);

                                                          {Calculate skipped
volumes}
                                                          {for volume
corrections}
   for GCT:=1 to NumOfGaps do
   if (TotalDumped[GCT]<>0) and (ClosureScheme[GCT]=Horizontal) then
   begin
      W:=DamTopWidth[GCT];
      H:=DamHeight[GCT];
      T0:=Tan(OLDDSS[GCT]);
      T1:=Tan(DSS[GCT]);
      if (t0<>0) and (t1<>0) then
      begin
         SkippedVolumeLeft[GCT]:=(W+H/T0)*
         (H*H*T1/(T0*T0) - H*H/T0 - SQR(H*T1/T0 - H)/T1);
         SkippedVolumeRight[GCT]:=SkippedVolumeLeft[GCT];
      end
      else
      begin
         SkippedVolumeLeft[GCT]:=0;
         SkippedVolumeRight[GCT]:=0;
      end;
   end;
   CalculateRequiredVolume(GapWidth);                  {Calculate new required
volume}
end;
```

6.3 The Menu Options in the unit GSIMOPTS

All of the menu options in the simulation program, either drawing procedures, calls to dialog boxes or calculation options, point to procedures in the library *GSIMOPTS*. From this library, the procedures are routed their way to other procedures, mostly in other libraries.

In some cases, the option procedure doesn't call any other procedure, but is dealt with in the *GSIMOPTS* unit itself (e.g. the *load* and *save* procedures). In some other procedures, some source code is added. In most cases this is, to distinguish which procedure should be called - for example if an option calls different procedures depending on the actual number of gaps to be closed. In those cases, the source code has been commented.

```
{*********************************************************}
{            CLOSIM v 1.00  --  MENU OPTION UNIT          }
{*********************************************************}
{ Containing the menu options of the Turbo Vision Graphics }
{ User Interface for the simulation program on the Closure }
{ of Tidal Basins. Mark Middag, TUDelft, 1994.            }
{*********************************************************}

Unit GSimOpts;

Interface

uses Crt, Dos, Graph, MCursor2, GApp,                          {Call to units}
     SIMVARS, SIMCALC, SIMTOOLS, GSIMDIAL, SIMDRAWS;

var P: Boolean;

procedure Opt_Load(FTL: FileString);                    {implementation of the}
procedure Opt_Save(FTS: FileString);                   {procedures in this unit}
procedure Opt_InitProcess;
procedure Opt_AdaptAll;
procedure Opt_Start;
procedure Opt_DateTime;
procedure Opt_Basin;
procedure Opt_RiverData;
procedure Opt_Closure;
procedure Opt_DefStorm;
procedure Opt_DepthsWidths;
procedure Opt_Tide;
procedure Opt_Dimensions;
procedure Opt_Methods;
procedure Opt_VesselDraft;
procedure Opt_Angles;
procedure Opt_CopyGap;
procedure Opt_Protection;
procedure Opt_GradualClose;
procedure Opt_Culvert;

Implementation                                  {Definition of written procedures}


procedure CalculateAngles(WayOfConstruction: IntArr; Var DSS, DSH: RealArr);
end;
```

```
if fc=false then
begin
 Started:=False;                                          {Process interrupted}
 for GCT:=1 to NumOfGaps do GapWidth[GCT]:=OrigGapWidth[GCT];
 Equilize_Profiles;                                  {Reset profiles to equilibrium}
 For GCT:=1 to NumOfGaps do
 if (Construction[GCT]=Finished) then
 begin
    Construction[GCT]:=Busy;                  {Define construction process as busy}
    GapDepthAtMSL[GCT]:=OrigGapDepthAtMSL[GCT];
    GapWidth[GCT]:=OrigGapWidth[GCT];
 end;
 JobHours:=0;                                {After, reset all building activities}
 JobDayNo:=1;
 Zipped[GCT]:=False;                                   {Structure did not collapse}
 NumOfKenterings:=0;                                {Reset hydraulic calculations}
 OrigTotalGapSurface:=0;
 OrigTotalWidth:=0;
 BasinLevel:=MeanSeaLevel;
 OldBasinLevel:=MeanSeaLevel;
 StartDrawX[1]:=10;                                        {For drawing of profiles}
 for GCT:=1 to NumOfGaps do
 begin
     OnceCableWay[GCT]:=cmNo;
     ProcessLeft[GCT]        :=  NotBegun;
     ProcessRight[GCT]       :=  NotBegun;
     OrigGapWidth[GCT]       :=  GapWidth[GCT];
     OrigGapDepthAtMSL[GCT]  :=  GapDepthAtMSL[GCT];
     OrigGapSurface[GCT]     :=  abs(OrigGapWidth[GCT]*OrigGapDepthAtMSL[GCT]);
     GapSurface[GCT]         :=  OrigGapSurface[GCT];
     OrigTotalGapSurface     :=  OrigTotalGapSurface+OrigGapSurface[GCT];
     OrigTotalWidth          :=  OrigTotalWidth+GapWidth[GCT];
     DamYL[GCT]              :=  OrigGapDepthAtMSL[GCT];
     NumOfSnaps[GCT]         :=  T(Min(GapWidth[GCT]/15,39)+1);
     if NumOfSnaps[GCT]<>0 then
     SnapWidth[GCT]          :=  GapWidth[GCT]/NumOfSnaps[GCT];
     DamYR[GCT]              :=  OrigGapDepthAtMSL[GCT];
     YDR[GCT]                :=  OrigGapDepthAtMSL[GCT];              {Sill to zero}
     XDRL[GCT]               :=  0;                                 {Toes to extremes}
     XDRR[GCT]               :=  OrigGapWidth[GCT];
     DumpedLeft[GCT]         :=  0;                                {No material dumped}
     DumpedRight[GCT]        :=  0;
     DumpedUp[GCT]           :=  0;
     UpMaxLevel[GCT]         :=  100;
     ErodedLeft[GCT]         :=  0;
     ErodedRight[GCT]        :=  0;
     ErodedUp[GCT]           :=  0;
     TotalEroded[GCT]        :=  0;                                {No material eroded}
     ActualVolume[GCT]       :=  0;
     TotalDumped[GCT]          :=  0;
     ProcessLeft[GCT]        :=  Start;              {Processes left in start phase}
     ProcessRight[GCT]       :=  Start;
     Construction[GCT]       :=  Busy;
     MaxInHole[GCT]          :=  0;
     MaxOutHole[GCT]         :=  0;
     GapMu[GCT]              :=  1;
     SkippedVolumeLeft[GCT]  :=  0;
     SkippedVolumeRight[GCT]:=  0;
     for SC:=0 to NumOfSnaps[GCT] do
   for Counter:=1 to 20 do                    {Restore original bottom protection}
   begin
      InProtY[GCT, SC, Counter]:=0;
      OutProtY[GCT, SC, Counter]:=0;
   end;
     StandLengthProt[GCT]:=ProtSection[GCT]*40;
     DamHeight[GCT]:=DamTop[GCT]-OrigGapDepthAtMSL[GCT];
     for SC:=0 to NumOfSnaps[GCT] do
```

```
      begin
     for Counter:=0 to 20 do InProtY[GCT,SC,Counter]:=0;
     for Counter:=0 to 20 do OutProtY[GCT,SC,Counter]:=0;
     LastProtIn[GCT, SC]    := 20;
     LastProtOut[GCT, SC]   := 20;
     InHoleScour[GCT, SC]   := 0;                          {Reset Scour Processes}
     InHoleDepth[GCT, SC]   := 0;
     OutHoleScour[GCT, SC]  := 0;
     OutHoleDepth[GCT, SC]  := 0;
      end;
  end;
  TotalGapSurface:=OrigTotalGapSurface;

  for GCT:=1 to NumOfGaps do if GCT<>1 then
    StartDrawX[GCT]:=StartDrawX[GCT-1]+10+GapWidth[GCT-1]*WidthFact;

  TidePeriod[1]:=PeriodM2;                          {shift variables for tidal periods}
  TidePeriod[2]:=PeriodS2;
  TidePeriod[3]:=PeriodK1;
  TidePeriod[4]:=PeriodO1;

  DetermineMaxima;                                    {Determine maximum values}

  DataChanged:=False;                                {Process has been initiated}

  Zipped[GCT]:=False;                               {Structure has not collapsed}

  ProcessInited:=True;

  CalculateRequiredVolume(OrigGapWidth);

 end;
end;

Procedure Opt_IntGaps;                              {Set default values for all gaps}
begin
   for GCT:=1 to 3 do
   begin
      begin
      WayOfConstruction[GCT]:=csTrucks;
      WeirSituation[GCT]:=InNormal;
      Construction[GCT]:=Busy;
      OnceCableWay[GCT]:=cmNo;
      OrigGapWidth[GCT]:=300;
      GapWidth[GCT]:=300;
      OrigGapDepthAtMSL[GCT]:=-10;
      OrigGapSurface[GCT]:=3000;
      GapFlow[GCT]:=0;
      Velocity[GCT]:=0;
      GapMu[GCT]:=0;
      DumpedLeft[GCT]:=0;
      DumpedRight[GCT]:=0;
      DumpedUp[GCT]:=0;
      ProcessLeft[GCT]:=Start;
      ProcessRight[GCT]:=Start;
      ActualVolume[GCT]:=0;
      XDRL[GCT]:=000;
      XDRR[GCT]:=000;
      YDR[GCT]:=-10;
      DamYL[GCT]:=000;
      DamYR[GCT]:=000;
      GapDepthAtMSL[GCT]:=-10;
      DamTopWidth[GCT]:=25;
      DamTop[GCT]:=5;
      DamHeight[GCT]:=0;
      DSH[GCT]:=0.3;
      DSS[GCT]:=0.3;
```

```
      DamD50[GCT]:=100;
      GradualLeft[GCT]:=0;
      GradualUp[GCT]:=0;
      UpMaxLevel[GCT]:=100;
      VerticalInAcc[GCT]:=2;
      GradualRight[GCT]:=0;
      StartDrawX[GCT]:=0;
      StandLengthProt[GCT]:=600;
      ProtSection[GCT]:=30;
      SkippedVolumeLeft[GCT]:=0;
      SkippedVolumeRight[GCT]:=0;
      SnapWidth[GCT]:=25;
      MaxOutHole[GCT]:=0;
      MaxInHole[GCT]:=0;
      MaxInHolePlace[GCT]:=0;
      MaxOutHolePlace[GCT]:=0;
      NumOfSnaps[GCT]:=24;
      GapMu[GCT]:=1;
      for SC:=1 to NumOfSnaps[GCT] do
      begin
         InHoleScour[GCT, SC]:=0;
         InHoleDepth[GCT, SC]:=0;
         oInHoleDepth[GCT, SC]:=0;
         OutHoleScour[GCT, SC]:=0;
         OutHoleDepth[GCT, SC]:=0;
         oOutHoleDepth[GCT, SC]:=0;
         LastProtIn[GCT, SC]:=0;
         LastProtOut[GCT, SC]:=0;
         AUUC34[GCT, SC]:=0;
         for Counter:=1 to 20 do
         begin
            InProtY[GCT, SC, Counter]:=0;
            OutProtY[GCT, SC, Counter]:=0;
         end;
      end;
   end;
end;

Procedure Opt_Load(FTL: FileString);                    {Loading a closure situation}

var  InCh: Text;
     ControleString: String[10];

begin
   MCur.SelectHourGlass;
   MessageBar^.ShowText('Now Loading File '+FTL);              {Show Message Bar}
   Opt_IntGaps;
   RememProfLev:=False;
   Assign(InCh, FTL); reset(InCh);
   ReadLn(InCh, ControleString);                  {Check whether it is a right file}
   if ControleString='CTBFILE' then
   begin
   ReadLn(InCh, BasinYear);
   ReadLn(InCh, BasinMonth);
   ReadLn(InCh, BasinDay);
   ReadLn(InCh, BasinHour);
   ReadLn(InCh, BasinMinute);
   ReadLn(InCh, JobDayNo);
   ReadLn(InCh, YearDay);
   ReadLn(InCh, DayS);
   ReadLn(InCh, JobHours);
   ReadLn(InCh, RiverFlow);
   ReadLn(InCh, CulvertSurface);
   ReadLn(InCh, Chezy);
   ReadLn(InCh, BottomD50);
   ReadLn(InCh, MeanSeaLevel);
```

```
ReadLn(InCh, ActSeaLevel);
ReadLn(InCh, OSeal);
ReadLn(InCh, OldBasinLevel);
ReadLn(InCh, BasinLevel);
ReadLn(InCh, TotalFlow);
ReadLn(InCh, LevelFactY);
ReadLn(InCh, LevelDrX);
ReadLn(InCh, NumOfKenterings);
ReadLn(InCh, StormSurge);
ReadLn(InCh, StormPeriod);
ReadLn(InCh, StormChance);
ReadLn(InCh, VesselDraft);
for Counter:=1 to 4 do
begin
   ReadLn(InCh, TideAmp[Counter]);
   ReadLn(InCh, TidePeriod[Counter]);
   ReadLn(InCh, TidePhase0[Counter]);
end;
ReadLn(InCh, TotalAmplitude);
ReadLn(InCh, EquiTotalGapSurface);
ReadLn(InCh, OrigMaxHeight);
ReadLn(InCh, OrigTotalWidth);
ReadLn(InCh, WidthFact);
ReadLn(InCh, FlowDirection);
ReadLn(InCh, OldFlowDirection);
for Counter:=1 to 5 do
begin
   ReadLn(InCh,KomDepth[Counter]);
   ReadLn(InCh,KomSur[Counter]);
end;
ReadLn(InCh, NumOfGaps);
for GCT:=1 to NumOfGaps do
begin
   ReadLn(InCh, WayOfConstruction[GCT]);
   ReadLn(InCh, WeirSituation[GCT]);
   ReadLn(InCh, Construction[GCT]);
   ReadLn(InCh, OnceCableWay[GCT]);
   ReadLn(InCh, OrigGapWidth[GCT]);
   ReadLn(InCh, GapWidth[GCT]);
   ReadLn(InCh, OrigGapDepthAtMSL[GCT]);
   ReadLn(InCh, OrigGapSurface[GCT]);
   ReadLn(InCh, GapFlow[GCT]);
   ReadLn(InCh, Velocity[GCT]);
   ReadLn(InCh, GapMu[GCT]);
   ReadLn(InCh, DumpedLeft[GCT]);
   ReadLn(InCh, DumpedRight[GCT]);
   ReadLn(InCh, DumpedUp[GCT]);
   ReadLn(InCh, ProcessLeft[GCT]);
   ReadLn(InCh, ProcessRight[GCT]);
   ReadLn(InCh, ActualVolume[GCT]);
   ReadLn(InCh, XDRL[GCT]);
   ReadLn(InCh, XDRR[GCT]);
   ReadLn(InCh, YDR[GCT]);
   ReadLn(InCh, DamYL[GCT]);
   ReadLn(InCh, DamYR[GCT]);
   ReadLn(InCh, GapDepthAtMSL[GCT]);
   ReadLn(InCh, DamTopWidth[GCT]);
   ReadLn(InCh, DamTop[GCT]);
   ReadLn(InCh, RequiredVolume[GCT]);
   ReadLn(InCh, DamHeight[GCT]);
   ReadLn(InCh, DSH[GCT]);
   ReadLn(InCh, DSS[GCT]);
   ReadLn(InCh, DamD50[GCT]);
   ReadLn(InCh, GradualLeft[GCT]);
   ReadLn(InCh, GradualUp[GCT]);
   ReadLn(InCh, UpMaxLevel[GCT]);
   ReadLn(InCh, VerticalInAcc[GCT]);
```

```
      ReadLn(InCh, GradualRight[GCT]);
      ReadLn(InCh, StartDrawX[GCT]);
      ReadLn(InCh, StandLengthProt[GCT]);
      ReadLn(InCh, ProtSection[GCT]);
      ReadLn(InCh, ProtQuality[GCT]);
      ReadLn(InCh, SkippedVolumeLeft[GCT]);
      ReadLn(InCh, SkippedVolumeRight[GCT]);
      ReadLn(InCh, SnapWidth[GCT]);
      ReadLn(InCh, MaxOutHole[GCT]);
      ReadLn(InCh, MaxInHole[GCT]);
      ReadLn(InCh, MaxInHolePlace[GCT]);
      ReadLn(InCh, MaxOutHolePlace[GCT]);
      ReadLn(InCh, NumOfSnaps[GCT]);
      for SC:=1 to NumOfSnaps[GCT] do
      begin
     ReadLn(InCh, InHoleScour[GCT, SC]);
     ReadLn(InCh, InHoleDepth[GCT, SC]);
     ReadLn(InCh, oInHoleDepth[GCT, SC]);
     ReadLn(InCh, OutHoleScour[GCT, SC]);
     ReadLn(InCh, OutHoleDepth[GCT, SC]);
     ReadLn(InCh, oOutHoleDepth[GCT, SC]);
     ReadLn(InCh, LastProtIn[GCT, SC]);
     ReadLn(InCh, LastProtOut[GCT, SC]);
     ReadLn(InCh, AUUC34[GCT, SC]);
     for Counter:=1 to 20 do
     begin
        ReadLn(InCh, InProtY[GCT, SC, Counter]);
        ReadLn(InCh, OutProtY[GCT, SC, Counter]);
     end;
     end;
   end;
   ReadLn(InCh, TotalGapSurface);
   ReadLn(InCh, OrigTotalGapSurface);
   ReadLn(InCh, TidalPrism);
   ReadLn(InCh, U02);
   ReadLn(InCh, ReducPerM);
   ReadLn(InCh, TimeSinceKent);
   close(InCh);
   DataChanged:=False;
   Zipped[GCT]:=False;
   MCur.SelectStdCursor;
   Delay(1000);
   MessageBar^.Hide;
   end
   else
   begin
      Dial_ControleFile;
   end;
end;

procedure Opt_Save(FTS: FileString);                    {Save a Closure Situation}

var Outch: Text;

begin
   if DataChanged then Opt_InitProcess;                  {If necessary, the closure}
   MCur.SelectHourGlass;                                {situation is first initiated}
   MessageBar^.ShowText('Now Saving File '+FTS);
   Assign(OutCh, FTS); rewrite(OutCh);
   WriteLn(OutCh, 'CTBFILE');
   WriteLn(OutCh, BasinYear);
   WriteLn(OutCh, BasinMonth);
   WriteLn(OutCh, BasinDay);
   WriteLn(OutCh, BasinHour);
   WriteLn(OutCh, BasinMinute);
   WriteLn(OutCh, JobDayNo:5,                        '  Job Day Number        ');
   WriteLn(OutCh, YearDay,                           '  Day of Year           ');
```

```
WriteLn(OutCh, DayS);
WriteLn(OutCh, JobHours:5:5,                    '  Hours of Process      ');
WriteLn(OutCh, RiverFlow:5:5,                   '  River Flow            ');
WriteLn(OutCh, CulvertSurface:5:5,              '  Culvert Surface       ');
WriteLn(OutCh, Chezy:5:5,                       '  Chezy Value           ');
WriteLn(OutCh, BottomD50:5:5,                   '  BottomD50             ');
WriteLn(OutCh, MeanSeaLevel:5:5,                '  Mean Sea Level        ');
WriteLn(OutCh, ActSeaLevel:5:5,                 '  Actual Sea Level      ');
WriteLn(OutCh, OSeal:5:5,                       '  Old Sea Level         ');
WriteLn(OutCh, OldBasinLevel:5:5,               '  Old Basin Level       ');
WriteLn(OutCh, BasinLevel:5:5,                  '  Actual Basin Level    ');
WriteLn(OutCh, TotalFlow:5:5,                   '  Total Flow throug Gaps');
WriteLn(OutCh, LevelFactY:5:5,                  '  Y factor LevelDrawing ');
WriteLn(OutCh, LevelDrX:5:5,                    '  Coord X Level Drawing ');
WriteLn(OutCh, NumOfKenterings:5,               '  Kenterings Till Now   ');
WriteLn(OutCh, StormSurge:5:5,                  '  Maximum Storm Surge   ');
WriteLn(OutCh, StormPeriod:5:5,                 '  Storm Duration        ');
WriteLn(OutCh, StormChance:5:5,                 '  Chance of Storm       ');
WriteLn(OutCh, VesselDraft:5:5,                 '  Draft of Vessels      ');
for Counter:=1 to 4 do
begin
   WriteLn(OutCh, TideAmp[Counter]:5:5,      '     Tidal Amplitude Factor');
   WriteLn(OutCh, TidePeriod[Counter]:5:5,   '     Tidal Period          ');
   WriteLn(OutCh, TidePhase0[Counter]:5:5,   '     Tidal Phase at T=0    ');
end;
WriteLn(OutCh, TotalAmplitude:5:5,              '  Max Amplitude Possible');
WriteLn(OutCh, EquiTotalGapSurface:5:5,         '  Gap Surf. Equilibrium ');
WriteLn(OutCh, OrigMaxHeight:5:5,               '  Original Maximum Depth');
WriteLn(OutCh, OrigTotalWidth:5:5,              '  Original Total Width  ');
WriteLn(OutCh, WidthFact:5:5,                   '  Drawing Width Factor  ');
WriteLn(OutCh, FlowDirection:5,                 '  Actual Flow Direction ');
WriteLn(OutCh, OldFlowDirection:5,              '  Last Flow Direction   ');
for Counter:=1 to 5 do
begin
   WriteLn(OutCh,KomDepth[Counter]:5:5,      '     Level in Basin        ');
   WriteLn(OutCh,KomSur[Counter]:5:5,        '     with Surface in Basin ');
end;
WriteLn(OutCh, NumOfGaps:5,                     '  Number of Gaps to Close');
for GCT:=1 to NumOfGaps do
begin
   WriteLn(OutCh, WayOfConstruction[GCT]:5,     '  Way of Construction   ');
   WriteLn(OutCh, WeirSituation[GCT]:5,         '  Actual Weir Situation ');
   WriteLn(OutCh, Construction[GCT]:5,          '  Construction Status   ');
   WriteLn(OutCh, OnceCableWay[GCT]:5,          '  Once Cable Way?       ');
   WriteLn(OutCh, OrigGapWidth[GCT]:5:5,        '  Original Width of Gap ');
   WriteLn(OutCh, GapWidth[GCT]:5:5,            '  Actual Width of Gap   ');
   WriteLn(OutCh, OrigGapDepthAtMSL[GCT]:5:5,   '  Original Depth of Gap ');
   WriteLn(OutCh, OrigGapSurface[GCT]:5:5,      '  Original Gap Surface  ');
   WriteLn(OutCh, GapFlow[GCT]:5:5,             '  Actual Gap Flow       ');
   WriteLn(OutCh, Velocity[GCT]:5:5,            '  Actual Velocity in Gap');
   WriteLn(OutCh, GapMu[GCT]:5:5,               '  Contraction in Gap    ');
   WriteLn(OutCh, DumpedLeft[GCT]:5:5,          '  Volume Dumped Left    ');
   WriteLn(OutCh, DumpedRight[GCT]:5:5,         '                  Right ');
   WriteLn(OutCh, DumpedUp[GCT]:5:5,            '                  Vertical');
   WriteLn(OutCh, ProcessLeft[GCT]:5,           '  Process Left          ');
   WriteLn(OutCh, ProcessRight[GCT]:5,          '  Process Right         ');
   WriteLn(OutCh, ActualVolume[GCT]:5:5,        '  Actual Volume of Dam  ');
   WriteLn(OutCh, XDRL[GCT]:5:5,                '  X left of Apron       ');
   WriteLn(OutCh, XDRR[GCT]:5:5,                '  X right of Apron      ');
   WriteLn(OutCh, YDR[GCT]:5:5,                 '  Apron Level           ');
   WriteLn(OutCh, DamYL[GCT]:5:5,               '  Dam Level Left        ');
   WriteLn(OutCh, DamYR[GCT]:5:5,               '  Dam Level Right       ');
   WriteLn(OutCh, GapDepthAtMSL[GCT]:5:5,       '  Actual Gap Depth      ');
   WriteLn(OutCh, DamTopWidth[GCT]:5:5,         '  Top Width of Dam      ');
   WriteLn(OutCh, DamTop[GCT]:5:5,              '  Required Top Level    ');
   WriteLn(OutCh, RequiredVolume[GCT]:5:5,      '  Required Dam Volume   ');
   WriteLn(OutCh, DamHeight[GCT]:5:5,           '  => Required Dam Height');
```

```
      WriteLn(OutCh, DSH[GCT]:5:5,                    '  Slope of Dam Head      ');
      WriteLn(OutCh, DSS[GCT]:5:5,                    '  Slope of Section       ');
      WriteLn(OutCh, DamD50[GCT]:5:5,                 '  D50 of Dam Material    ');
      WriteLn(OutCh, GradualLeft[GCT]:5:5,            '  Speed Closure Left     ');
      WriteLn(OutCh, GradualUp[GCT]:5:5,              '                  Right   ');
      WriteLn(OutCh, UpMaxLevel[GCT]:5:5,             '  Level Limit Vertical   ');
      WriteLn(OutCh, VerticalInAcc[GCT]:5:5,          '  Vertical Inaccuracy    ');
      WriteLn(OutCh, GradualRight[GCT]:5:5,           '                    UpSide ');
      WriteLn(OutCh, StartDrawX[GCT]:5:5,             '  Drawing of Sections    ');
      WriteLn(OutCh, StandLengthProt[GCT]:5:5,        '  Length of Protection   ');
      WriteLn(OutCh, ProtSection[GCT]:5:5,            '  Length of BP section   ');
      WriteLn(OutCh, ProtQuality[GCT],                '  Quality of BP          ');
      WriteLn(OutCh, SkippedVolumeLeft[GCT]:5:5,      '  Lately Skipped Left    ');
      WriteLn(OutCh, SkippedVolumeRight[GCT]:5:5,     '  Lately Skipped Right   ');
      WriteLn(OutCh, SnapWidth[GCT]:5:5,              '  Width between Scour SC');
      WriteLn(OutCh, MaxOutHole[GCT]:5:5,             '  Maximum Scour Outside ');
      WriteLn(OutCh, MaxInHole[GCT]:5:5,              '  Maximum Scour Inside  ');
      WriteLn(OutCh, MaxInHolePlace[GCT]:5:5,         '  Place Mx Scour Inside ');
      WriteLn(OutCh, MaxOutHolePlace[GCT]:5:5,        '  Place Mx Scour Outside');
      WriteLn(OutCh, NumOfSnaps[GCT]:5,               '  Number of Scour Snaps ');
      for SC:=1 to NumOfSnaps[GCT] do
      begin
     WriteLn(OutCh, InHoleScour[GCT,SC]:5:5,   '       Scour Hole Inside ');
     WriteLn(OutCh, InHoleDepth[GCT,SC]:5:5,   '       Reduced Scour     ');
     WriteLn(OutCh, oInHoleDepth[GCT,SC]:5:5,  '       Last Scour Depth  ');
     WriteLn(OutCh, OutHoleScour[GCT,SC]:5:5,  '       Scour Hole Outside');
     WriteLn(OutCh, OutHoleDepth[GCT,SC]:5:5,  '       Reduced Scour     ');
     WriteLn(OutCh, oOutHoleDepth[GCT,SC]:5:5,'       Last Scour Depth  ');
     WriteLn(OutCh, LastProtOut[GCT,SC]:5:5,   '       Last protect   OUT');
     WriteLn(OutCh, LastProtIn[GCT,SC]:5:5,    '       Last protect    IN');
     WriteLn(OutCh, AUUC34[GCT,SC]:5:5,        '       Scour Integral    ');
     for Counter:=1 to 20 do
     begin
        WriteLn(OutCh, InProtY[GCT, SC, Counter]:5:5  '  Protection Level InFl ');
        WriteLn(OutCh, OutProtY[GCT, SC, Counter]:5:5,'  Protection Level OutFl');
     end;
      end;
   end;
   WriteLn(OutCh, TotalGapSurface:5:5,              '  Actual Gap Surface    ');
   WriteLn(OutCh, OrigTotalGapSurface:5:5,          '  Original Total Surface');
   WriteLn(OutCh, TidalPrism:5:5,                   '  Tidal Prism           ');
   WriteLn(OutCh, U02:5:5,                          '  Square Velocity Integr');
   WriteLn(OutCh, ReducPerM:5:5,                    '  Velocity^5 Integral   ');
   WriteLn(OutCh, TimeSinceKent:5:5,                '  Time since Last Kent  ');
   close(OutCh);
   Delay(1000);
   MessageBar^.Hide;
   MCur.SelectStdCursor;
end;

procedure Opt_DepthsWidths;                    {Input of Depths and Widths of gaps}

begin
   P:=True;
   if (ProcessInited=False) and                                  {Ask for confirmation}
      (DialSeriesOn=False)  and
      (SomethingDumped) then P:=Dial_ReallyChange;
   if P then begin
   Case NumOfGaps of                                    {Depends on number of gaps}
      1: Dial_DepthsWidths1(GapWidth, GapDepthAtMSL, GapMu);
      2: Dial_DepthsWidths2(GapWidth, GapDepthAtMSL, GapMu);
      3: Dial_DepthsWidths3(GapWidth, GapDepthAtMSL, GapMu);
   end;
   MaxCheck;
   for GCT:=1 to NumOfGaps do OrigGapWidth[GCT]:=GapWidth[GCT];
   DataChanged:=True; Started:=False; end;
end;
```

```
procedure Opt_Tide;                                          {Input Tidal Information}
begin
   P:=True;
   if (ProcessInited=False) and
      (DialSeriesOn=False) and
      SomethingDumped then P:=Dial_ReallyChange;
   if P then begin
   Dial_Tide(TideAmp, TidePeriod, TidePhase0);
   DataChanged:=True; Started:=False; end;
end;

procedure Opt_Protection;                                {Input bottom protection data}

var Warning: Boolean;

begin
   if NumOfGaps=1 then Dial_Protection1(StandLengthProt);
   if NumOfGaps=2 then Dial_Protection2(StandLengthProt);
   if NumOfGaps=3 then Dial_Protection3(StandLengthProt);
   Warning:=False;
   for GCT:=1 to NumOfGaps do
   begin
      ProtSection[GCT]:=StandLengthProt[GCT]/40;          {Verify bottom protection}
      if (ProtSection[GCT]=0) and (WayOfConstruction[GCT]<>csSandDump) then
     Warning:=True;
   end;

   if Warning then Warn_Protection;
end;

procedure Opt_GradualClose;                                        {Input closure data}

var OldDamD50: RealArr;
    ss: String[40];

begin
   for GCT:=1 to NumOfGaps do OldDamD50[GCT]:=DamD50[GCT];
   repeat
   if NumOfGaps=1 then Dial_GradualClose1(
                     GradualLeft, GradualRight, GradualUp, UpMaxLevel, DamD50);
   if NumOfGaps=2 then Dial_GradualClose2(
                     GradualLeft, GradualRight, GradualUp, UpMaxLevel, DamD50);
   if NumOfGaps=3 then Dial_GradualClose3(
                     GradualLeft, GradualRight, GradualUp, UpMaxLevel, DamD50);
   MaxCheck;
   for GCT:=1 to NumOfGaps do                                              {Verification}
   begin
      if ((ProcessInited=False) and (DamD50[GCT]<>OldDamD50[GCT])) and
     ((DumpedLeft[GCT]<>0)  and (DumpedRight[GCT]<>0)) and
     ((ProcessLeft[GCT]=Start)     or (ProcessRight[GCT]=Start) or
      (ProcessLeft[GCT]=StartToes) or (ProcessRight[GCT]=StartToes)) then
      begin
     Warn_OnlyFullProcess;                              {Only change if process is full}
     DamD50[GCT]:=OldDamD50[GCT];
      end;
      if (DamD50[GCT]<OldDamD50[GCT])
      and ((DumpedLeft[GCT]<>0) or (DumpedRight[GCT]<>0))
      and (DataChanged=False) then
      begin
     Warn_OnlySmallToBig;                             {Only bigger material than before}
     DamD50[GCT]:=OldDamD50[GCT];
      end;
      ss:='';
      Case WayOfConstruction[GCT] of
      csTrucks: if (GradualUp[GCT]<>0) then ss:='trucks, only horizontal';
      csVessels: if (GradualLeft[GCT]<>0) or (GradualRight[GCT]<>0) then
         ss:='vessels, only vertical';
```

```
      csCableWay: if (GradualLeft[GCT]<>0) or (GradualRight[GCT]<>0) then
         ss:='a cable way, only vertical';
      csSandDump: if (GradualUp[GCT]<>0) then
         ss:='sand supplation, only horizontal';
      end;
      if ss<>'' then
      begin
         Warn_MethodAndWay(ss);                              {If user must be warned}
      end;
      CalculateAngles(WayOfConstruction, DSS, DSH);
   end;
   until ss='';
   for GCT:=1 to NumOfGaps do
   begin
      If DumpedLeft[GCT]=0  then DamYL[GCT]:=YDR[GCT];
      if DumpedRight[GCT]=0 then DamYR[GCT]:=YDR[GCT];
   end;
end;

procedure Opt_Dimensions;                                   {Input dam dimensions}

var ODamTop: RealArr;

begin
   for GCT:=1 to NumOfGaps do ODamTopWidth[GCT]:=DamTopWidth[GCT];
   for GCT:=1 to NumOfGaps do ODamTop[GCT]:=DamTop[GCT];
   if NumOfGaps=1 then Dial_Dimensions1(DamTopWidth, DamTop);
   if NumOfGaps=2 then Dial_Dimensions2(DamTopWidth, DamTop);
   if NumOfGaps=3 then Dial_Dimensions3(DamTopWidth, DamTop);
   MaxCheck;
   for GCT:=1 to NumOfGaps do
   begin
      if Started then DamTop[GCT]:=ODamTop[GCT];
      If (WayOfConstruction[GCT]=csCableWay) and (DamTopWidth[GCT]>10) then
      begin
         DamTopWidth[GCT]:=10;
         Warn_CableWayDamTopWidth;                  {Minimum dam width with cableway}
      end;
   end;
end;

procedure Opt_Methods;                                    {Input of construction method}

var VesselsChosen,
    Warning: Boolean;
    OWayOfConstruction: IntArr;

begin
      for GCT:=1 to NumOfGaps do
      OWayOfConstruction[GCT]:=WayOfConstruction[GCT];
      ExitDialSeries:=False;
      DialSeriesOn:=True;
      if NumOfGaps=1 then Dial_Methods1(WayOfConstruction);
      if NumOfGaps=2 then Dial_Methods2(WayOfConstruction);
      if NumOfGaps=3 then Dial_Methods3(WayOfConstruction);
      VesselsChosen:=False;
      for GCT:=1 to NumOfGaps do if WayOfConstruction[GCT]=csVessels then
      VesselsChosen:=True;
      if ExitDialSeries=False then                {If vessels chosen, enter draft}
      if VesselsChosen=True then Dial_VesselDraft(VesselDraft);
      MaxCheck;
      Warning:=False;
      for GCT:=1 to NumOfGaps do
      if (ProtSection[GCT]=0) and (WayOfConstruction[GCT]<>csSandDump) then
         Warning:=True;
      if Warning then Warn_Protection;                {Verification of protection}
   end;
```

```
   case WayOfConstruction[GCT] of                {Reset other construction methods}
    csTrucks: GradualUp[GCT]:=0;
    csVessels: begin GradualLeft[GCT]:=0; GradualRight[GCT]:=0; end;
    csSandDump: GradualUp[GCT]:=0;
    csCableWay: begin GradualLeft[GCT]:=0; GradualRight[GCT]:=0; end;
   end;
   CalculateAngles(WayOfConstruction, DSS, DSH);       {Calculate angles of slopes}
   if Adapting=False then Opt_Dimensions;                        {Dialog box series}
   if SomethingDumped then
   for GCT:=1 to NumOfGaps do
   if (OWayOfConstruction[GCT]=csVessels) or
      (OWayOfConstruction[GCT]=csCableWay) then
   if (2*(DamTop[GCT]-YDR[GCT])/Tan(DSS[GCT])+DamTopWidth[GCT])>
     ODamTopWidth[GCT] then
     begin
        Warn_NextDamTopWidth;                              {Verification width of sill}
        WayOfConstruction[GCT]:=OWayOfConstruction[GCT];
        DamTopWidth[GCT]:=ODamTopWidth[GCT];
        CalculateAngles(WayOfConstruction, DSS, DSH);
     end;
end;

procedure Opt_VesselDraft;                                    {Input of vessel draft}
begin
   Dial_VesselDraft(VesselDraft);
   MaxCheck;
end;

procedure Opt_Angles;                                         {Input of slope angles}
begin
   P:=True;
   if (ProcessInited=False) and
      (DialSeriesOn=False)  and
      SomethingDumped then P:=Dial_ReallyChange;
   if P then begin
   for Counter:=1 to NumOfGaps do                  {Rads to degrees and vice versa}
   begin
      DSH[Counter]:=DSH[Counter]*PiToDeg;
      DSS[Counter]:=DSS[Counter]*PiToDeg;
   end;
   CalculateAngles(WayOfConstruction, DSS, DSH);
   if NumOfGaps=1 then Dial_Angles1(DSS, DSH);
   if NumOfGaps=2 then Dial_Angles2(DSS, DSH);
   if NumOfGaps=3 then Dial_Angles3(DSS, DSH);
   MaxCheck;
   for Counter:=1 to NumOfGaps do
   begin
      DSH[Counter]:=DSH[Counter]*DegToPi;
      DSS[Counter]:=DSS[Counter]*DegToPi;
   end;
   CalculateAngles(WayOfConstruction, DSS, DSH);                  {Calculate angles}
   DataChanged:=True; Started:=False; end;
end;

procedure Opt_Basin;                                {Input data on surface of basin}
begin
   P:=True;
   if (ProcessInited=False) and
      (DialSeriesOn=False)  and
      SomethingDumped then P:=Dial_ReallyChange;
   if P then begin
   Dial_Basin(KomDepth, KomSur);
   MaxCheck;
   DataChanged:=True; Started:=False; end;
end;

procedure Opt_RiverData;                                      {Input river flow data}
```

```
begin
   Dial_RiverData(RiverFlow);
   MaxCheck;
end;

procedure Opt_DefStorm;                                          {Storm defenition}
begin
   Dial_DefStorm(StormSurge, StormPeriod, StormChance);
   MaxCheck;
end;

procedure Opt_Closure;                                  {General information on Closure}

begin
   P:=True;
   if (ProcessInited=False) and
      (DialSeriesOn=False)  and
      SomethingDumped then P:=Dial_ReallyChange;
   ONOG:=NumOfGaps;
   if P then
   begin
      ExitDialSeries:=False;
      DialSeriesOn:=True;
      Dial_Closure(NumOfGaps, Chezy, BottomD50);
      MaxCheck;
      if ExitDialSeries=False then
      if NumOfGaps>ONOG then Opt_AdaptAll;
   end;
end;

procedure Opt_Culvert;                                  {Construction of a spillway}
begin
   Dial_Culvert(CulvertSurface);
   MaxCheck;
end;

procedure Opt_DateTime;                                 {Input of basin date and time}
begin
   Dial_DateTime(BasinYear, BasinMonth, BasinDay, BasinHour, BasinMinute);
end;

procedure Opt_AdaptAll;                                      {Input of a new closure}

begin
   DataChanged:=True;
   ExitDialSeries:=False;
   DialSeriesOn:=True;
   Started:=False;

   if ExitDialSeries=False then Opt_Tide;           {Only go on if no ESC has been}
   if ExitDialSeries=False then Opt_Basin;         {pressed or box has been canceled}
   if ExitDialSeries=False then Opt_RiverData;
   if ExitDialSeries=False then
   begin
      if NumOfGaps=1 then Dial_Protection1(StandLengthProt);
      if NumOfGaps=2 then Dial_Protection2(StandLengthProt);
      if NumOfGaps=3 then Dial_Protection3(StandLengthProt);
   end;
   for GCT:=1 to NumOfGaps do
   begin
      ProtSection[GCT]:=StandLengthProt[GCT]/40;
   end;
   Adapting:=True;
   if ExitDialSeries=False then Opt_Methods;                {Go on with the dialog}
   Adapting:=False;                                                    {box series}
   if ExitDialSeries=False then Opt_GradualClose;
   if ExitDialSeries=False then Opt_DepthsWidths;
```

```
   Adapting:=True;
   if ExitDialSeries=False then Opt_Dimensions;
   Adapting:=False;
   if ExitDialSeries=False then Opt_Angles;
   if ExitDialSeries=False then Opt_Culvert;

   DialSeriesOn:=False;

end;

procedure Opt_Start;                                    {Start the calculation process}

begin
   if Dial_Startup=True then                                          {Startup Dialog Box}
   begin
      Dial_Closure(NumOfGaps, Chezy, BottomD50);
      MaxCheck;
      if (ExitDialSeries=False) then Opt_AdaptAll;     {If required, a new closure}
   end;                                                  {situation can be entered}
   Warning:=False;
   for GCT:=1 to NumOfGaps do
   begin
      if (StandLengthProt[GCT]<>0) then
      if StandLengthProt[GCT] < (DamTopWidth[GCT]+
      2*(DamTop[GCT]-OrigGapDepthAtMSL[GCT])/Tan(DSS[GCT])) then
      begin Warning:=True; StandLengthProt[GCT]:=0; end;
   end;
   if Warning=True then Warn_BotProt;        {Check and warn for bottom protection}
   ONOG:=NumOfGaps;
   if DataChanged=True then                          {If necessary, initiate process}
   begin
      Opt_InitProcess;
   end;
   ProcessInited:=False;
   ActSeaLevel:=SeaLevel(                                  {Calculate actual sealevel}
      MeanSeaLevel, TideAmp, TidePeriod, TidePhase0, JobHours);
end;

end.
```

6.4 The Drawing of Results in the unit SIMDRAWS

The results of the calculations represent the actual situation on the closure site. The user of the program should have a good impression of this situation, in order to undertake the right construction methods.

For the graphical presentation of these results, in the simulation program a special window is opened which contains the following information:

* The *cross section* of the closure gaps, in which depth, width, and dam advance have been shown;
* Information on the *basin level, sea level* and *flow velocity*.

In case the closure is with bigger material, and the bottom has been protected, this information is extended with:

* The *scour hole contours* at the end of both sides of the bottom protection;
* A *top view* of the construction in which the damage of the bottom protection is shown in different colors;
* A *length section* of the construction in which the settlement of the bottom protection is visualized.

The Procedures

The procedures for the drawing of this information have been put apart in the unit *SIMDRAWS*. They are:

Procedure Draw_Levels:	Drawing a graph in which figure the level inside of the basin, at sea, and the velocity in the closure gap. The procedure header communicates the variables *X1, Y1, X2* and *dY*, which indicate the size of the graph window. The variables which are drawn on the screen are *BasinLevel, SeaLevel* and the array *Velocity*.

Procedure Draw_LevelSpace:

This procedure wipes the level drawing area each time this is necessary. It requires the same variables *X1, Y1, X2* and *dY* as the Draw_Levels procedure, to indicate the size of the window.

Procedure Draw_TopView;

This procedure draws a top view of the closure. Independent of its advance, from the beginning the whole required dam structure is drawn. The bottom protection is shown, and with colors the damage of the bottom protection is indicated. Variables needed are:

PX, PY, DX, DY	Indication for window size
StandLengthProt	Length of the protection
NumOfSnaps	Number of protection sections
DamTopWidth	Width of dam top
DamHeight	Required dam height
DSS	Dam side slope angle
InProtY, OutProtY	Protection settlements
ProtSection	Length of protection section

Procedure Draw_Length:

A length section is drawn, in which the required dam section is shown. The bottom protection settlements and the scour holes are indicated with different colors. Again, the window size is characterized by four variables, this time by *X1, Y1, dX, dY.* Other required variables are:

DrawLengthX	Scaling factor
StandLengthProt	Length of the protection
DamTopWidth	Width of dam top
DamHeight	Required dam height
DSS	Dam side slope angle
InProtY, OutProtY	Protection settlements
ProtSection	Length of protection section
LastProtOut	Last protecting section outside
LastProtIn	Last protecting section inside

Procedure Draw_Profile:

This procedure draws a cross section of the closure gap. The contours of the scour holes inside and outside the basin are drawn in green and cyan, colors which coincide with the same locations in other procedures.

The original profile is drawn in dark gray; the actual state of the dam is indicated in white, with top level, slopes and sill level. The sea- and basin level oscilate in light blue and light red. The window size is defined by *X1, Y1, X2* and *dY*. Other information required is stored in the variables:

WidthFact	Scaling factor for the profiles
OrigTotalWidth	Original total width of gaps
OrigMaxHeight	Original greatest gap height
RememProfLev	Whether profiles were drawn
StartDrawX	Starting location for drawings
OrigGapDepth	Original gap depth at MSL
DamYL, DamYR	Location of dam head tops
YDR	Top level of sill
XDRL, XDRR	Location of dam toes

Procedure Draw_Contours: In the drawing of the cross sections, the contours of the scour holes at both sides of the structure are drawn, calling this procedure.

Procedure Overall_Draw: During each time step, this procedure calls the drawing of the *cross section* and the *level information*.

Procedure Overall_SlackDraw: Each time slack water occurs, this procedure calls the drawing of the *scour hole contours,* the *top view*, and the *length section*.

The unit *SIMDRAWS* has been printed below.

```
{*********************************************************}
{        CLOSIM v 1.00  --  GRAPHIC PRESENTATION UNIT      }
{*********************************************************}
{ Containing the drawing procedures for the simulation     }
{ program on the Closure of Tidal Basins.                  }
{ Written in Turbo Pascal 7.0 by Mark Middag, TUDelft, 1994}
{*********************************************************}

unit Simdraws;

interface

uses MyGraph3,                    {Advanced Graphics Library of Turbo Vision Graphics}
     SIMVARS,                                        {Variables of Main Program}
```

```
     SIMTOOLS;                                          {Various Tools of Main Program}

Procedure Draw_Levels(x1,y1,x2,dY: Real);             {Drawing of Levels and Velocity}
Procedure Draw_TopView(px,py,dx,dy: Real);                    {Drawing of Top View}
Procedure Draw_Length(X1,Y1,dX,dY: Real);                {Drawing of Length Section}
Procedure Draw_Profile(x1,y1,x2,dY: Real);                {Drawing of Cross Section}
Procedure Draw_Contours(x1,y1,x2,dY: Real);           {Drawing of Scour Hole Contours}
Procedure Draw_LevelSpace(x1,y1,x2,dY: Real);            {Space for Drawing of Levels}
Procedure Overall_Draw(WX1,WY1,WDX,WDY: Real);            {Drawing of All Information}
Procedure Overall_SlackDraw(WX1,WY1,WDX,WDY: Real);            {Topview and Contours}

implementation

Procedure Draw_LevelSpace(x1,y1,x2,dY: Real);          {Space for Drawing of Levels}

var LevelFactY: Real;                                          {Level Scaling Factor}

begin
   Wipe(WX1,WY1,WX1+WDX,WY1+WDY);                      {Erase Whole Window ViewPort}
   SetColor(White);                                                {Set drawing color}
   LevelDrX:=0;                                                     {Set X coordinate}
   if TotalAmplitude<>0 then                           {Scale Factor Level Drawings}
     LevelFactY:=dY/(TotalAmplitude+StormSurge);          {depends on maximum level}
   Box(X1,Y1-dY-1,X2,Y1+dY+1);                                        {Draw Level Box}
   MoveCurs(X1,Y1+MeanSealevel*LevelFactY);                             {Draw X-Axis}
   DrawTo(X2,Y1+MeanSealevel*LevelFactY);

   For Counter:=1 to T((x2-x1)/5) do                          {Draw Grid for Levels}
     For Counter2:=1 to 10 do
     begin
       Plot(X1+Counter*5-3,Y1+Counter2*dY/10,LightGray);
       Plot(X1+Counter*5-3,Y1-Counter2*dY/10,LightGray);
     end;
   SetColor(White);                                              {Restore Drawing Color}
end;

Procedure Draw_Levels(x1,y1,x2,dY: Real);             {Drawing of Levels and Velocity}

var LevelFactX: Real;                                 {Stretch factor for level graph}
    XDR, YD1, YD2, YD3: Real;

begin
     LevelDrX:=LevelDrX+TimeStep/300;                              {Reset X-Coordinate}
     LevelFactX:=(x2-x1)/TimeStep;
     if TotalAmplitude<>0 then                       {Scale Factor Level Drawings}
        LevelFactY:=dY/(TotalAmplitude+StormSurge);

     XDR:=X1+LevelFactX*LevelDrX;                   {Determine X and Y coordinates}
     YD1:=Y1-ActSeaLevel*LevelFactY;                    {of the level and velocity}
     YD2:=Y1-BasinLevel*LevelFactY;                             {points to be drawn}
     YD3:=Y1-Velocity[1]*dY/10;

     Plot(XDR, YD1, LightBlue);                      {Outside level in light blue}
     Plot(XDR, YD2, LightRed);                        {Inside level in light red}
     Plot(XDR, YD3, LightGreen);                    {Flow velocity in light green}

     if LevelDrX*LevelFactX > (x2-x1-2) then                {If end of drawing area}
     begin                                                     {reached, then erase}
        Draw_LevelSpace(x1,y1,x2,dY);
     end;
end;

Procedure Draw_TopView(px,py,dx,dy: Real);                    {Drawing of Top View of}
                                                                      {Closure sites}
var PixCol:Integer;
    VDist, LeftX: Real;
```

```
begin
   VDist:=Px;
   for GCT:=1 to NumOfGaps do                                    {For all gaps}
   if (StandLengthProt[GCT]>35) then                {If they have bottom protection}
   begin
      LeftX:=VDist;                              {First draw untouched bottom protection}
      SetColor(White);
      FillBox(LeftX+dx, PY+DY*(DamTopWidth[GCT]/2+DamHeight[GCT]/
              Tan(DSS[GCT]))/ProtSection[GCT]+1,
              LeftX+(NumOfSnaps[GCT]+1)*dx,PY+DY*21);
      FillBox(LeftX+dx, PY-DY*(DamTopWidth[GCT]/2+DamHeight[GCT]/
              Tan(DSS[GCT]))/ProtSection[GCT]-1,
              LeftX+(NumOfSnaps[GCT]+1)*dx,PY-DY*21);

      for SC:=1 to NumOfSnaps[GCT] do                {Then, for all width sections}
      begin
         VDist:=VDist+dX;
         for PSC:=1 to 20 do                        {and for all protection sections}
         begin
            if (PY+PSC*DY)>                       {If protection exceeds construction}
            (PY+DY*(DamTopWidth[GCT]/2+DamHeight[GCT]/Tan(DSS[GCT]))/
            ProtSection[GCT]) then
            begin
               if  InProtY[GCT,SC,PSC]<>0  then                {if has been settled}
               begin
                  PixCol:=LightGray;                             {Draw in light gray}
                  if PSC>LastProtIn[GCT,SC]          {if settlement is fatal, then}
                  then PixCol:=DarkGray;                         {draw in  dark gray}
                  SetColor(PixCol);
                  if (PixCol=DarkGray) or (PixCol=LightGray) then
                  FillBox(VDist,PY+PSC*DY,VDist+dx,PY+dy+PSC*DY);
               end;
            end;
            if (PY-PSC*DY)<                                 {for other side the same}
               (PY-DY*(DamTopWidth[GCT]/2+DamHeight[GCT]/Tan(DSS[GCT]))/
               ProtSection[GCT]) then
            begin
               if OutProtY[GCT,SC,PSC]<>0 then
               begin
                  PixCol:=LightGray;
                  if PSC>LastProtOut[GCT,SC] then PixCol:=DarkGray;
                  SetColor(PixCol);
                  if (PixCol=LightGray) or (PixCol=DarkGray) then
                  FillBox(VDist,PY-PSC*DY,VDist+dx,PY-dy-PSC*DY);
               end;
            end;
         end;
      end;
      SetColor(Blue);                                {draw dam slopes in dark blue}
      FillBox(LeftX+dx, PY-DY*(DamTopWidth[GCT]/2+
              DamHeight[GCT]/Tan(DSS[GCT]))/ProtSection[GCT],
              VDist+dx,PY+DY*(DamTopWidth[GCT]/2+
              DamHeight[GCT]/Tan(DSS[GCT]))/ProtSection[GCT]);
      SetColor(LightBlue);                             {draw dam top in light blue}
      FillBox(LeftX+dx, PY-DY*(DamTopWidth[GCT]/2)/ProtSection[GCT],
              VDist+dx,PY+DY*(DamTopWidth[GCT]/2)/ProtSection[GCT]);
      SetColor(LightGray);                          {restore drawing color for text}
      VDist:=VDist+dX*10;
   end;
end;

Procedure Draw_Length(X1,Y1,dX,dY: Real);                    {Length section of dam}
                                                             {and bottom protection}
var InMaxSect, OutMaxSect: Integer;
    LengthFactX,
    InHoleLevel, OutHoleLevel,
    PX1, PX2, PY1, PY2, YDist: Real;
```

```
begin
   YDist:=30;
   InMaxSect:=T(MaxInHolePlace[GCT]);          {in which section the hole is minimum}
   OutMaxSect:=T(MaxOutHolePlace[GCT]);        {in which section the hole is minimum}
   for GCT:=1 to NumOfGaps do
   if StandLengthProt[GCT]<>0 then
   begin
                                                        {Determine levels and factors}
      InHoleLevel:=InHoleDepth[GCT, InMaxSect]+InProtY[GCT,
      InMaxSect, T(LastProtIn[GCT, InMaxSect])];
      OutHoleLevel:=OutHoleDepth[GCT, OutMaxSect]+OutProtY[GCT,
      OutMaxSect, T(LastProtOut[GCT, OutMaxSect])];
      LengthFactX:=0.8*dX/(DamTopWidth[GCT]/2+DamHeight[GCT]/
      Tan(DSS[GCT])+StandLengthProt[GCT]);
      PX1:=DamTopWidth[GCT]/2;
      PY1:=30*dY+GCT*YDist*dY;                          {Set various coordinates}
      PX2:=PX1+DamHeight[GCT]/Tan(DSS[GCT]);
      PY2:=PY1+DamHeight[GCT]*dY;
      Wipe(X1-LengthFactX*(20*ProtSection[GCT]-                    {Wipe old area}
           MaxInHole[GCT]/Tan(Beta)),
           Y1+PY1,
           X1+LengthFactX*(20*ProtSection[GCT]+
           MaxOutHole[GCT]/Tan(Beta)),
           Y1+PY2+dY*Max(OutHoleLevel, InHoleLevel));

      SetColor(LightGray);                                  {Draw dam section}
      SetLineStyle(DottedLn, 0, NormWidth);                  {in dotted lines}
      MoveCurs(X1-PX2*LengthFactX, Y1+PY2);
      DrawTo(X1-PX1*LengthFactX, Y1+PY1);
      DrawTo(X1+PX1*LengthFactX, Y1+PY1);
      DrawTo(X1+PX2*LengthFactX, Y1+PY2);
      SetLineStyle(0, 0, NormWidth);                        {reset line style}

      SetColor(White);                          {Draw bottom protection for inflow}
      MoveCurs(X1+1*ProtSection[GCT]*LengthFactX,
           Y1+(PY2+InProtY[GCT, InMaxSect, 1]*dY));
      for PSC:=1 to T(LastProtIn[GCT, InMaxSect]) do
          DrawTo(X1+PSC*ProtSection[GCT]*LengthFactX,
                 Y1+(PY2+InProtY[GCT, InMaxSect, PSC]*dY));

      if Construction[GCT]<>Collapsed          {Draw maximum scour hole for inflow}
      then begin
         SetColor(Green);                                   {Inflow section in green}
         if StandLengthProt[GCT]=0 then
         begin
         MoveCurs(X1,
             Y1+PY2+dY*InHoleLevel);
         DrawTo(X1+(LastProtIn[GCT, InMaxSect]*ProtSection[GCT]+
             40)*LengthFactX,
             Y1+PY2+dY*InHoleLevel);
         end
         else
         DrawTo(X1+(LastProtIn[GCT, InMaxSect]*ProtSection[GCT]+
             MaxInHole[GCT]/Tan(Beta))*LengthFactX,
             Y1+PY2+dY*InHoleLevel);
      end;
                                              {Draw bottom protection for outflow}
      SetColor(White);
      MoveCurs(X1-1*ProtSection[GCT]*LengthFactX,
           Y1+(PY2+OutProtY[GCT, OutMaxSect, 1]*dY));
      for PSC:=1 to T(LastProtOut[GCT, OutMaxSect]) do
          DrawTo(X1-PSC*ProtSection[GCT]*LengthFactX,
                 Y1+(PY2+OutProtY[GCT, OutMaxSect, PSC]*dY));
                                             {Draw maximum scour hole for outflow}
      if Construction[GCT]<>Collapsed then
      begin
         SetColor(Cyan);
```

```
         if StandLengthProt[GCT]=0 then
         begin
         MoveCurs(X1,
             Y1+(PY2+dY*OutHoleLevel));
         DrawTo(X1-(LastProtOut[GCT, OutMaxSect]*ProtSection[GCT]+
             40)*LengthFactX,
             Y1+(PY2+dY*OutHoleLevel))
         end
         else
         DrawTo(X1-(LastProtOut[GCT, OutMaxSect]*ProtSection[GCT]+
             MaxOutHole[GCT]/Tan(Beta))*LengthFactX,
             Y1+(PY2+dY*OutHoleLevel));
      end;
   end;
end;

procedure Overall_Draw(WX1,WY1,WDX,WDY: Real);                {Draw Levels & Profile}
                                                               {into graphics window}
begin
                                                  {Only if number of gaps did not change}
   if (ONOG=NumOfGaps) then Draw_Levels(WX1+340*WDX/640,WY1+130*WDY/440,
                                        WX1+620*WDX/640,50*WDY/440);
   if (ONOG=NumOfGaps) then Draw_Profile(WX1+40*WDX/640,WY1+100*WDY/440,
                                        WX1+300*WDX/640,40*WDY/440);
end;

procedure Overall_SlackDraw(WX1,WY1,WDX,WDY: Real);          {Draw Length Section,}
                                                            {Top View and Scour Hole}
                                                               {Contours into window}
begin
   if Beta<>0 then                                                        {Error Trap}
   begin
      Draw_Length(WX1+480*WDX/640,WY1+160*WDY/440, 200*WDX/640, 2*WDY/440);
      Draw_TopView(WX1+ 50*WDX/640,WY1+280*WDY/440,
                   ((13.5-NumOfGaps*2)/NumOfGaps)*WDX/640, 3*WDY/440);
      Draw_Contours(WX1+ 40*WDX/640,WY1+100*WDY/440,WX1+300*WDX/640,
                                                              40*WDY/440);
   end;
end;

Procedure Draw_Profile(x1,y1,x2,dY: Real);              {Draw Cross Sections of Gaps}

var xl, DrPrSzX, DrPrSzY, HoleFactY,
    XUpSlL, XUpSlR, OWidthFact: Real;
    OYSea, OYBin: Integer;
    ZipPlaceFact, ZipPlace: Real;

begin

    OWidthFact:=WidthFact;

    if OrigTotalWidth<>0 then                                    {For scale drawings}
       WidthFact:=((X2-X1)-10*NumOfGaps)/OrigTotalWidth;

    if OWidthFact<>WidthFact then SetStartDraws;            {If some width changed,}
                                                                 {redraw section}
    DrPrSzX:=1;
    If OrigMaxHeight=0 then DrPrSzY:=0.4                          {Minimum Scale}
                       else DrPrSzY:=-(OrigMaxHeight/dY);          {Scale factor}

    for GCT:=1 to NumOfGaps do if (RememProfLev=True) then

    if ((T((10*Oseal)/DrPrSzY))<>OYSea) or                    {If something changed}
       (T((10*OBasinLevel)/DrPrSzY)<>OYBin) or
       (Started=False) then

    begin
```

```
SetColor(DarkGray);                                          {Draw Original Profile}
MoveCurs(X1+StartDrawX[GCT]/DrPrSzX,Y1-DamTop[GCT]/DrPrSzY);
DrawTo(X1+StartDrawX[GCT]/DrPrSzX,
   Y1-(OrigGapDepthAtMSL[GCT]/DrPrSzY));
DrawTo(X1+(StartDrawX[GCT]+OrigGapWidth[GCT]*WidthFact)/DrPrSzX,
   Y1-OrigGapDepthAtMSL[GCT]/DrPrSzY);
DrawTo(X1+(StartDrawX[GCT]+OrigGapWidth[GCT]*WidthFact)/DrPrSzX,
Y1-DamTop[GCT]/DrPrSzY);

SetColor(Black);                                          {ReMoveCurs Old Profile}
XUpSlL:=Max(0,oXDRL[GCT]-(oDamYL[GCT]-oYDR[GCT])/Tan(DSH[GCT]));
XUpSlR:=Min(OrigGapWidth[GCT],oXDRR[GCT]+(oDamYR[GCT]-oYDR[GCT])/
                                                  Tan(DSH[GCT]));
MoveCurs(X1+StartDrawX[GCT]/DrPrSzX+0*WidthFact/DrPrSzX,
   Y1-oDamYL[GCT]/DrPrSzY);
DrawTo(X1+StartDrawX[GCT]/DrPrSzX+(XUpSlL)*WidthFact/DrPrSzX,
   Y1-oDAMYL[GCT]/DrPrSzY);
DrawTo(X1+StartDrawX[GCT]/DrPrSzX+oXDRL[GCT]*WidthFact/DrPrSzX,
   Y1-oYDR[GCT]/DrPrSzY);
DrawTo(X1+StartDrawX[GCT]/DrPrSzX+oXDRR[GCT]*WidthFact/DrPrSzX,
   Y1-oYDR[GCT]/DrPrSzY);
DrawTo(X1+StartDrawX[GCT]/DrPrSzX+(XUpSlR)*WidthFact/DrPrSzX,
   Y1-oDAMYR[GCT]/DrPrSzY);
DrawTo(X1+StartDrawX[GCT]/DrPrSzX+OrigGapWidth[GCT]*WidthFact/DrPrSzX,
   Y1-oDAMYR[GCT]/DrPrSzY);

                                                          {Draw New Profile}
SetColor(White);
XUpSlL:=Max(XDRL[GCT]-(DamYL[GCT]-YDR[GCT])/Tan(DSH[GCT]),0);
XUpSlR:=Min(OrigGapWidth[GCT],XDRR[GCT]+(DamYR[GCT]-YDR[GCT])/
                                                  Tan(DSH[GCT]));
MoveCurs(X1+StartDrawX[GCT]/DrPrSzX+0*WidthFact/DrPrSzX,
   Y1-DamYL[GCT]/DrPrSzY);
DrawTo(X1+StartDrawX[GCT]/DrPrSzX+(XUpSlL)*WidthFact/DrPrSzX,
   Y1-DamYL[GCT]/DrPrSzY);
DrawTo(X1+StartDrawX[GCT]/DrPrSzX+XDRL[GCT]*WidthFact/DrPrSzX,
   Y1-YDR[GCT]/DrPrSzY);
if (Zipped=False) then
begin
   DrawTo(X1+StartDrawX[GCT]/DrPrSzX+XDRR[GCT]*WidthFact/DrPrSzX,
      Y1-YDR[GCT]/DrPrSzY)
end
else                                        {If collapse by zipping has occurred}
begin
   ZipPlaceFact:=Random(100);                          {Could occur at any place}
   ZipPlace:=0.008*(XDRL[GCT]*ZipPlaceFact +
                                        XDRR[GCT]*(100-ZipPlaceFact));
   DrawTo(X1+StartDrawX[GCT]/DrPrSzX+(ZipPlace+10)*WidthFact/DrPrSzX,
      Y1-YDR[GCT]/DrPrSzY);
   DrawTo(X1+StartDrawX[GCT]/DrPrSzX+(ZipPlace+20)*WidthFact/DrPrSzX,
      Y1-YDR[GCT]/DrPrSzY+80);
   DrawTo(X1+StartDrawX[GCT]/DrPrSzX+(ZipPlace+30)*WidthFact/DrPrSzX,
      Y1-YDR[GCT]/DrPrSzY);
   DrawTo(X1+StartDrawX[GCT]/DrPrSzX+XDRR[GCT]*WidthFact/DrPrSzX,
      Y1-YDR[GCT]/DrPrSzY);
end;
DrawTo(X1+StartDrawX[GCT]/DrPrSzX+(XUpSlR)*WidthFact/DrPrSzX,
   Y1-DAMYR[GCT]/DrPrSzY);
DrawTo(X1+StartDrawX[GCT]/DrPrSzX+OrigGapWidth[GCT]*WidthFact/DrPrSzX,
   Y1-DAMYR[GCT]/DrPrSzY);

SetColor(Black);
MoveCurs((X1+StartDrawX[1])/DrPrSzX-6,
                        Y1-Oseal/DrPrSzY);              {Remove Last SeaLevel}
DrawTo(X2/DrPrSzX+6,Y1-Oseal/DrPrSzY);
SetColor(LightBlue);
MoveCurs((X1+StartDrawX[1])/DrPrSzX-6,
                    Y1-ActSeaLevel/DrPrSzY);            {Draw Actual SeaLevel}
```

```
      DrawTo(X2/DrPrSzX+6,Y1-ActSeaLevel/DrPrSzY);

      SetColor(Black);                                    {Remove last Basin Level}
      MoveCurs((X1+StartDrawX[1])/DrPrSzX-6,Y1-(OBasinLevel)/DrPrSzY);
      DrawTo(X2/DrPrSzX+6,Y1-(OBasinLevel)/DrPrSzY);
      SetColor(LightRed);                                 {Draw Actual Basin Level}
      MoveCurs((X1+StartDrawX[1])/DrPrSzX-6,Y1-(BasinLevel)/DrPrSzY);
      DrawTo(X2/DrPrSzX+6,Y1-(BasinLevel)/DrPrSzY);
      Setcolor(White);                                                {Reset Drawing
Color}
    end;

    RememProfLev:=True;                          {Set flag for remembering values}
    oseal:=ActSeaLevel;
    oBasinLevel:=BasinLevel;
    OYBin:=T((OBasinLevel)/DrPrSzY);
    OYSea:=T((Oseal)/DrPrSzY);
    TotalWidth:=0;
    for GCT:=1 to NumOfGaps do                                    {Remember old
values}
    begin                                                  {so they can be erased in}
       oYDR[GCT]:=YDR[GCT];                                      {the next time step}
       oXDRL[GCT]:=XDRL[GCT];
       oXDRR[GCT]:=XDRR[GCT];
       oDamYL[GCT]:=DamYL[GCT];
       oDamYR[GCT]:=DamYR[GCT];
       TotalWidth:=TotalWidth+GapWidth[GCT];
    end;
end;

Procedure Draw_Contours(x1,y1,x2,dY: Real);              {Draw Scour Hole Contours}

var DrPrSzX, DrPrSzY, HoleFactY: Real;

begin
    DrPrSzX:=1;                                                      {Scale factors}
    HoleFactY:=-1;

    if OrigTotalWidth<>0 then                                    {For scale drawings}
       WidthFact:=((X2-X1)-10*NumOfGaps)/OrigTotalWidth;

    If OrigMaxHeight=0 then DrPrSzY:=0.4                             {Minimum scale}
                       else DrPrSzY:=-(OrigMaxHeight/dY);

    for GCT:=1 to NumOfGaps do                                          {for all gaps}
    if (StandLengthProt[GCT]<>0) then                  {If the bottom is protected}
    begin
      if RememContours=True then                     {If they have been drawn already}
      for SC:=1 to NumOfSnaps[GCT] do
      begin
        SetColor(Black);                          {ReMove old Scour Contour for Inflow}
        MoveCurs(X1+Round((StartDrawX[GCT]+(SC-1)*
                                    SnapWidth[GCT]*WidthFact)/DrPrSzX),
           Y1-Round((OrigGapDepthAtMSL[GCT]/DrPrSzY)+
                                  oInHoleDepth[GCT, SC-1]*HoleFactY));
        DrawTo(X1+Round((StartDrawX[GCT]+
                                 (SC-0)*SnapWidth[GCT]*WidthFact)/DrPrSzX),
           Y1-Round((OrigGapDepthAtMSL[GCT]/DrPrSzY)+
                                   oInHoleDepth[GCT, SC]*HoleFactY));

        SetColor(Black);                           {ReMove old Scour Contour for Outflow}
        MoveCurs(X1+Round((StartDrawX[GCT]+
                                 (SC-1)*SnapWidth[GCT]*WidthFact)/DrPrSzX),
           Y1-Round((OrigGapDepthAtMSL[GCT]/DrPrSzY)+
                                 oOutHoleDepth[GCT, SC-1]*HoleFactY));
        DrawTo(X1+Round((StartDrawX[GCT]+
                                 (SC-0)*SnapWidth[GCT]*WidthFact)/DrPrSzX),
```

```
          Y1-Round((OrigGapDepthAtMSL[GCT]/DrPrSzY)+
                                        oOutHoleDepth[GCT, SC]*HoleFactY));

        SetColor(Cyan);                                  {Draw New Contour for Outflow}
        MoveCurs(X1+Round((StartDrawX[GCT]+
           (SC-1)*SnapWidth[GCT]*WidthFact)/DrPrSzX),
           Y1-Round((OrigGapDepthAtMSL[GCT]/DrPrSzY)+
           (OutHoleDepth[GCT, SC-1]+OutProtY[GCT,SC-1,
           T(LastProtOut[GCT,SC-1])])*HoleFactY));
        DrawTo(X1+Round((StartDrawX[GCT]+
           (SC-0)*SnapWidth[GCT]*WidthFact)/DrPrSzX),
           Y1-Round((OrigGapDepthAtMSL[GCT]/DrPrSzY)+
           (OutHoleDepth[GCT, SC]+OutProtY[GCT,SC,
           T(LastProtOut[GCT,SC])])*HoleFactY));

        SetColor(Green);                                  {Draw New Contour for Inflow}
        MoveCurs(X1+Round((StartDrawX[GCT]+
           (SC-1)*SnapWidth[GCT]*WidthFact)/DrPrSzX),
           Y1-Round((OrigGapDepthAtMSL[GCT]/DrPrSzY)+
           (InHoleDepth[GCT, SC-1]+InProtY[GCT,SC-1,
           T(LastProtIn[GCT,SC-1])])*HoleFactY));
        DrawTo(X1+Round((StartDrawX[GCT]+
           (SC-0)*SnapWidth[GCT]*WidthFact)/DrPrSzX),
           Y1-Round((OrigGapDepthAtMSL[GCT]/DrPrSzY)+
           (InHoleDepth[GCT, SC]+InProtY[GCT,SC,
           T(LastProtIn[GCT,SC])])*HoleFactY));

      end;
      for SC:=0 to NumOfSnaps[GCT] do                          {Remember Old Values}
      begin
        oInHoleDepth[GCT, SC]:=
        InHoleDepth[GCT, SC]+InProtY[GCT,SC,T(LastProtIn[GCT,SC])];
        oOutHoleDepth[GCT, SC]:=
        OutHoleDepth[GCT, SC]+OutProtY[GCT,SC,T(LastProtOut[GCT,SC])];
      end;
    end;
    RememContours:=True;                              {Set Flag for Remembered Values}
end;

end.
```

6.4 *The Toolbox Unit*

Various procedures for all units of the simulation program are so general, that they are put apart in one *toolbox unit* so they can be called by all other procedures.

The procedures which have been written in the unit *SIMTOOLS:*

Function N(Number: Real): String:	Converts real *Number* into string *N*.
Function I(Number: Real): String:	Converts integer *Number* into string *I*.
Function T(Number: Real): Integer	Converts real *Number* into integer *T* (also called truncate).
Function Sgn(Number: Real): Integer:	Returns sign *Sgn* of argument *Number*: * -1 for negative numbers; * 0 for zero; * 1 for positive numbers.
Function Power(x,a: Real): Real	Returns the *Power a* of *x* (this is x^a)
Function Max(a,b: Real): Real	Returns the maximum value *Max* of the arguments *a* and *b*.
Function Min(a,b: Real): Real	Returns the minimum value *Min* of the arguments *a* and *b*.
Function Tan(x: Real): Real	Returns the *Tan* function of *x*.
Function ArcSin(x: Real): Real	Returns the *ArcSin* function of *x*.
function Log(A: Real): Real	Returns the *Log* function of *x*.
Function SeaLevel(MeanSeaLevel: Real; TideAmp, TidePeriod, TidePhase0: RealArr; JobHours: Real): Real	Calculates the sea level *SeaLevel* at a certain time according to the tidal conditions.
Function MouseX: Integer	Mouse X coordinate, *MouseX*.

Function MouseY: Integer	Gives the Mouse Y coordinate, *MouseY.*
Function MouseK: Integer	Mouse key status (pressed left, right, none), returned in *MouseK.*
Function MousePressed: Boolean	Detects whether mouse has been pressed *(MousePressed=True)* or not *(=False)*
Function MouseStabile: Boolean	Detects whether mouse is moved or buttons are pressed (if not, *MouseStabile=True)*
Function Value(S: String): Real	Returns the numerical *Value* of the string *S.*
Procedure WaitMouse	Waits till the mouse key has been pressed and released.
Procedure Opt_ToggleSound	Toggles sound *on* or *off.*
Procedure Beep	Produces a short *beep.*
Procedure MultiBeep	Produces a series of short *beeps.*
Procedure EverySecond	Programmer's use for interrupts and window handling.
Procedure Opt_Quit	Quit program and go back to DOS.
procedure CalculateRequiredVolume(Width: RealArr)	Calculates the required volume of material for a dam with the actual dimensions.
Procedure StartPermission(var Ers: String;; var Perm: Boolean)	When the calculation process is started, the data on the closure is checked before. This prevents the program from errors when wrong data has been entered (e.g. a material size of 0 mm); also, it can prevent irrealistic figures (e.g. when the user tries to dump big material with a pipeline)
Procedure SetStartDraws	Resets the drawing variables for another closure situation.
Procedure NoActivities	Stops all construction activities.

Procedure Wipe(x1,y1,x2,y2: Real)	Erases an area *(x1, y1, x2, y2)* of the screen.
Procedure Tempor_Interr_Calc	Temporary interrupts the calculation process (e.g. when the user is asked for input in a dialog box).
Procedure Tempor_Resume_Calc	If the calculations were interrupted with *Tempor_Interr_Calc*, they can be resumed by calling this procedure.

Drawing Procedures

The procedures below have been written, in order to extend the drawing procedures of Pascal 7.0 itself. The coordinates are now automaticly scaled to the monitor type. The variables which are passed to the drawing procedures, now can be *real* as well as *integer*. This is useful, because the results of the calculations are often floating point variables, which normally in Turbo Pascal have to be truncated before they can be drawn on the screen.

Procedure Plot(x1,y1,Color: real)	Extended procedure *PutPixel*.
Procedure DrawTo(x1,y1: Real)	Extended procedure *LineTo*.
Procedure MoveCurs(x1,y1: Real)	Extended procedure *Move*.
Procedure DrawLine(x1,y1,x2,y2: Real)	Extended procedure *Line*.
Procedure TextXY(x1,y1: Real; S: String)	Extended procedure *OutTextXY*.
Procedure Box(x1,y1,x2,y2: Real)	Extended procedure *Rectangle*.
Procedure FillBox(x1,y1,x2,y2: Real)	Extended procedure *Bar*.
Procedure DataTextXY(*X , Y: Integer* *S: String* *Data: Real)*	Extended procedure *OutTextXY*, which now also can write numerical data to the screen.

The Pascal listing of this unit has been printed below.

```
{*********************************************************}
{                CLOSIM v 1.00  --  TOOLBOX UNIT          }
{*********************************************************}
{ Containing various tools for the simulation program on  }
{ the Closure of Tidal Basins, written in Turbo Pascal 7.0 }
{ and Turbo Vision Graphics by Mark Middag, TUDelft, 1994. }
{*********************************************************}

Unit SimTools;

Interface

uses CRT, DOS, Memory,                                          {Dos Units}
     GObjects, GDrivers,                                {Turbo Vision Graphics}
     MCursor2, GMENU6,
     GViews,  GDialogs,
     GMsgBox, GStdDlg,
     GApp, GColors, GWindow,
     BMPDrvr, GBut,
     MyGraph3, GraphPrc,                          {Extended Graphics Procedures}

     SIMVARS;                              {Simulation Program Variables and Constants}


function  N(Nummer: Real): String;                          {Converts Real into String}
function  I(Nummer: Real): String;                       {Converts Integer into String}
function  T(Nummer: Real): Integer;                        {Converts Real into Integer}
function  Sgn(Nummer: Real): Integer;                {Indicates Sign of Number (-1,0,1)}
function  Power(x,a: Real): Real;                                {Normal Power Function}
function  Max(a,b: Real): Real;                    {Gives the maximum of two numbers}
Function  Min(a,b: Real): Real;                    {Gives the minimum of two numbers}
function  Tan(x: Real): Real;                                                {Tan Function}
function  ArcSin(x: Real): Real;                                         {Arc Sin Function}
function  Log(A: Real): Real;                                               {Log Function}
function  SeaLevel(MeanSeaLevel: Real;                        {Calculation of Sea Level}
                   TideAmp,
                   TidePeriod,
                   TidePhase0: RealArr;
                   JobHours: Real): Real;
function  MouseX: Integer;                                          {Mouse X Coordinate}
function  MouseY: Integer;                                          {Mouse Y Coordinate}
function  MouseK: Integer;                                            {Mouse Key Status}
function  MousePressed: Boolean;                                  {Whether Mouse Pressed}
function  MouseStabile: Boolean;                                  {Whether Mouse Touched}
Function  Value(S: String): Real;                          {Numerical Value of a String}
procedure WaitMouse;                                              {Wait for Mouse Event}
procedure Opt_ToggleSound;                                         {Toggle Sound On/Off}
procedure Beep;                                                       {Produce one Beep}
procedure MultiBeep;                                             {Produce Series of Beeps}
procedure EverySecond;                                             {Programmer's Use Only}
procedure Opt_Quit;                                                           {Quit to Dos}
procedure Plot(x1,y1,Color: real);                                {Extended Plot Function}
procedure DrawTo(x1,y1: Real);                                    {Extended Draw Function}
procedure MoveCurs(x1,y1: Real);                                  {Extended Move Function}
procedure DrawLine(x1,y1,x2,y2: Real);                            {Extended Line Function}
procedure TextXY(x1,y1: Real; S: String);                 {Extended OutTextXY Function}
procedure Wipe(x1,y1,x2,y2: Real);                                                {Wipe Area}
procedure Box(x1,y1,x2,y2: Real);                                                  {Draw Box}
procedure FillBox(x1,y1,x2,y2: Real);                                              {Fill Area}
procedure CalculateRequiredVolume(Width: RealArr);                          {Required Volume}
```

```
Procedure DataTextXY(X, Y: Integer; S: String; Data: Real);          {Text Numbers}
Procedure StartPermission(                                  {Check whether the Process}
          Var Ers: String; var Perm: Boolean);                        {can be started}
Procedure SetStartDraws;                                          {Reset Draw Variables}
Procedure NoActivities;                                  {Stop All Construction Activities}
Procedure Tempor_Interr_Calc;                      {Temporary Interruption of Calculations}
Procedure Tempor_Resume_Calc;                        {Continue Calculations if needed}

const NoB = 1;                                                          {Button Status}
      BothB = 2;
      RightB= 3;
      LeftB = 4;

var MX, MY: Real;                                          {Scaling Factor for Drawings}

Implementation                                          {Definition of written procedures}


procedure Tempor_Interr_Calc;                      {Temporary interruption of calculations}
begin
   OStarted:=Started; Started:=False;
end;

procedure Tempor_Resume_Calc;                          {continue calculations if needed}
begin
   Started:=OStarted;
end;

function N(Nummer: Real): String;                    {Convert real variables into strings}

Var s:String[20];

begin
   Str(Nummer:0:2, S);
   if Nummer*10=Round(Nummer*10) then Str(Nummer:0:1, S);
   if Nummer=Round(Nummer) then Str(Nummer:0:0, S);
   N:=S;
end;

function I(Nummer: Real): String;                  {Convert integer variables into strings}

var s: String[25];

begin
   Str(Nummer:0:0, S);
   I:=S;
end;


function T(Nummer: Real): Integer;                   {Convert integer variables into reals}

var N: Integer;

begin
   N:=Round(Nummer);
   T:=N;
end;

function Sgn(Nummer: Real): Integer;                    {Return sign of variable (-1,0,1)}

var s: Integer;

begin
   if nummer<0 then s:=-1;
   if nummer>0 then s:=1;
   if nummer=0 then s:=0;
```

```
   Sgn:=S;
end;

Function Value(S: String): Real;                    {Return Numerical Value of String}

var N: Real;
    Control: Integer;

begin

   Val(S, N, Control);
   Value:=N;

end;

Function SeaLevel(MeanSeaLevel: Real;                              {Calculate Sea Level}
                  TideAmp,
                  TidePeriod,
                  TidePhase0: RealArr;
                  JobHours: Real): Real;

var phase, SL: real;

begin

  SL:=MeanSeaLevel;
  for Counter:=1 to 4 do                  {Add Actual Value of All Tidal Compontents}
  begin
      if TidePeriod[Counter]<>0 then
      begin
         Phase:=((JobHours*3600 + TidePhase0[Counter]) / TidePeriod[Counter]);
         SL:= SL + TideAmp[Counter]*sin(2*pi*phase);
      end;
  end;
  if TimeToStorm>0 then                                            {Add Storm Setup}
  begin
     if StormChance=1 then StormChance:=0;
     ActStormSurge:=Abs(1-Cos(2*PI*(1-Abs(TimeToStorm/(StormPeriod*3600)))))
                    *StormSurge/2;
     SL:=SL+ActStormSurge;
     TimeToStorm:=TimeToStorm-TimeStep;
     if TimeToStorm<0 then TimeToStorm:=0;
     if TimeToStorm=0 then ActStormSurge:=0;
  end;
  SeaLevel:=SL;
end;

Function Power(x, a: Real): Real;                  {Return power of argument}

var xb, ab, pow: Real;

begin
    xb:=abs(x);
    ab:=abs(a);
    if xb<>0 then pow:=exp(ab*ln(xb))
             else pow:=0;
    if a<0 then pow:=1/pow;
    power:=pow;

end;

function Log(A: Real): Real;                       {Return logarithm of argument}

begin
   if A>0 then Log:=ln(A)/Ln(10)
          else Log:=0;
end;
```

```
Function Max(a,b: Real): Real;                          {Return Maximum of two variables}

begin
   if a>b then Max:=A
          else Max:=B;
end;

Function Min(a,b: Real): Real;                          {Return Minimum of two variables}

begin
   if a<b then Min:=A
          else Min:=B;
end;

Function Tan(x: Real): Real;                                    {Return Tan of Argument}

var Tn: Real;

begin

    if Cos(x)<>0 then Tn:=Sin(X)/Cos(X)
                 else Begin WriteLn('Tan Zero Impossible'); Halt; end;

    Tan:=Tn;

end;

procedure NoActivities;                                 {Stop All Construction Activities}

begin
   for GCT:=1 to NumOfGaps do
   begin
         GradualLeft[GCT]:=0; GradualRight[GCT]:=0; GradualUp[GCT]:=0;
   end;
end;

function ArcSin(x: Real): Real;                                 {Return Arc Sin of Argument}

begin

    ArcSin := ArcTan (x/SqRt (1-Sqr(x)));

end;

Procedure WaitMouse;                                    {Wait until mouse has been pressed}

begin

    repeat until mousepressed;
    repeat until mousepressed=false;

end;

Function MouseStabile: Boolean;                 {Whether Mouse is being Touched or Not}

begin
    if MouseX*MouseY = MouseSum then MouseStabile:=True
                               else MouseStabile:=False;

    MouseSum:=MouseX*MouseY;
end;

procedure Beep;                                                         {Produce short beep}

begin
   for Counter:=0 to 10 do
   begin
```

```
      Sound(800+Counter*50); Delay(5); NoSound; Delay(2);
   end;
end;

procedure MultiBeep;                                          {Produce Series of Beeps}

begin
   Beep; Beep; Beep; Beep;
   Beep; Beep; Beep; Beep;
end;

procedure Opt_ToggleSound;                                      {Toggle Sound On/Off}
begin
  if SoundOn=True  then SoundOn:=False
                   else SoundOn:=True;
end;

Procedure EverySecond;                                {Programmer's Use, Window Management}

begin
  if mousek=2 then delay(500);
  if mousek=3 then begin halt; end;
  if (DrawCounter/40)=Int(DrawCounter/40) then
     begin
        if SomethingDrawn=False then OverallOpen:=False;
        if SomethingWritten=False then DataBoxOpen:=False;
     end;
  Inc(DrawCounter);
  if (DrawCounter/40)=Int(DrawCounter/40) then
     begin SomethingDrawn:=False; SomethingWritten:=False end;

end;

procedure Opt_Quit;                                           {Exit Program, Go to Dos}
begin
   Halt;
end;

Procedure Get_Mouse_Action                                                {Mouse Status}
          (var But: Integer; var Hor,Ver: integer);
var
  Reg: registers;
begin
    with Reg do
    begin
        Ax := 3;
        Intr($33,Reg);
        Hor := Cx div 8;
        Ver := Dx div 8;
              {$B+}
        If ((Bx and $1) <> $1)  and  ((Bx and $2) <> $2) then
        begin
            But := NoB;
            exit;
        end;
        If ((Bx and $1) = $1)  and   ((Bx and $2) = $2) then
           But := BothB
        else
        begin
            If (Bx and $1) = $1 then
               But := LeftB
            else
               But := RightB;
        end;
              {$B-}
    end;
end;
```

```
Function Mousek;                                              {Mouse Key Status}

var a,b,c: integer;

begin
   Get_Mouse_Action(a,b,c);
   if a = NoB then mousek:=0;
   if a = leftb then mousek:=1;
   if a = rightb then mousek:=2;
   if a = bothb then mousek:=3;
end;

Function MousePressed;                          {Whether Mouse Key Pressed or Not}

begin
   if Mousek>0 then MousePressed:=True
               else MousePressed:=False;
end;

Function Mousex;                                              {Mouse X Coordinate}

var a,b,c: integer;

begin
   Get_Mouse_Action(a,b,c);
   Mousex:=b;
end;


Function Mousey;                                              {Mouse Y Coordinate}

var a,b,c: integer;

begin
   Get_Mouse_Action(a,b,c);
   Mousey:=c;
end;

                                {The Graphics Functions below have been extended;}
                                    {they now are scaled according to the monitor}
                                  {type and can deal with real variables as well.}

procedure Plot(x1,y1,Color: real);                            {Extended Plot Function}
begin
  MY:=GetMaxY/480; MX:=GetMaxX/640;
  PutPixel(Round(X1*MX),Round(Y1*MY), Round(Color));
end;

procedure DrawTo(x1,y1: Real);                                {Extended Draw Function}
begin
  MY:=GetMaxY/480; MX:=GetMaxX/640;
  LineTo(Round(X1*MX),Round(Y1*MY));
end;

procedure MoveCurs(x1,y1: Real);                              {Extended Move Function}
begin
  MY:=GetMaxY/480; MX:=GetMaxX/640;
  MoveTo(Round(X1*MX),Round(Y1*MY));
end;

procedure DrawLine(x1,y1,x2,y2: Real);                        {Extended Line Function}
begin
  MY:=GetMaxY/480; MX:=GetMaxX/640;
  MoveTo(Round(X1*MX),Round(Y1*MY));
  LineTo(Round(X2*MX),Round(Y2*MY));
end;
```

```
procedure Wipe(x1,y1,x2,y2: Real);                              {Erase Area of Screen}
begin
  MY:=GetMaxY/480; MX:=GetMaxX/640;
  SetFillStyle(SolidFill,0);
  Bar(Round(X1*MX),Round(Y1*MY),Round(X2*MX),Round(Y2*MY));
end;

procedure TextXY(x1,y1: Real; S: String);                    {Extended OutTextXY function}
begin
  SetTextStyle(DefaultFont, HorizDir, 1);
  SetTextJustify(LeftText, CenterText);
  MY:=GetMaxY/480; MX:=GetMaxX/640;
  Wipe(X1*Mx-04, Y1*My-8, X1*MX+100, Y1*My+8);
  OutTextXY(Round(X1*MX),Round(Y1*MY),S);
end;

Procedure DataTextXY(X, Y: Integer;                          {Numerical Data to OutTextXY}
                     S: String; Data: Real);
begin
   Wipe(X-4,Y-8, X+180, Y+8);
   TextXY(X,Y, S + N(Data));
end;

procedure Box(x1,y1,x2,y2: Real);                                                {Draw Box}
begin
  MY:=GetMaxY/480; MX:=GetMaxX/640;
  Rectangle(Round(X1*MX),Round(Y1*MY),Round(X2*MX),Round(Y2*MY));
end;

procedure FillBox(x1,y1,x2,y2: Real);                              {Fill Area on Screen}

var T: Integer;

begin
   for T:=Round(X1) to Round(X2) do DrawLine(t,y1,t,y2);
end;

procedure CalculateRequiredVolume(Width: RealArr);          {Required Volume of Dam}

var H: Real;                                                              {Height variable}

begin
   for GCT:=1 to NumOfGaps do                                  {calculation for all gaps}
   begin
      H:=DamTop[GCT]-YDR[GCT];                            {Height still to be constructed}
      if Tan(DSS[GCT])<>0 then                                  {Not to be divided by zero}
      RequiredVolume[GCT]:=                          {Volume which is still to be dumped}
           Width[GCT]*(                                                  {Length of dam}
           (DamTopWidth[GCT]*H)+                                  {Rectangular centre}
           (H/Tan(DSS[GCT]))*H);                                                 {Slope}
      RequiredVolume[GCT]:=                                {In case the angle is changed,}
      RequiredVolume[GCT]+                              {material can have been dumped before}
      ActualVolume[GCT];
   end;
end;

procedure SetStartDraws;                                    {Reset all Drawing Variables}
begin
    for GCT:=1 to NumOfGaps do
    if GCT<>1 then
       StartDrawX[GCT]:=StartDrawX[GCT-1]+10+GapWidth[GCT-1]*WidthFact
             else
       StartDrawX[GCT]:=10;
end;

procedure StartPermission                                   {Check All Data before start}
          (Var Ers: String; var Perm: Boolean);
```

```
begin
    Perm:=True;                                  {If nothing wrong, permission is given}
    Ers:='';
    for GCT:=1 to NumOfGaps do
    begin
       if RequiredVolume[GCT]=0 then Ers:='dimensions';        {If Something wrong,}
       if (GapWidth[GCT]<=0) then Ers:='gap width';               {no permission is}
       if (DamTop[GCT]<=0)  then Ers:='dam top';                        {given and a}
       if (Tan(DSH[GCT])=0) then Ers:='dam head angle';           {warning appears.}
       if (Tan(DSS[GCT])=0) then Ers:='dam side angle';
       if (DamD50[GCT]<=0.15)  then Ers:='dam material';
    end;
    if Ers<>'' then Perm:=False;
end;

end.
```

6.6 The Dialog Unit

Within the user interface of *Turbo Vision Graphics*, the communication with the program user takes place through dialog- and message boxes. These boxes can be designed by an external program, a *dialog box designer*, which produces complex pascal files.

In the simulation program, a lot of dialog boxes have been used. A summary of them is written below. The procedure names ending with a number 1, also exist for the cases of two and three gaps.

Dialog Boxes for Data Input

Procedure Dial_RiverData(*var RiverFlow: Real)*	Asks for a river flow into the basin. The variable *RiverFlow* is adapted.
Procedure Dial_DateTime(*var YE, MO, DA,* *HO, MI: String)*	In this dialog box, the date and time of the closure are changed. The time variables *YE, MO, DA, HO, MI,* which are strings, are adapted.
Procedure Dial_Tide(*var TideAmp,* *TidePhase0: RealArr)*	The four components of the tidal conditions can be changed here. The arrays *TideAmp* and *TidePhase0* are entered.
Procedure Dial_Dimensions1(*var DamTopWidth,* *DamTop: RealArr)*	This dialog box asks for the dam top width and the dam top level of the dam to be built. The variables are arrays.
Procedure Dial_Methods1(*var WayOfConstruction: IntArr)*	Here the method of construction is chosen. The variable is the integer array *WayOfConstruction* which can be *csTrucks, csVessels, csCableWay* or *csSandDump*.
Procedure Dial_Angles1(*var DSS, DSH: RealArr)*	Change the head and side angle of the dam to be built. The variables *DSS* and *DSH* are expressed in radials.

Procedure Dial_GradualClose1(
var GradualLeft,
GradualRight,
GradualUp,
UpMaxLevel,
DamD50: RealArr);

Change the productions from left, right or upside (the arrays *GradualLeft, GradualRight* and *GradualUp).* A maximum sill level can be entered *(UpMaxLevel),* as well as the particle size of the material *(DamD50).*

Procedure Dial_Basin(
var KomDepth,
KomSur: RealArr)

Of the function of the basin storage surface, five interpolation points are asked. They are entered in the two arrays *KomDepth* and *KomSur.*

Procedure Dial_Closure(
var NumOfGaps: Integer;
var Chezy, BottomD50: Real);

General information on the closure: Number of gaps *(NumOfGaps),* Chezy value *(Chezy)* and size of bottom material particles *(BottomD50).*

Procedure Dial_DefStorm(
var StormSurge,
StormPeriod,
StormChance: Real)

This dialog box is for definition of a storm, giving it's maximum surge, the duration, and the probability. The used variables are floating point.

Procedure Dial_Culvert(
var CulvertSurface: Real)

A culvert can be 'constructed' entering it's flow surface, stored in the variable *CulvertSurface.*

Procedure Dial_Protection1(
var LengthProt: RealArr:
ProtQuality: IntArr)

The bottom protection is defined as the length and quality of it. The length is a real value between 30 and 2000 meters, while the quality of it can be *Good* or *Bad.*

Procedure Dial_DepthsWidths1(
var GapWidth,
GapDepthAtMSL,
GapMu: RealArr)

The depth *(GapDepthAtMSL),* the width *(GapWidth)* and the contraction of each of the gaps is entered here.

Procedure Dial_VesselDraft(
var VesselDraft: Real)

The maximum draft of the vessels is stored in the real variable *VesselDraft.*

Procedure Dial_Logfiles(

Here, the user can decide which

var ErosLog, *VeloLog,* *LevelLog,* *GapLog: Boolean)*	results have to be written to disk, each time step. He can choose of *production and erosion, velocities, levels* or the *gap dimensions.*

Program Warnings

Procedure Warn_Material(*MaterialSize,* *GapNo: Integer)*	A warning for a material error (*MaterialSize* zero or less) in one of the gaps *(GapNo).*
Procedure Warn_OnlyFullProcess	Warning: in case of horizontal closure material sizes can only be adapted when the top level has been reached.
Procedure Warn_OnlySmallToBig	When the material size is changed in process, the material size chosen has to be bigger than the actual size.
Procedure Warn_Protection	Warning: this way of construction should not be applied without a bottom protection.
Procedure Warn_Zipped	Warning: the construction has collapsed by a dumping inaccuracy.
Procedure Warn_VesselDraft	Warning: dumping cannot be done from vessels with this draft.
Procedure Warn_MethodAndWay(ss: String)	Warning: With this grain size, the chosen construction method cannot be applied. The variable *ss* is used to communicate to the user.
Procedure Warn_CaissonClosure	Warning: Caisson closure has not been implemented in the program yet.
Procedure Warn_SkippedNoChange	Warning: The material sizes cannot be changed as long as there is material skipped (for an explanation, see manual volume 1)

Procedure Warn_HelpOn	Help has been put on.
Function Dial_ReallyChange: Boolean	Dialog warning, asking whether the user really wants to make this change.

Other Program Message Boxes

Procedure HelpBox(S: String)	Dialog box for the help text *S*.
Function Dial_FileSelect(*Title: String;* *Mask, Try:PathStr;* *var FT: PathStr): Boolean*	File selector with title *Title*. Within *Mask* and proposing file *Try*, a file named *FT* is returned if the selector has not been canceled.
Procedure Dial_Intro	Introduction text box of the program.
Function Dial_Startup: Boolean	At the start of a calculation, a dialog box with information is given and the user is asked whether he wants to change data.

The structure of the procedures, needed for these boxes, is extensive and complex. Therefore, the whole listing of the dialog unit (in which all procedures have the same appearance) has not been reproduced here. The Pascal listing of the unit header, one dialog box (*Dial_Closure)* and one message box *(Dial_VesselDraft)* can be sufficient:

```
{*********************************************************}
{              CLOSIM v 1.00  --  DIALOG BOX UNIT         }
{*********************************************************}
{ Containing the Dialog boxes of the Turbo Vision Graphics }
{ User Interface for the simulation program on the Closure }
{ of Tidal Basins. Mark Middag, TUDelft, 1994.            }
{*********************************************************}

Unit GSimDial;

Interface

uses CRT, DOS, Memory,                                  {Dos and TP Units}
     MyGraph3,                                         {Extended Graphics}
     GObjects, GDrivers,                             {Turbo Vision Graphics}
     MCursor2, GMENU6,
     GViews,  GDialogs, GMsgBox, GStdDlg,
     GApp, GColors, GWindow,
     BMPDrvr, GBut,
```

```
     SIMVARS,                                              {Variables and Constants}
     SIMTOOLS;                                                    {Toolbox Unit}

procedure Dial_RiverData(var RiverFlow: Real);                          {River Flow}
procedure Dial_DateTime(var YE, MO, DA, HO, MI: String);       {Basin Date and Time}
procedure Dial_Tide(var TideAmp, TidePeriod, TidePhase0: RealArr);       {Tidal data}
procedure Dial_Dimensions1(var DamTopWidth, DamTop: RealArr);        {Dam Dimensions}
procedure Dial_Dimensions2(var DamTopWidth, DamTop: RealArr);        {Dam Dimensions}
procedure Dial_Dimensions3(var DamTopWidth, DamTop: RealArr);        {Dam Dimensions}
procedure Dial_Methods1(var WayOfConstruction: IntArr);         {Head and Side angle}
procedure Dial_Methods2(var WayOfConstruction: IntArr);         {Head and Side angle}
procedure Dial_Methods3(var WayOfConstruction: IntArr);         {Head and Side angle}
procedure Dial_Angles1(var DSS, DSH: RealArr);                  {Head and Side angle}
procedure Dial_Angles2(var DSS, DSH: RealArr);                  {Head and Side angle}
procedure Dial_Angles3(var DSS, DSH: RealArr);                  {Head and Side angle}
procedure Dial_GradualClose1(var GradualLeft, GradualRight,              {Production}
  GradualUp, UpMaxLevel, DamD50: RealArr);
procedure Dial_GradualClose2(var GradualLeft, GradualRight,              {Production}
                  GradualUp, UpMaxLevel, DamD50: RealArr);
procedure Dial_GradualClose3(var GradualLeft, GradualRight,              {Production}
  GradualUp, UpMaxLevel, DamD50: RealArr);
procedure Dial_Basin(var KomDepth, KomSur: RealArr);     {Basin storage information}
procedure Dial_Closure(var NumOfGaps: Integer;
      var Chezy, BottomD50: Real);                  {General information on closure}
procedure Dial_DefStorm(
  var StormSurge, StormPeriod, StormChance: Real);                {Storm definition}
procedure Dial_Culvert(var CulvertSurface: Real);                  {Culvert surface}
procedure Dial_Protection1(var LengthProt: RealArr);             {Bottom protection}
procedure Dial_Protection2(var LengthProt: RealArr);             {Bottom protection}
procedure Dial_Protection3(var LengthProt: RealArr);             {Bottom protection}
procedure Dial_DepthsWidths1(
            var GapWidth, GapDepthAtMSL, GapMu: RealArr);          {Depth and Width}
procedure Dial_DepthsWidths2(
            var GapWidth, GapDepthAtMSL, GapMu: RealArr);          {Depth and Width}
procedure Dial_DepthsWidths3(
            var GapWidth, GapDepthAtMSL, GapMu: RealArr);          {Depth and Width}
procedure Dial_ControleFile;
procedure Dial_VesselDraft(Var VesselDraft: Real);         {draft of dumping vessels}
procedure Dial_Logfiles(
 var ErosLog, VeloLog, LevelLog, GapLog: Boolean);          {Which logfiles to write?}
procedure Warn_Material(MaterialSize, GapNo: Integer);  {Warning for Material Size}
procedure Warn_OnlyFullProcess;                      {Warning for Early Process Change}
procedure Warn_OnlySmallToBig;                        {If smaller particles are chosen}
procedure Warn_Protection;                              {Warning for bottom protection}
procedure Warn_Zipped;                             {Warning: wait until nothing skipped}
procedure Warn_VesselDraft;                          {Warning: depth for vessels needed}
procedure Warn_MethodAndWay(ss: String);               {Warning for Construction Method}
procedure Warn_CaissonClosure;                    {Caisson Closure: Not Implemented Yet}
procedure Warn_SkippedNoChange;                             {Warning for Skipped Volumes}
procedure Warn_HelpOn;                                    {Warning: Help has been put on}
procedure HelpBox(S: String);                            {Empty box for help information}
function Dial_ReallyChange: Boolean;                   {Warning for Change of Parameters}
function  Dial_FileSelect(Title: String; Mask, Try:
               PathStr; var FT: PathStr): Boolean;     {Turbo Vision File Selector}
procedure Dial_Intro;                                              {Intro Explanation}
Function  Dial_Startup: Boolean;                                  {Startup Explanation}

implementation

var Dialog : PDialog;
    B : PView;
    R : TRect;
    Control : Word;
    Warning: Boolean;
    U: String;

type
```

```
    RecString=String[45];

Procedure Dial_Closure(                                    {Dialog box with general}
  var NumOfGaps:Integer;
{information}
  var Chezy,
      BottomD50: Real);

type
   RecClosureData = record                                 {Input in dialog box occurs}
     SNUM, SCH, SBD50: RecString;                                {in temporal record}
   end;

var ClosureData: RecClosureData;

begin
   DialY1:=2;                                                        {Top of Dialog Box}
   DialY2:=25;                                                    {Bottom of Dialog Box}
   MakePlace(9, DialY1, 76, DialY2);                                {Reserve View Space}
   Dialog:=New(PDialog,                                              {Insert Dialog Box}
   Init(R,'General Information on Closure'));                         {Dialog Box Title}
   Tempor_Interr_Calc;                                          {Interrupt Calculations}
   ExitDialSeries:=False;                          {Reset Exit Flag for Series of boxes}
   with Dialog^ do                                                {Within dialog box, do:}
   begin
      MakePlace(2, 2, 63, 4);                                {Reserve relative view space}
      Insert(New(PgStaticText, Init(R,                                {for comment text:}
 'Enter the number of gaps and the bottom conditions.',
  DefaultOpts)));

      MakePlace(35,5,45,6);                                             {Reserve space}
      B:=New(PgInputLine, Init(R,45,DefaultOpts));                    {for input field}
      Insert(B);
      MakePlace(5,5,35,6);
      Insert(New(PgLabel, Init(R, '~N~umber of Gaps: ',                        {Add label}
       B, DefaultOpts)));

      MakePlace(35,7,45,8);                                             {Reserve space}
      B:=New(PgInputLine, Init(R,45,DefaultOpts));                    {for input field}
      Insert(B);
      MakePlace(5,7,35,8);
      Insert(New(PgLabel, Init(R, '~C~hezy Value:',                            {Add Label}
       B, DefaultOpts)));

      MakePlace(35,9,45,10);                                            {Reserve Space}
      B:=New(PgInputLine, Init(R,45,DefaultOpts));                    {for input field}
      Insert(B);
      MakePlace(5,9,35,10);
      Insert(New(PgLabel, Init(R, '~D~50 Bottom Material:',                    {Add Label}
       B, DefaultOpts)));

      MakePlace(44,DialY2-5,54,DialY2-3);                            {Reserve space for}
      Insert(New(PButton,                                            {Cancel button and}
      Init(R, '~C~ancel',cmCancel, bfDefault)));                             {insert it;}
      MakePlace(55,DialY2-5,65,DialY2-3);                            {Reserve space for}
      Insert(New(PButton,                                                {OK button and}
      Init(R, '~O~K',cmOK, bfDefault)));                                     {insert it}
   end;

   with ClosureData do
   begin
      SNUM:=N(NumOfGaps);                                  {Put data to be changed}
      SCh:=N(Chezy);                                                 {into record}
      SBD50:=N(BottomD50);
   end;

   Dialog^.SetData(ClosureData);                          {Put record into Dialog Box}
```

```
   Control:=DeskTop^.ExecView(Dialog);                    {Execute Defined Dialog Box}

   if Control<>cmCancel                                                {If not canceled,}
   then begin
      Dialog^.GetData(ClosureData);                    {Put data dialog box into Record}
      with ClosureData do
      begin
 val(SNUM,NumOfGaps,control);                                {Retrieve changed variables}
 val(SCh,Chezy,control);                                                    {from record}
 val(SBD50, BottomD50, control);
      end;
   end
   else ExitDialSeries:=True;                          {If canceled, in a series of boxes}
                                                      {the entire series should be stopped}

   Tempor_Resume_Calc;                                    {Resume calculations if needed}

end;

procedure Warn_VesselDraft;                              {Warning: Construction by vessels}
                                                                 {cannot continue at this depth}
begin
      Tempor_Interr_Calc;                           {Interrupt calculations during dialog}

      MessageBox(                                                    {MessageBox Procedure}
         'construction by vessels is not'#13+
         'possible because of their draft.'+#13+
         'Choose another method.'#13,                                          {Warning box,}
 nil, mfWarning or mfOKButton);                              {containing one OK button}

      Tempor_Resume_Calc;                                {Resume interrupted calculations}
end;
```

6.7 The Help Unit

The source of the help unit of the program is quite simple. When a user has indicated that he requires help, a help flag is set. In the handling of the events, whenever this help flag is detected as set, the chosen procedure is not called, but the the event is passed to a procedure named *HelpOnTopic(Topic)*.

This help procedure has been written below. The procedure contains a text for each *Topic*; after that, a dialog box is shown in which the text is printed. After the user has read this information, the event and the help flag are cleared.

```
{*********************************************************}
{             CLOSIM v 1.00  --  HELP FILE UNIT           }
{*********************************************************}
{ Containing the help files for the simulation program on  }
{ the Closure of Tidal Basins; written in Turbo Pascal 7.0 }
{ and Turbo Vision Graphics by Mark Middag, TUDelft, 1994. }
{*********************************************************}

Unit SIMHELP;

interface

uses CRT, DOS, Memory,                                    {Dos and TP Units}
     MyGraph3,                                          {Extended Graphics}
     GObjects, GDrivers,                              {Turbo Vision Graphics}
     MCursor2, GMENU6,
     GViews,  GDialogs,
     GMsgBox, GStdDlg,
     GApp, GColors, GWindow,
     BMPDrvr, GBut,

     GSIMDIAL;                                             {Dialog Box Unit}

procedure Help_On(Topic: Integer);

var s1,s2,                                               {Help Line Variables}
    s3,s4,
    s5,s6,
    s7,s8: string[70];

implementation

procedure Help_On(Topic: Integer);                          {Help on Topic}

begin
    s1:='The help files for this option';                 {Reset Help Lines}
    s2:='are not yet available.';
    s3:=''; s4:='';
    s5:=''; s6:='';
    s7:=''; s8:='';

    case Topic of                                    {Set help lines for the}
                                                            {chosen topic}
    cmSetColors
    begin
    s1:='You can select your own set of colors for the desktop.';
    s2:='';
```

```
s3:='At any time, you can save the actual desktop using the';
s4:='option Save Desktop in the drop down menu.';

s6:='If you want to start the program with your new set of';
s7:='colors, rename the desktop file to SIMFREE.DSK.';
end;
cmOverallWindow
:begin
s1:='The overall window shows in a graphical way';
s2:='general information on the closure.';

s4:='In the upper left corner a cross section of the';
s5:='closure is shown. At the right side, a graph is';
s6:='made, showing information of inside and outside';
s7:='levels and the velocity. A top view of the';
s8:='closure and a length section are shown beneath.';
end;
cmDataBoxWindow
:begin
S1:='The data window shows numerical information on';
S2:='the closure.';

S4:='For the explanation of the values you are referred';
S5:='to the hydraulic manual of the program.';
end;
cmOptionsSave:
begin
s1:='You can save the desktop using this option.';
s2:='';
s3:='If you want the program to start up with your new';
s4:='desktop, rename it to SIMFREE.DSK';
end;
cmOptionsLoad:
begin
s1:='Load your saved desktop using this option.';
s2:='';
end;
cmAbout
:begin
s1:='The about-box gives you information on the';
s2:='program name and version.';
end;
cmOpen
:begin
s1:='Load a closure situation with this option.';
s2:='';
s3:='At any time you want, you can save a closure';
s4:='situation, and retrieve it again with this';
s5:='option. ';
end;
cmSave
:begin
s1:='With this option a closure situation is saved, using';
s2:='the name by which it lately was retrieved or saved.';
s4:='Saving of closure situation can be done at any time.';
end;
cmSaveAs
:begin
s1:='Save a closure situation with this option.';
s2:='';
s3:='At any time you want you can save a closure';
s4:='situation. With this option you are first asked';
s5:='to give a file name.';
end;
cmQuit:
begin
s1:='With this option you quit the program and return to';
```

```
s2:='the DOS prompt.';
end;
cmDosShell:
begin
s1:='With this option you can temporally go to the DOS prompt;';
s2:='you can return to the program typing EXIT.';
end;
cmDefault
:begin
s1:='Load a default situation with this option.';
s2:='';
s3:='The program starts up with a default situation.';
s4:='Applying this option, you can get back to this';
s5:='situation at any time. You can change the';
s6:='default closure situation by saving a closure';
s7:='as DEFAULT.CTB.';
end;
cmInitiateProcess
:begin
s1:='Init Process resets the actual closure back to';
s2:='its equilibrium. All closure-related variables';
s3:='are set back to their original value.';
end;
cmStart
:begin
s1:='You can start the process using this topic.';
s2:='';
s3:='For this, you can also use the stop watch button.';
s4:='The process can be started and stopped at any';
s5:='time. If a change is made, the closure gap is';
s6:='automatically reset to its equilibrium.';
end;
cmInterr
:begin
s1:='This option, linked to the stop watch button as';
s2:='well, enables you to interrupt the process at';
s3:='any time. It can be restarted using the same';
s4:='button, or the START option of the main menu.';
end;
cmProcessToggle
:begin
s1:='The stop watch button starts and interrupts the';
s2:='process at any time.';
end;
cmToggleSound
:begin
s1:='This option enables you to put the sound off and';
s2:='on, just as you wish.';
end;
cmDateTime
:begin
s1:='The construction job has it''s own virtual date';
s2:='and time.';
s4:='This option enables you to change them.';
end;
cmBasin
:begin
s1:='A higher basin level enlarges the storage surface.';
s3:='You can enter five data points between which the';
s4:='level-storage relation can be interpolated.';
end;
cmClosure
:begin
s1:='Here you can enter general data on the closure.';
s2:='';
end;
cmDepthsWidths
```

```
:begin
s1:='Enter the width and the depth of the gap.';
s2:='If the equilibrium profile is calculated, the';
s3:='depth is ajusted.';
end;
cmTide
:begin
s1:='Enter the tidal information with this option.';
s2:='';
s3:='The tide, as used in the program, exists of';
s4:='four components (M2, S2, O1, K1), of which the';
s5:='amplitude and phase at T=0 are entered.';
end;
cmRiverData
:begin
s1:='Data with respect to a river is simply entered';
s2:='as an average river flow.';
s4:='Remember, when there is a river flow, you should';
s5:='construct a culvert as a spill-out.';
end;
cmDimensions
:begin
s1:='You can enter the dam dimensions (dam top heigth and';
s2:='width), as well as the construction method for each of';
s3:='the dams.';
s5:='Remember, that this cannot be changed during the';
s6:='construction process.';
end;
cmAngles
:begin
s1:='You can enter the angles (side angle in cross';
s2:='sections or head angle).';

s4:='Remember, the angle is only optional if you are';
s5:='dumping from vessels. Otherwise, the angle you enter';
s6:='is ajusted to the natural slope, according to the';
s7:='dumping method and the particle sizes.';
end;
cmProtection
:begin
s1:='A stone dam is supposed to be constructed in the';
s2:='middle of the protection you can enter here.';
s4:='The protection can have a good either bad quality;';
s5:='good protection can stand settlements, bad can not.';
end;
cmGradualClose
:begin
s1:='The data with regard to the activities can be entered';
s2:='at any time. The productions are asked for, which should';
s3:='be entered according to the production method. A vertical';
s4:='construction always has a dumping inaccuracy of at least';
s5:='two times the grain diameter. Entering a maximum level,';
s6:='a sill can be constructed. The last parameter to enter';
s7:='is the grain diameter D50 of the dumping material.';
end;
cmDefStorm
:begin
s1:='You can define a storm, entering four caracteristics:';
s2:='';
s3:='     1. The maximum storm surge (m);';
s4:='     2. The duration of the storm (h);';
s5:='     3. The maximum wave height (m);';
s6:='     4. The probability of a storm (1/day).';
end;
cmStartStorm
:begin
s1:='Use this option (or the storm button) to set the ';
```

```
    s2:='probability of the storm temporarily to 1.';
    s3:='Because of this, a storm will occur right after';
    s4:='having chosen this option.';
    end;
    cmCaissonClose
    :begin
    s1:='A caisson closure isn''t possible yet to construct.';
    s2:='';
    s3:='It might be implemented into later versions';
    s4:='of the program.';
    end;
    cmCulvert
    :begin
    s1:='When there is a river flow, a culvert should be';
    s2:='constructed as a spill-out. The flow surface should be';
    s03:='big enough.';
    end;
    cmHelpOnTopics
    :begin
    s1:='You are just using this option to obtain help.';
    s2:='';
    end;
  end;
  HelpBox(S1+^M+                                    {Execute procedure HelpBox from}
          S2+^M+                                      {GSIMDIAL with the help lines}
          S3+^M+                                           {as defined by the topic}
          S4+^M+
          S5+^M+
          S6+^M+
          S7+^M+
          S8);
  HelpRequired:=False;                                          {Put off help status}
  end;
end.
```

7. Adapting, Extending and Compiling the Program

7.1 Working at the Pascal Source Code

To make changes in the Pascal program, the source code should be adapted and be compiled with the *Turbo Pascal* compiler. Because of the *Turbo Vision Graphics* environment which has been used for the simulation program, some extra files are needed in addition to the standard version of this compiler, to effectuate this compilation. These unit files can also be found on the Programmer's disk. A complete index of the files needed, is written below. All units mentioned, should have the corresponding file with extension *TPU* on disk.

---

1. Turbo Pascal: version 6.0 or higher

Specific units to be available:	*CRT*	*(standard dos routines)*
	DOS	*(standard dos routines)*
	GRAPH	*(standard graphics routines)*
	MEMORY	*(standard memory routines)*

2. Turbo Vision Graphics: version 1.5 or higher

Specific units to be available:	*MYGRAPH3*	*(advanced graphics routines)*
	GOBJECTS	*(object routines)*
	GDRIVERS	*(driver routines)*
	GVIEWS	*(view routines)*
	GMENU6	*(various menu routines)*
	GMSGBOX	*(message box routines)*

	GDIALOGS	*(dialog box routines)*
	GSTDDLG	*(standard dialog boxes)*
	GAPP	*(Turbo Vision application)*
	GCOLORS	*(color procedures)*
	GWINDOW	*(window procedures)*
	GBUT	*(button procedures)*
	BMPDRVR	*(bitmap driver)*
3. Units of the Simulation Program:	*CLOSIM.PAS*	*(main program shell)*
	SIMVARS.PAS	*(variables and constants)*
	GSIMOPTS.PAS	*(menu options)*
	GSIMDIAL.PAS	*(dialog boxes)*
	SIMCALC.PAS	*(hydraulic calculations)*
	SIMTOOLS.PAS	*(various program tools)*
	SIMDRAWS.PAS	*(drawing procedures)*
	SIMHELP.PAS	*(program help files)*
4. Other files needed for the compilation:	*ICLOAD.OBJ*	*(load button)*
	ICSAVE.OBJ	*(save button)*
	ICTOGGLE.OBJ	*(toggle process button)*
	ICQUIT.OBJ	*(quit button)*
	ICNEWCL.OBJ	*(button for new closure)*
	ICGRAD.OBJ	*(button for activities)*
	ICSHELL.OBJ	*(button for dos shell)*
	ICWINDOW.OBJ	*(button for graph window)*
	ICDATABOX.OBJ	*(button for databox)*
	ICSTORM.OBJ	*(storm button)*
5. Files needed to run the program:	*OPENING.BMP*	*(bitmap opening screen)*
	DEFAULT.CTB	*(one default closure)*
	SIMFREE.DSK	*(an empty desktop)*

Once compilated the program *CLOSIM.PAS*, the changes in the source code will be implemented and a new executable *CLOSIM.EXE* will be produced. It is important, allways to make a backup copy of the programmer's disk, and not adapt the original files.

7.2 Adding a Calculation Feature

When a new calculation feature is required, a procedure should be added to the file *SIMCALC.PAS*. Do not forget to make a declaration of this new procedure before the

implementation of it, in the header of the unit. The name of the procedure then should be added to the loop in the procedure *Calculate*. It is important to consider the variables used in the new procedure, and have them *initiated* in the procedure *Opt_InitProcess*, and *saved* in the procedure *Opt_Save*.

If all these things have been done accurately, the calculation procedure should be tested, and the results implemented in the graphical and numerical presentation.

7.3 Adding a Drawing Feature

All drawing procedures are object-oriented, and belong to a view. This view is updated from the main program shell *CLOSIM* in the procedure *DrawOverall*, which calls several subroutines from the unit *SIMDRAWS*. Dependent of which aspect is wished to be added to these subroutines, the feature has to be added both to the main shell as to this unit *SIMDRAWS*.

When a new feature is added as a procedure to the unit *SIMDRAWS*, again it must be declared in the header of this unit. The reader should carefully watch the other drawing procedures, in order to understand properly how the graphical information is dealt with. Programming in views is rather difficult; the coordinate systems are relative and the handling of the view is outside of the programmer's influence. Therefore, the *Turbo Vision Programmer's Guide* and the *Turbo Pascal Programmer's Reference* should be counseled thoroughly. This, in combination with the procedures which have already been implemented in the program, offers the possibility to be able to adapt and extend the graphical presentation of the simulation program.

Some things are useful to be mentioned. The *Overall* window of the program is scaled to its required size. Everything drawn in this window, will scale as well. When the *DataBox* window size is changed though, the contents are not scaled, but come outside the scope of the window. Both windows can be used for graphical and numerical information; but because the numerical information cannot be scaled, this should only be applied in the *DataBox* window. Because of these aspects, the coordinate organisation of both windows is quite different. The storing of those coordinate systems (which vary during the running of the program) into records should be understood well before the drawing procedures are adapted or extended.

7.4 Adding a Menu Option

The adaptation and extention of the main menu is easy, but should be effectuated exactly as described below. The example is given for the addition of an option named *Option* to the drop down menu of the program. This option should show a message box on the

screen. If the reader tries to follow these guidelines and succeeds in adding this message box to the program, he should be able to extend the drop down menu with any option. Between brackets, the name of the Pascal source file and the comment which has to be looked for (using the search function of the Pascal editor) have been shown.

---

1. *Add a constant to the constant list. This constant should have the name cmOption and may have any value between 0 and 255 which has not been used yet.* *[SIMVARS: menu constants]*

2. *Add the procedure Option to the procedure list. (Add comments!)* *[CLOSIM: declaration of procedures]*

3. *Add the event cmOption to the event list. It should point to Option, and thus have the following form:*
   --- cmOption: Option; --- *[CLOSIM: TProgram.HandleEvent]*

4. *Add the procedure Option to the main program. This should have the form:* *[like CLOSIM: Construct Spillway]*

```
Procedure TSimApp.Option;
begin
  Opt_Option;
end;
```

5. *Add the procedure Opt_Option to the Option unit. This option should have the following form:* *[like GSIMOPT: Opt_Culvert]*

```
Procedure Opt_Option;
begin
        MessageBox('This is your new option',
                        nil, mfInformation or mfOKButton);
end;
```

6. *Add the new option to the menu. Hereto, the way the other menu options have been implemented should be studied.* *[CLOSIM: InitMenuBar]*

When this has been done, the program can be compiled and run, and a new option will figure in the main drop down menu.

If in one of the other menus (the status line or the button bar) a new option is required, just replace the sixth item of the description, and do the same thing to *InitStatusLine* or *InitToolBar*. To add a new button, follow the guideline below:

---

1. *Draw a 32 x 32 bytes button in bitmap format, *.BMP, with for example the Windows PaintBrush tool. Save it as IC*.BMP (at the asterisk should be your own button's name).*
   Example: ICBUT.BMP.

2. *Run the program BMP2BMG from the programmer's disk, adding the name of the bitmap button you just made. In this case, you should enter:*

   BMP2BMG ICBUT.BMP

3. *Convert this binary file to an OBJ file, using the DOS program BINOBJ. In this example, you should enter:*

   BINOBJ ICBUT.BMG ICBUT ICBUT

4. *You could now delete the BMG file. Do not delete the BMP file, in case you later want to adapt the button.*

5. *Now add the button to the program. Link it (see [CLOSIM: external;] for an example) and adapt the toolbar [CLOSIM: number of buttons].*

6. *Now run the program, instantly save the new desktop and change the name of this desktop (SIM.DSK) to SIMFREE.DSK.*

6.5 Adding a Dialog Box

The dialog boxes have a rather complex structure. The use of dialog editors can be recommanded, but up till this point these dialog boxes have been designed by hand. When reference is made to a dialog box, the main variables which could be changed with it should be mentioned in the dialog box header. The principle of the dialog boxes is, that the variables to adapt are shifted into a record. The variables in this record have the same *type* as the variables to be adapted. With this record, the dialog is effectuated. Afterwards, the variables from the record are shifted into the original variables again.

If a new dialog box is required, it is useful to copy an existing box (e.g. *Dial_Riverflow*) and adapt this to the requirements. This provides a possibility to get to understand the structure of these boxes.

7.6 Adding a Help Function

When a new option is added to the menu, or when new calculation or drawing features have been implemented, it could be wishful to add an explanation about its characteristics to the help file of the program. Hereto, open the unit *SIMHELP.PAS* and look how the other events have been implemented. Just add the new event and the seven possible strings which describe it, and help for this function will be available.

CLOSIM v 1.0

A Simulation Program on the Closure of Tidal Basins

Mark N.K. Middag
August 1994

PART III

Program User's Manual

The program Closim v 1.0 has been written
in fulfillment of the requirements for a Master's Degree from the
University of Technology of Delft, The Netherlands
in cooperation with the Dutch Ministry of Public Works

Professor: Prof. Ir. K. d'Angremond

Supervisors:
Ing. G.A. Beaufort
Dr. Ir. J.A. Cser
Ir. F.C. van Roode
Ir. G.J. Schiereck

Preface

This is the *user's manual* of the simulation program on the closure of tidal basins *CLOSIM v 1.0*. The program was written in fulfilment of the requirement for a Master's degree in Hydraulic Engineering at the University of Technology in Delft.

With this manual, anyone with a slight idea of the closure practice should be able to simulate the closure of a tidal inlet. Some examples of closures which can be found on the *user's disk*, have been commented in chapter *6*.

For the description of the computer program source code, the reader is referred to volume *II* of this paper. Volume *I* contains the hydraulic details of the simulations in *CLOSIM v 1.0*.

Mark N.K. Middag,

July 1994.

Contents

1. Introduction

In this manual, the simulation program on the closure of tidal basins *CLOSIM v 1.0* has been described. In the following chapter, the user of the program is introduced in the closure practice. Some theoretical backgrounds have been given. A more extensive description of the used techniques can be found in the first paper of this manual.

Chapter *3* deals about the scope of the simulation program. Various aspects of the simulations could be extended in future versions.

Chapter *4* tells the user how to install and run the program. Chapter *5* introduces the user in the main menu, by discussing the main features and options. In chapter *6* three closure examples are executed with the program.

Appendix *A* displays the contents of the data files; appendix *B* for the log files. In appendix *C*, a quick user reference for the program is provided.

2. The Closure of Tidal Basins

2.1 Introduction

In this chapter, a brief introduction on the closure of tidal basins is given. The main aim of this introduction is to accompany the simulation program *CLOSIM* and provide sufficient information on the various aspects of closures which the program covers.

The information given in this chapter will be rather global. A more detailed description of the applied calculation techniques can be found in the chapters *3* to *6* of the description of the hydraulic aspects of the simulation program, *volume I* of this paper. For a detailed technical overview the reader is referred to the existing, scarce literature with respect to the subject, which has been listed in *appendix F* of *volume I*. To that appendix, all references in this chapter refer.

2.2 Tidal Basins

A *tidal basin* can simply be defined as any basin in connection with the sea. As the tide can freely enter and leave the basin, a stable situation will exist in which the tidal variations, the basin area and the bottom material determine the flow area of the branch between basin and sea. It is possible that the estuary has a river inflow. A schematisation of such a tidal basin can be seen in figure *2.1*:

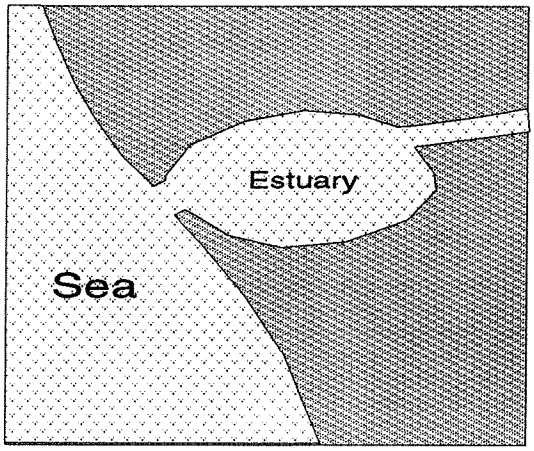


Figure 2.1: A simple tidal estuary

2.3 Closure Works

For various reasons a decision can be made to intervene in the process of inflowing and outflowing water, by closing the gap between sea and estuary: one could think of reasons such as the reclamation of land, the creation of a fresh water reservoir, the generation of tidal energy, and so on.

The closure of a tidal basin is technically difficult, and has numerous side effects which have to be kept in mind when deciding whether an estuary should be closed or not. Due to the multitude of aspects to be considered, the closure practice has had a long history full of struggle, through which experience has been gained by trial and error.

In reality, before a decision is made for tidal basin closure, a great number of considerations must be made with respect to the reason for closure, the location, and the impact which the closure will make on the environment and surrounding area. Various investigations and feasibility studies have to be made. As said, for the simulation program, these aspects have not been considered. The user of the program is not in any way confronted with these questions: his task can be described as purely constructional.

With respect to the structural part of the design, in practice a specific strategy is recognized. At first, in combination with the possible closure methods, the local hydraulic conditions are considered. The closure method that is finally chosen, also depends on the materials that will be used. The occurring velocities during construction lead to a force which these materials have to withstand.

2.4 Geometrical distinction of closure works

In the various methods for closing a tidal basin, two distinctions can be made:

* A distinction by *geometrical characteristics* of the closure;
* A distinction by *structural characteristics* of the closure.

In this and in the next chapter, both distinctions will be specified.

Based on *geometrical characteristics*, three types of closures can be considered:

* *Horizontal closures*, when the profile is restricted horizontally;
* *Vertical closures*, when the profile is restricted vertically;
* *Combined closures*, when a vertically restricted closure is finished horizontally.

2.4.1 Horizontal closures

When a gap is closed horizontally, the closure material is dumped from the channel banks. While this is done over the full height of the dam, the dam head moves progressively forward and is used as a base for further construction activities.

The dam head as well as the bed in front of the dam head, are seriously under threat from the currents. The construction should not be undermined; therefore precautions should be made. This can be done by constructing a bottom protection before the cross sectional area is decreased. Such a protection of the bed exists of a resistant layer, which can be spread out over several hundreds of meters of the initial channel profile.

The horizontal closure method can be applied quite easily from the banks of the gap. This clarifies why this method has been used so often. In figure *2.2*, a schematized view on the horizontal closure method is given:

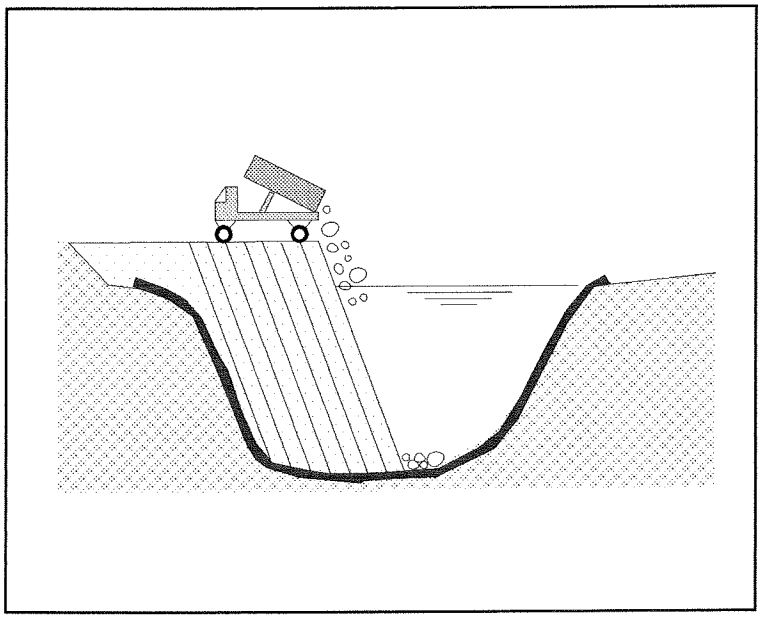


Figure 2.2: The horizontal closure method

2.4.2 Vertical closures

The *vertical closure method* distinguishes itself in dumping layers of material onto the bottom of the gap. Gradually, the gap area then decreases in vertical direction. As the construction advances, the water flow velocities increase above the dam crest, until the flow over the weir becomes critical; from then on, the flow velocities will only decrease.

In comparison with the horizontal closure method, the current attack on the construction and bottom protection is less severe; but the area undergoing current attack is much larger.

For the execution of vertical closures there are various possibilities. Temporal bridges or cable ways can be used, from which the material is dropped; dumping can also be done from dump vessels. These vessels though can only be applied as long as the water above the dam crest is deep enough.

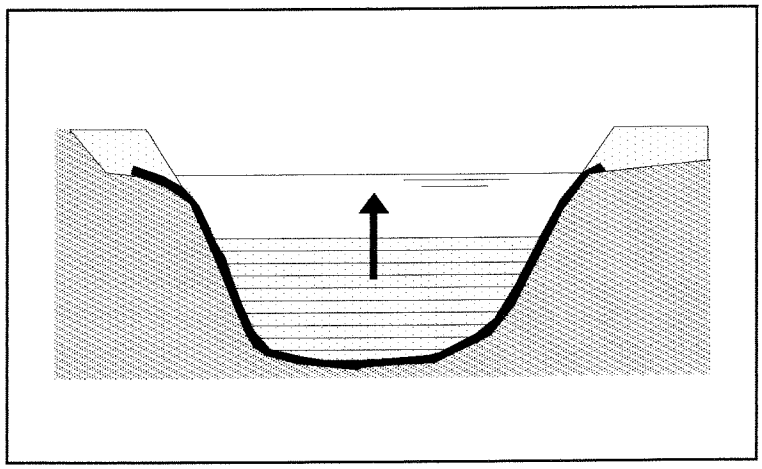


Figure 2.3: The vertical closure method

2.4.3 Combined closures

The third geometrical possibility exists in a *combined closure*. First a sill is realised over the whole width of the gap that is to be closed; thereafter, the construction is finished by dumping material from the banks of the gaps.

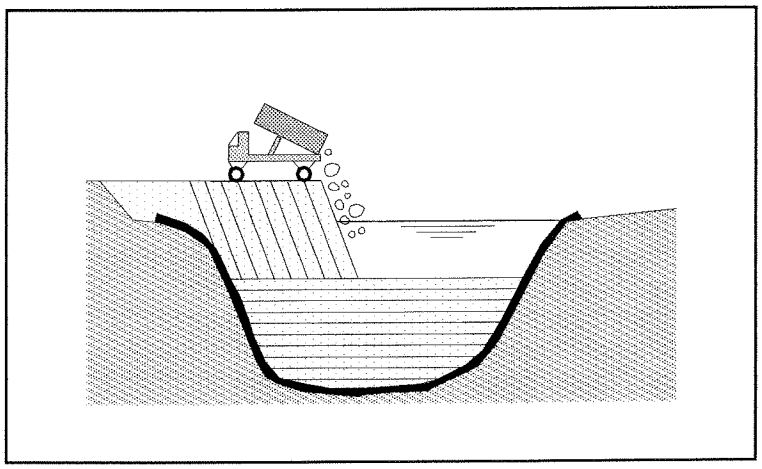


Figure 2.4: A combined closure

After a sill has been constructed, the gap could also be closed with large elements, such as caissons. In the actual version of the simulation program, this possibility has not been

considered as of yet. In figure 2.5 the caisson closure type has been illustrated:

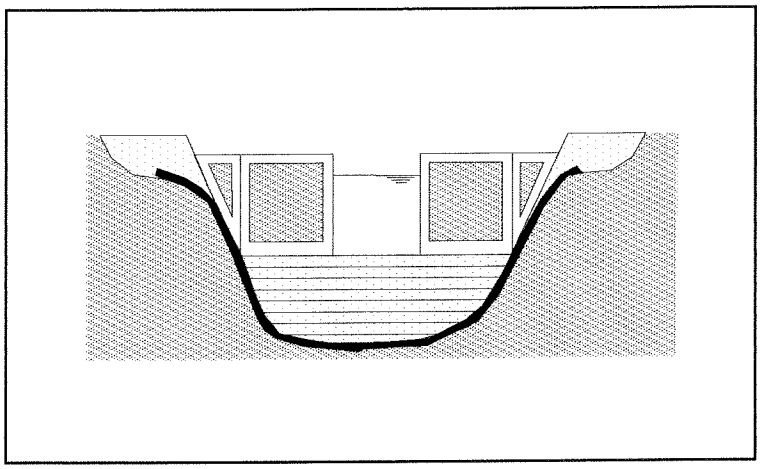


Figure 2.5: The caisson closure method

2.5 *Structural distinction of closure works*

Based on structural and material characteristics of the closing practice, three types of closures can be discerned:

* *A sand closure:* when this method is chosen, only sand is used to close the gap. It will be clear that a closure with sand only will not be possible in every circumstance. Both the hydraulic conditions (currents and waves) as well as the availability of sand are aspects to be taken into account. In the Dutch situation, especially the availability of sand is a positive aspect when compared to the use of quarry stone, which needs to be imported.

  The sand closure usually occurs horizontally, in which the sand is brought into suspension and discharged into the gap from the banks. In other occasions the sand is dumped vertically from hopper dredgers or split barges, after which the dam is completed horizontally.

* *Gradual closure with larger material:* in this case quarry stone, concrete blocks or gabions are used to close the gap. This construction method has been applied most frequently. This closure type can be applied either horizontally or vertically. Often, the larger material is not available, or transporting it to the closure site would cause this option to be too uneconomic. In this case, concrete blocks could be made.

* *Closure with caissons:* If a gap is closed applying caissons, first a sill is constructed vertically to provide a foundation for the closure elements. After that, the elements are brought into the gap and placed on the flattened bottom. Special elements can be used such as open caissons; when these are placed in the closure gap, they can be closed suddenly. Such a sudden closure could be interesting because that way gaps with high occurring velocities could be closed.

In the following chapter, the sand closure and the gradual closure with larger elements have been considered in detail.

2.5.1 Sand Closures

2.5.1.1 Method of Closure

A *sand closure* is executed by the application of great volumes of sand, which are supplied either by a hopper dredger or split barge from which the sand is dumped, or by a pipeline which takes a water-sediment suspension to the head of the dam.

A general characteristic of sand closures is the shifting and loss of the construction material. A sand closure is based on the requirement that the production of sand is larger than the loss. Also when the current velocities in the closure gap are not very high, sand losses occur. Losses can be distinguished in *net loss* and *gross loss* [Huis in 't Veld, 1987]. Gross loss is sand that is moved to another place than where it was dumped, but still fits within the future dam profile; net loss is material which is taken beyond the profile of the final dam. It can be seen, that after construction the latter material has been of no use, and that net loss should be avoided as much as possible. In figure *2.6* the different types of losses have been illustrated:

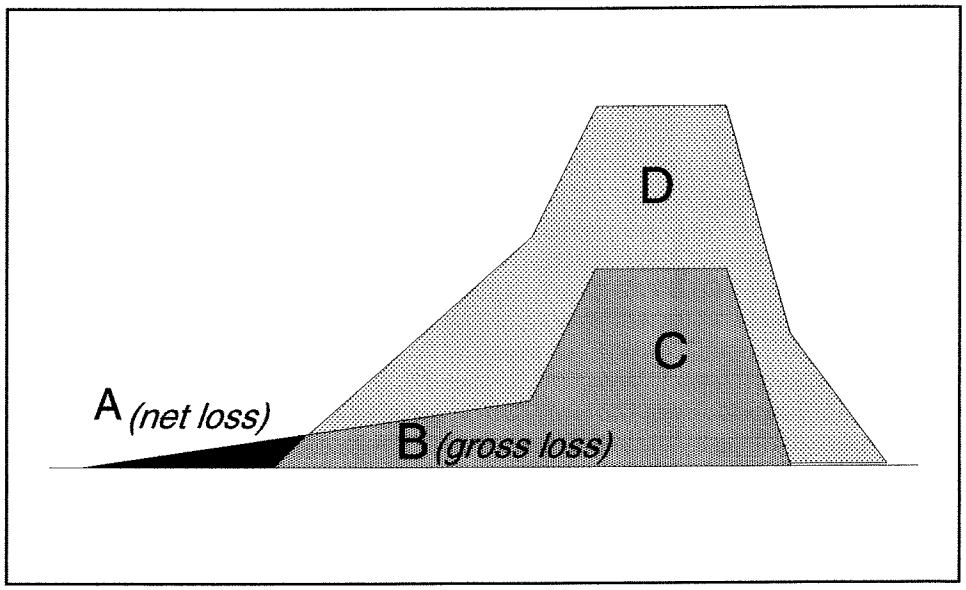


Figure 2.6: A sand dam cross section

In this figure, *D* indicates the dam volume that will finally be constructed. The smaller volume *C* indicates the stage, in which the gap has been closed already. Hence, *B* has been "lost" during the construction of *C*, but will be useful because it reduces the required volume for the final volume *D (gross loss)*; volume *A* though, has been lost during the construction of *C* and will not be useful *(net loss)*.

In the simulation program, both volumes *B* and *A* will be considered as loss. It is as if the task for the user of the program is to construct only *C*. The losses *B* are especially important because they reduce the sand production for the closure and hence the advancement of the closure.

In studying whether a sand closure will be possible, attention has to be paid to the maximum losses which will occur per tidal period. When the greatest possible production (also per tidal period) is higher than this maximum loss, the dam closure by sand will be possible. The production always needs to be higher, because of the insecurities in the values of both calculated losses and predicted productions.

As shown in figure *2.7*, the losses as a function of the gap area can be described by a curve with one extreme [Konter et al, 1992]. In most cases, these extreme losses will take place when the gap area is about 0 to 30 % of the initial gap area.

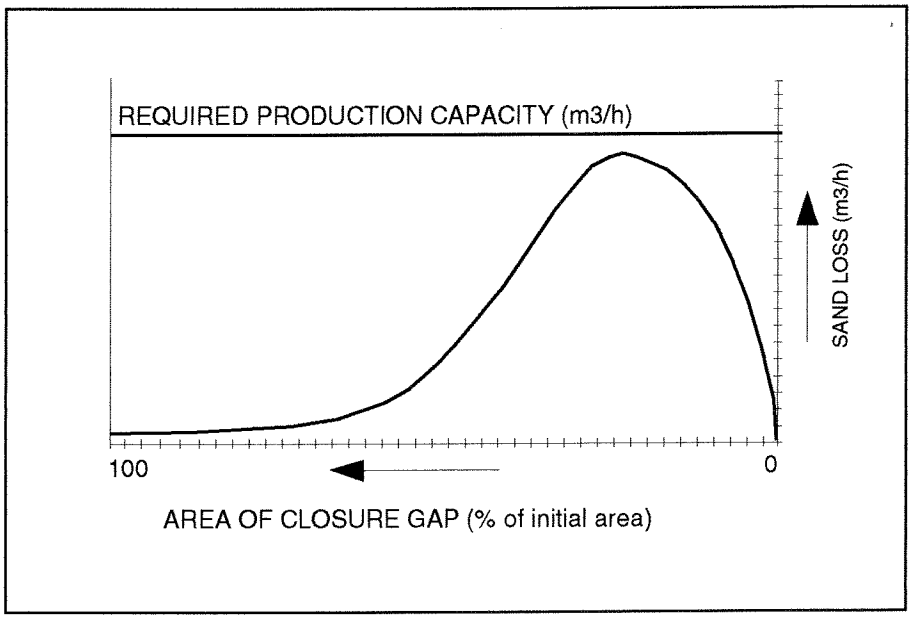


Figure 2.7: Sand losses during closure

It is interesting to see, that the maximum sand loss does not occur when the gap is nearly closed. Then, the current velocities are high, but the width at which erosion takes place is so small that the total sand loss is small too.

In general it can be concluded from experience, that sand closures are possible when the maximum current velocity does not exceed 2.0 to 2.5 m/s [Konter et al, 1992].

2.5.1.2 Cross sectional Design of a Sand Dam

To choose the right design and construction method of a sand dam can be very important for the dam cross section and the losses that will occur. The most chosen construction method for sand dams is the horizontal one in which pipelines discharge sand in suspension from the banks into the gap.

Crest width

In case the dam construction is horizontal, the crest width depends on the amount of pipelines that will provide the filling material. In practice, it is favourable to restrict the crest width, because it reduces the dam volume and both gross and net loss. A wider crest has the advantage that it leaves more available working space.

Equipment

In the actual version of the program no attention has been paid to equipment - pumps, pipe lines which have to be lengthened, etcetera -, but closing activities are simply entered as a production (in *volume per time*). Future versions of the program could be extended with respect to this, and thus make the program to be more realistic.

Particle sizes

Especially in the case of a sand dam, the dam slopes depend a lot on the size of the particles. Hence, when smaller material sizes are used, there will be more losses and at the same time the total dam volume (and thus the required amount of material for the closure) will be larger as a result of the gentle slopes.

2.5.2 Gradual closures with larger elements

2.5.2.1 Method of Closure

When a gradual closure is made with quarry stones or concrete elements, the construction can be either vertical or horizontal. In case of a horizontal construction the dam heads are exposed to the strongest current attack. The upstream side of the dam head has to withstand the majority of the load. If the closure is vertical, the hydraulic load is less severe, but occurs over the whole width of the gap.

In both construction methods, the velocities that will occur during closure are different. As has been said before, when the closure is vertical, at a certain moment the velocity will not increase anymore because the flow on the weir has become critical. At that time, the velocities do not depend on the level difference, but only on the energy height of the upstream water with respect to the sill. Hence, when the closure is continued, velocities will be lower. For horizontal closure, the velocities increase until the whole gap is closed. The velocities for horizontal and vertical closure have been shown in figure *2.8* on the next page. Also the possibility for a sudden closure as mentioned in section *2.5* is indicated.

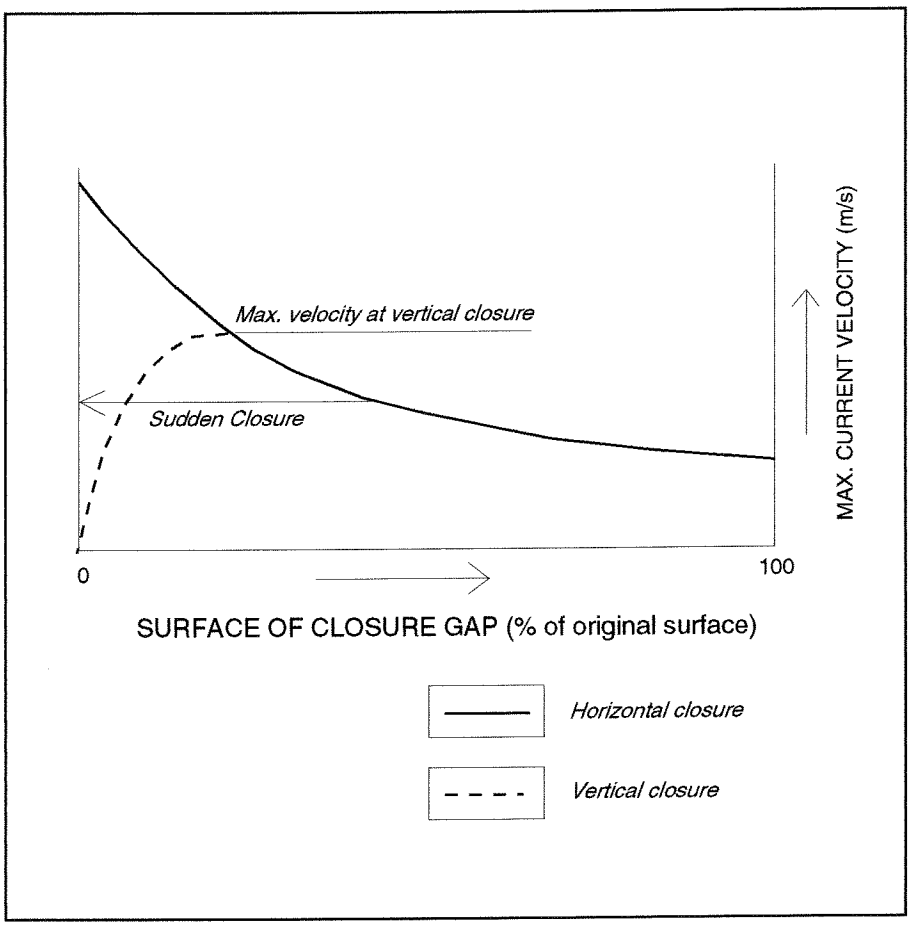


Figure 2.8: Current velocities during closure

As a result of this difference, a vertical closure can be executed with smaller stone sizes than a horizontal closure.

2.5.2.2 The gradual horizontal closure

If a closure gap is constricted horizontally, the current velocities increase in proportion to the gradual decrease of the cross section area. Therefore, for the last gap to be closed, heavy material is needed since the current attacks are severe then. In areas with a large tidal range, the horizontal closure is not favourable because of these high velocities. If it is still decided to close horizontally, the last gap could be closed by a larger element (*sudden closure*).

Another solution would be, to build a sill first, maybe even so high that critical flow is achieved, and then continue the closure horizontally. This is called a *combined closure.*

2.5.2.3 The gradual vertical closure

Characteristic of a gradual vertical closure is the raising of the sill over the whole width of the gap. It is possible to construct in horizontal layers, but also to follow the shape of the cross section of the channel to be closed. Quarry stone, clay- or sandbags are used for a vertical closure. The dumping of these materials can be done in various manners, as:

* By *manual labour* or by *trucks* over a (temporal) bridge;
* By *floating equipment,* such as stone dump vessels or floating cranes;
* By *cableway;*
* By *helicopters.*

As has been said, in the simulation program no attention has been paid to the various equipments that can be used. Before a vertical closure is initiated, the user has to decide whether the closure will be done from *above* the water line (e.g. by cable way or helicopters) or from *below* the water line (e.g. by dump barges). This is, as will be explained later, important for the calculation of the dam advancement.

2.5.2.4 Bottom Protection

When a dam is built to close a tidal basin, the inflow and outflow through the narrowed gap will be reduced, causing a decrease in the tidal range in the basin. But at the same time, the current velocities through the remaining opening will be higher. As a consequence, the scouring effect on the bottom near the dam will be increased, which endangers the stability of the river bottom and thus the foundation of the structure to be built on this bottom. This implies that the bottom, when consisting of easily erodible sand, must be protected by current-resistant material. Naturally, these problems will not be encountered at places where the bottom material consists of rock.

When a bottom protection is laid, scour holes will develop at the end of this protection. These scour holes, which depend on the magnitudes of turbulence and velocity and the

period of exposure, could threaten the construction for stability reasons. The optimum length of such a protection should be considered. In the picture, an example of closing structure, bottom protection and scour hole has been given.

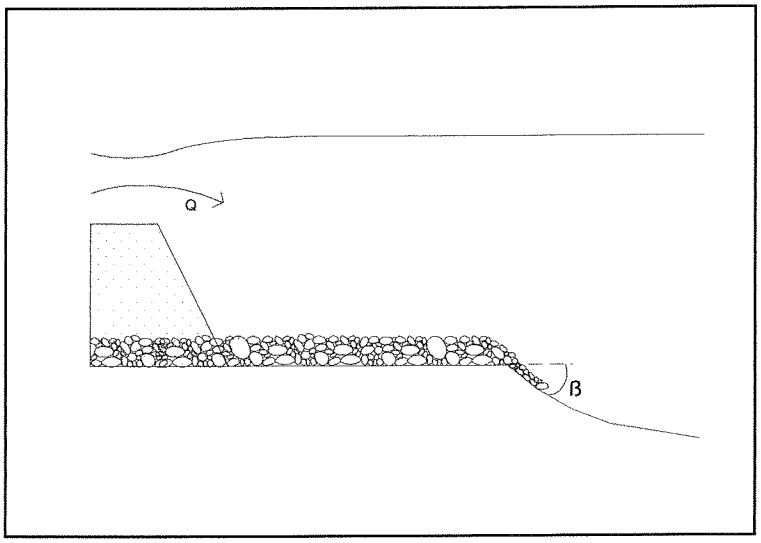


Figure 2.9: The bottom protection

The longer the protection, the more expensive; but a longer protection is effective in two ways: the scour holes that develop will be smaller, because the flows will be less turbulent at the end of a bottom protection, and at the same time the threat of the scour will be less, because it will occur further away from the structure.

For the protection of the bed, there are various options available to the designer. The protection can be prefabricated, or constructed in situ; it can be permeable or impermeable; many materials can be used. Each of these types of bottom protection will respond differently to the loads that will be present.

After a scour hole has been formed behind the protection, this hole could collapse when a certain depth or slope is reached. The behaviour of the bottom protection in case of such a collapse is an important point of attention. If the protection settles, but its function is not destroyed by this settling, the lowered bottom is still protected and the maximum scour still occurs at the same location as before the settlement; an example of such a bottom protection could be achieved when geotextiles are used. If settlement does damage the protection, the situation changes. Collapse of the scour holes will then lead to a shorter bottom protection: the location of the maximum scour approaches the dam construction and the total scour will increase, because closer to the dam the vortices are

more intense. In this case, collapsing scour holes behind the bottom protection will soon threaten the construction itself.

In the actual version of the simulation program, no attention has been paid to the materials that can be used to protect the bottom. The quality aspect though, has been considered: the user of the program can decide how a bottom protection will respond to a collapse of the scour holes behind it, by marking the quality of the bottom protection to be made as being good or bad. A *good* bottom protection will keep its function when it has settled, while a *bad* protection, once distorted, will not protect anymore.

The quality of the bottom protection is important with respect to the scour depths (which depend on the length of the bottom protection), and to the place where this scour occurs (at the end of the protection). In figure *2.10*, this has been explained:

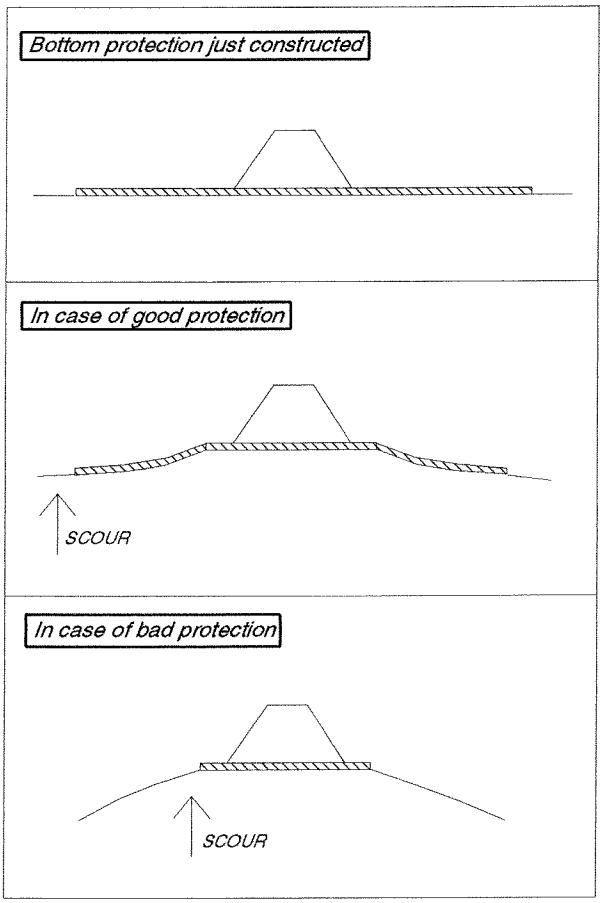


Figure 1.10: Protection and settlements

In case of a good protection, the scour will go on at the same location as before the collapse. In case of bad quality though, for this section the bottom protection length will be decreased. Scour will then occur right behind the new end of the bottom protection.

2.5.3 Sudden Closure

The sudden closure such as with caissons has not been considered in this version of the program. When the program is extended to cover this closure type, a series of extra calculations have to be included as well, because the design of a sudden closure has several other implications than a gradual closure.

3. Simulation of Closure Works

3.1 The Simulation Program

The program *CLOSIM v 1.0* has been written to simulate closure works in tidal inlets. Aim of the program has been, to give an impression of the phenomena that might occur in the closure practice. The program would help, to bridge the gap that exists between theory and practice, by enabling the user to close a fictive basin and by calculating the situation at the closure site.

In the previous chapter, an introduction in the closure practice has been given. In order to be able to get acquainted with the main phenomena that occur during closure, the simulation program *CLOSIM v 1.0* was written. The actual version of the program contains a lot of assumptions and restrictions that should be worked out in future versions.

In the following chapters, the mean features of the program are explained.

3.2 Scope of the Simulation Program

As said before, the phenomena that occur during closure are complex and therefore the scope of the simulations in *CLOSIM v 1.0* has been restricted. The program only deals with gradual closures; sudden closures with caissons could be added in future versions.

With respect to the *input* of the program, there are various limitations. The dimensions of basin, gap and structure are roughly schematised; the maximum number of closure

gaps is three. The bottom material is assumed to be granular and non-cohesive; further details about the soil conditions are not included. Of the weather conditions, only storm is regarded; the direct impact of waves on the structure has not been taken into account.

The *calculations* apply existing designing relationships, which are extended where needed to be suitable for the simulation program. As has been explained in section 3.2.2 of volume *I*, the flow calculations are executed with the *storage model*, which has several implications for the cases that can be considered. For the slopes that will occur when the material is dumped, rough estimations had to be done, because no better information was available. Also on the quantitative erosion of coarse dam material, no investigations have been done yet. In the program, existing relationships have been extrapolated, as has been explained in chapter *5* of volume *I*. For these erosion calculations, various flow regimes should be counted for. An important parameter for these calculations is the *porosity* of the structure, which has not been dealt with, but which should be included in future versions. Contraction effects of the currents that pass the dam head, have been ignored up till now.

Important is also the limited number of failure mechanisms that has been included. The most important of these mechanisms have been illustrated in figure *3.1*:

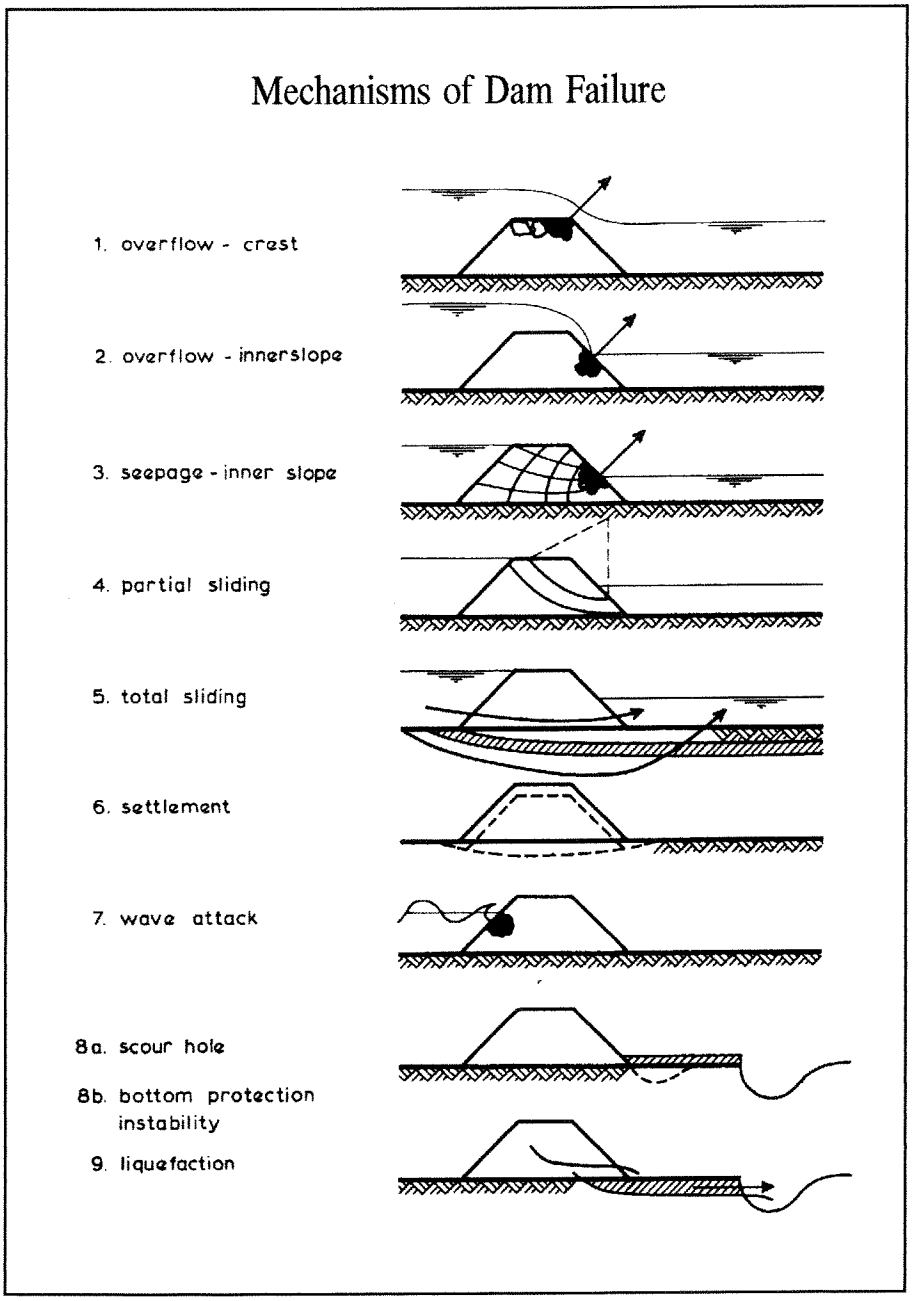


Figure 3.1: Failure mechanisms for closure dams

When these mechanisms are observed, it is clear that a lot of main failure mechanisms should be included in future versions of the program. In *CLOSIM v 1.0*, most of the attention has been paid to the mechanisms *1* and *8*.

4. Getting Started

4.1 How to Install the Program from the User's Disk

The simulation program *CLOSIM v 1.0* exists in two versions, a *programmer's version* and a *user's version*. The programmer's version has been described extensively in volume II of the description paper.

To install the user's version of the program, simply enter the user's disk in drive A and type *A:\INSTALL*. The installation program will write the simulation program on your hard disk in an own subdirectory *CLOSIM*. After installation, this directory should at least contain the following files:

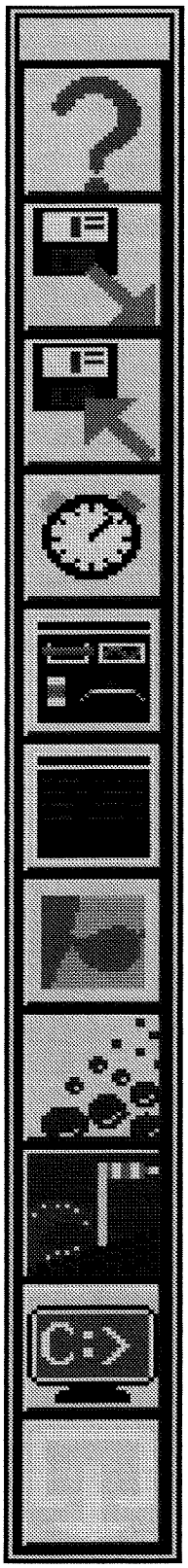
* *CLOSIM.EXE*	The executable program;
* *CLOSIM.BMP*	The opening screen bitmap file;
* *DEFAULT.CTB*	A default closure situation;
* *SIMFREE.DSK*	An empty desktop file.
SAND1.CTB	An example for sand closure
CABLE1.CTB	An example for cable way closure
STONE1.CTB	An example for a rock fill closure

The files which are indicated with an asterisk, are indispensable for running the program.

4.2 Running the program

When the program is installed on drive C, it can be started by moving to the directory *C:\CLOSIM* and typing *CLOSIM <ENTER>*. The opening screen will appear, and the default values will be loaded.

You will now see an empty desktop on the screen. At the top, the *drop down menu* can be selected with the mouse; at the bottom, a *status line* is shown. With the button bar at the left side of the screen some of the options are available directly. Which, has been shown in the figure on the next page.



PROGRAM INFORMATION

LOAD CLOSURE SITUATION

SAVE CLOSURE SITUATION

START / STOP CALCULATIONS

OPEN GRAPHICS WINDOW

OPEN NUMERIC WINDOW

ENTER NEW CLOSURE

GRADUAL CLOSURE

START A STORM

TEMPORALLY TO DOS

QUIT PROGRAM

In chapter 5 all options of the drop down menu have been discussed separately.

To execute the simulation of the closure of the default simulation, you might click on the button with the stopwatch. When you have done this, the question appears whether you want to change data on the closure situation. When you select *Adapt*, a series of dialog boxes will pop up in which you can define the closure situation (see chapter 5).

When you do not want to adapt data, two windows will appear on the desktop and the simulation process will start. The screen will look like this:

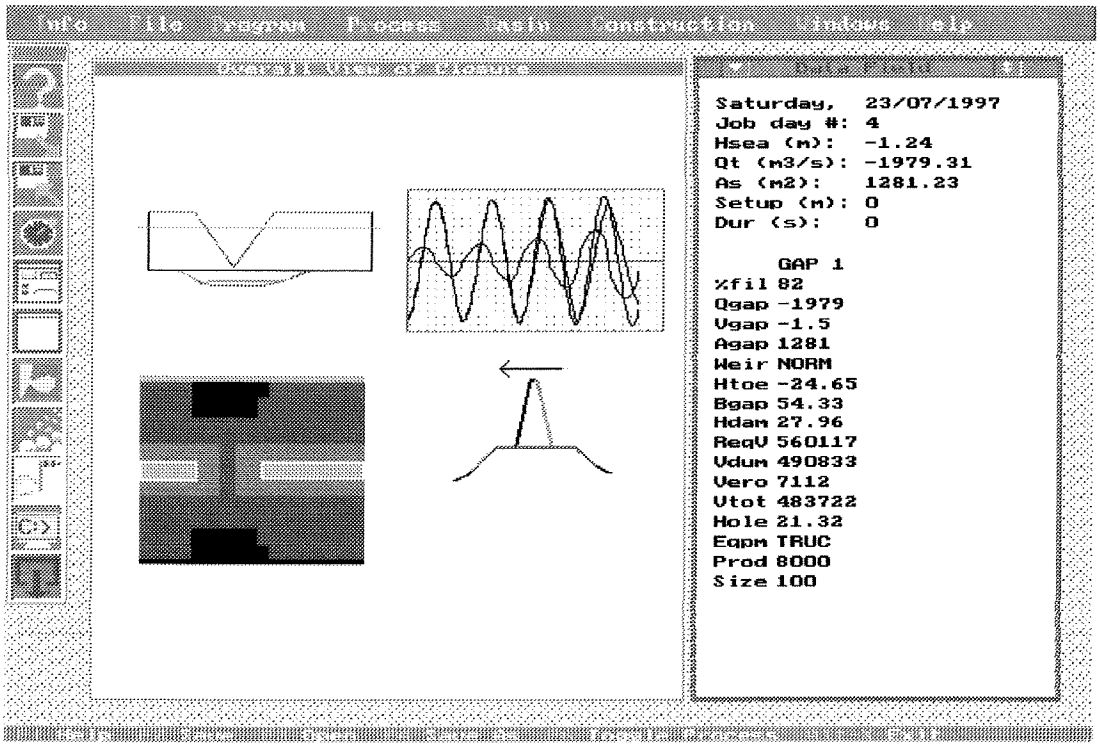


Figure 4.2: A hardcopy of the program screen

In the left window (the *Overall Window*), the results of the calculations will be presented graphically; in the right window (the *DataBox Window)*, numerically. In the next paragraphs of this chapter the contents of both windows will be discussed.

4.3 The Overall Window

As said before, the *Overall Window* displays the results of the calculations graphically. In this window, we can distinguish four items. The four items of this window have been explained below:

1. The Length Section of the Dam

In the upper left corner of the screen, the length section of the dam is shown. In this section, the basin level and sea level are drawn, the original gap profile, and the actual gap profile. In the figure, the scour hole contour behind the bottom protection is drawn:

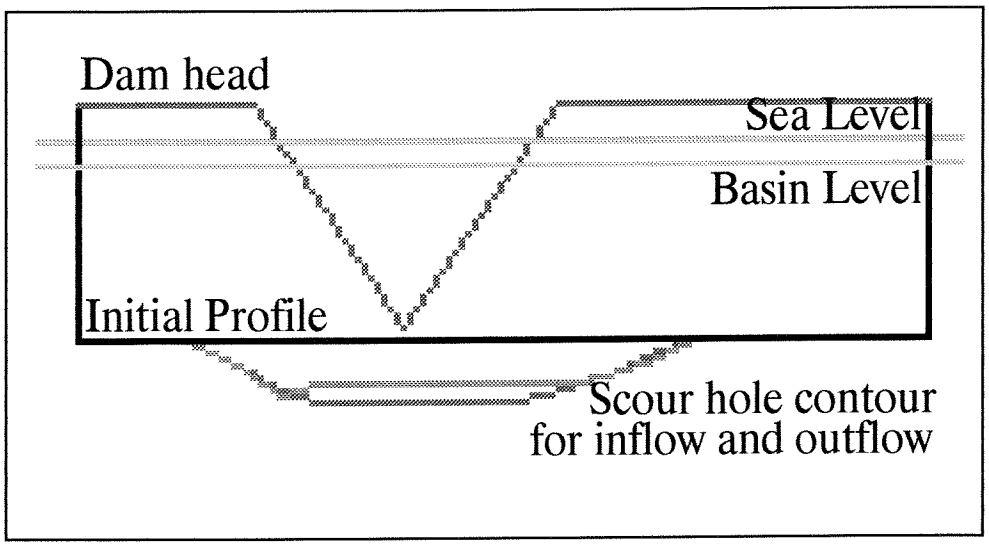


Figure 4.3: The Cross Section in the Program

2. The Level and Velocity Graphs

In the upper right corner of the window, the levels and velocities are shown. The basin level is drawn red, the sea level blue. The velocities are indicated in light green:

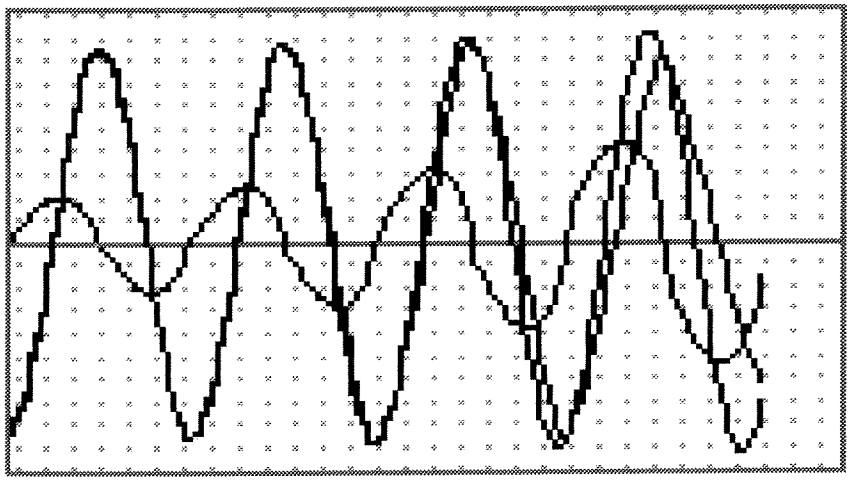


Figure 4.4: The Level Graph in the Program

3. The Top View of the Dam and Bottom Protection

Down below, at the left side of the window, a top view of the closure is given. In this top view the advancement of the dam is shown. The indicated areas at both ends of the bottom protection, are sections in which settlement has taken place:

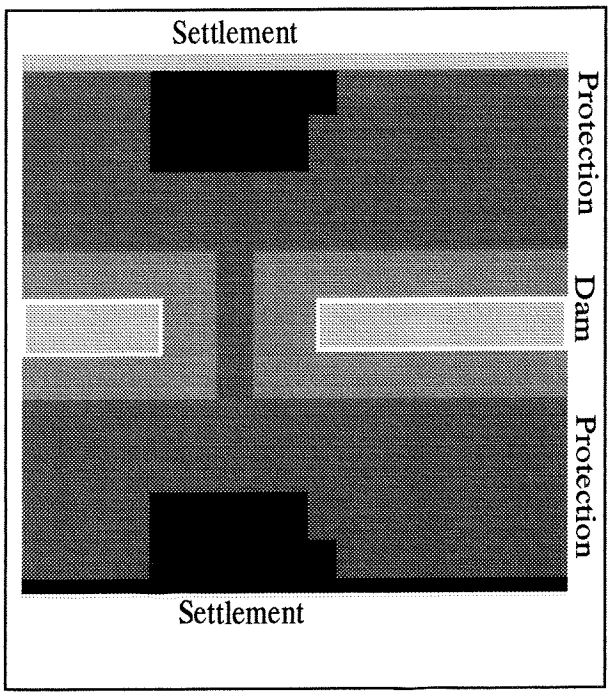


Figure 4.5: Top view of the closure

4. The Cross Section over the Dam and Bottom Protection

The item right below in the screen is a cross section over the dam. The dam is drawn, the bottom protection, as well as the deepest scour hole at the end of the protection:

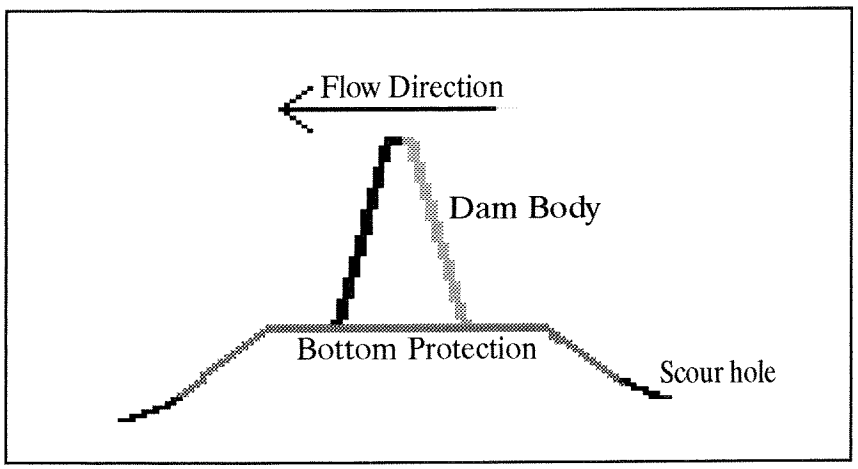


Figure 4.6: Cross section of Dam

4.4 The DataBox Window

The *DataBox window* presents the results of the calculations in numerical form. In *Appendix C*, a quick reference of all parameters in the *DataBox* window has been given. It should be mentioned, that when this window is turned off, the process calculations are done much faster.

5. The Program Menu Options

5.1 The Information Menu

5.1.1 About the Program

The about box from the information menu contains general information about the program.

5.1.2 Introducing the User

With this option, or with the *question mark button* at the left side of the screen, the user obtains a general introduction to the program in which he is told what to do.

5.2 The File Menu

5.2.1 Saving a Closure Situation

At any time, a closure situation can be saved. It does not matter whether the closure has been started already or not. The name for the closure is the last name with which the closure has been saved or loaded. If you want to give a new name to the closure, then use the option Save As, which will provide a file selector box in which you can enter the new name. The file name should have the extension *CTB*; if no extension is entered, *CTB* is added automatically.

In *Appendix A*, the structure of a *CTB* file has been shown. The *CTB* data files have been commented, and can be edited with any ASCII editor such as the DOS program *EDIT*.

The Save option can be chosen using the drop down menu, the Save button or the Status Line on the bottom of the screen. The *F2* key activates the Save option, the *F4* key the Save As option.

5.2.2 Loading a Closure Situation

The closure situations on disk can be loaded into memory, and the calculations can be continued. Choose the file to load with the file selector. The closure files of *version 1.0* have the extention *CTB*. If the chosen file doesn't have the CLOSIM header or if the disk station is not ready, a warning appears.

The load option can be chosen from the drop down menu, with the *F2* key, clicking the status line at the bottom of the screen or with the load button at the left side of the screen. If the default situation must be loaded, press *F9* or choose the file *DEFAULT.CTB* to be loaded.

5.2.3 Changing the Main Directory

The main directory of the closure program can be changed, if closure situations are written in other subdirectories or if, for example, the data files should be written on floppy disk. The default directory is the directory from which the program was started.

5.2.4 Go to DOS shell

At any time the user can temporally reach the DOS shell from the simulation program. This can be useful to do some disk operation, or to peek into another program and then come back into the actual closure situation.

When the option is chosen using the drop down menu or the shell button at the left side of the screen, the dos prompt will appear. To come back into the program, type *EXIT*.

5.2.5 Quitting the simulation program

With the *QUIT button*, the *drop down menu*, the *status line* or the *ALT-X* key, the user can leave the program and go to the DOS shell. The actual closure will be abandoned; the user is not requested whether the closure situation should be saved.

5.3 The Program-related Options

5.3.1 Toggle Sound

With this option, the sound options of the simulation program can be toggled on or off. It should be mentioned, that version 1.0 of the program not much attention has been paid to sound features; the only tunes the program produced are the buzzers when a closure is finished or collapsed. It can be useful though to disable or enable these buzzers for demonstration purposes, especially when more features are added in future versions of the program.

5.3.2 Select Color Palette

With the Select Colors option, the screen colors can be defined. Especially when the program is used for demonstration or when a monochrome monitor is used, the colors should be adjusted.

When the desktop is saved, the colors are also written.

5.3.3 Log Files

When the user of the program wants the results of the calculations to be written to disk, this option offers the possibility to elect which results are written. When the option is selected, a dialog box appears on the screen. The data that can be written on file is:

The production and erosion in the gaps

The flow velocity (time or gap area dependent)

The gap dimensions (width and depth)

The levels (basin level and sea level)

One or more of these log files can be selected, before or during the calculation processes. When the calculations go on, the required results are written in the specific log files, which each have the extension *LOG*. The production and erosion data are stored in the file *EROSPROD.LOG*; the velocity in *VELOCITY.LOG*. The gap dimensions are written in *GAPDIMEN.LOG*, and the levels in *LEVELS.LOG*.

After the user leaves the program or whenever the log file option is chosen from the main menu, the log files are closed. They are now ready to be imported in any spreadsheet. From a spreadsheet, for example *LOTUS 123* or *QUATTRO PRO*, the data can be imported (using the commands *slash / File / Import*). After, a graph can be made selecting the *A* column as *X-series* and the *B, C* and *D* column as series *A, B* and *C*. In the graphs the whole behaviour of the saved parameter will be shown.

5.3.4 Load and Save Desktop

The *desktop* of the program (*menus, colors* and *window locations*/sizes) can be written to disk at any time. Especially in future versions of the program this can be useful, if more windows are used, for example to focus on the several characteristic details of the closure. In this stage, the desktop will allways be saved as *SIM.DSK*.

5.4 The Process-related Options

5.4.1 Load a Default Situation

When the program is started, the file *DEFAULT.CTB* is loaded, providing all default values for the closure situation. Whenever this default data file is changed, the program will start with other default values.

At the time the calculation processes are busy, it can be useful to return to the startup values. The Default option can be chosen from the drop down menu, with the status line or with the *F9* key. When this is done, the actual closure is shut down and all windows are closed.

When the default closure has only one gap, the other gaps do not exist but allways have program-determined default values as well. These values will show up if the number of gaps is increased.

5.4.2 Entering Data for a New Closure

When a entirely new closure situation has to be entered, this option can be chosen from the *drop down menu* or with the *Closure* button. All dialog boxes for the input of data concerning the closure will pop up, and the user will be able to change all information.

When the number of gaps is increased, a new gap will be added with the default characteristics. With the dialog boxes, these data can be changed. At any time the series of boxes can be interrupted by canceling a dialog box or pressing the escape key.

When a closure process is started, the user is asked whether to adapt the data, or to start with the actual information. If adaptations are required, the *New Closure* procedure is called automatically.

5.4.3 Initiate Process

At any time, the closure can be initated. The result of this procedure is that the closure will be put to the situation as if there never has been a dam construction. The flow profile is reset to an equilibrium, where erosion and sedimentation cause a stable situation; all dumped dam material is removed; the bottom protection is restored to its original unsettled state, and is assumed to be constructed recently - which results in no scour holes behind the construction.

When data about the conditions of the closure situation is changed (such as information

about the tidal components or the area of the tidal basin), the process is initiated automatically. In other cases the option can be useful when a closure is begun (or finished) and has to be restarted. The closure can be initiated from the drop down menu.

5.4.4 Starting the Calculation Process

When the data for the closure has been entered, the calculations can be started. Before starting, the user is asked whether he wants to change the actual information on the closure. After that, the information is verified. If the values are incorrect, a warning appears and the user is enabled to change the data.

When all data is correct, the process starts and two windows are opened. The left window on the screen contains a graphical presentation of the results of the calculations. The right window gives numerical informatio on the closure situation. In *chapter ###* the contents of both windows has been described.

In several cases the program interrupts the process calculations. Every time a user dialog takes place, the calculations stop temporally. When a main situation change occurs, the process stops so the user can focus on the actual situation. In these cases, with the *Start* option the calculations are continued.

The *Start* option can be selected from the *drop down menu*, with the *F5* key, with the *status line* or using the *Stopwatch Toggle button*. With this *Toggle button* the process is either started or interrupted.

5.4.5 Interrupting the calculations

The process can be interrupted in three ways. As said, the program can stop the calculations while a user dialog takes place or a major change occurs. The user can want to interrupt the screen temporally - for example when an explanation is given and the actual screen has to be considered. He can do this simply by pressing the DOS pause key.

In other cases, it can be useful to interrupt the process while the main menu is still available. For this, the interrupt option can be selected (from the *drop down menu*, the *status line*, the *F6* key or with the *Stopwatch Toggle Button*). While the process is interrupted, decisions can be made, the process can be saved or a hardcopy of the screen can be made. With the *Start* option (*F6* key or the *Toggle button*) the process can be continued.

5.5 Input of the Closure Situation

5.5.1 General information on the Closure

When the option is chosen, a dialog box appears in which the number of gaps, the Chezy value and the grain size of the bottom material at the closure location can be entered.

As mentioned before, when the number of gaps is increased, for the new gap the default values will be valid. In that case the user is asked to adapt all these values for the new gap.

In version 1.0, of the bottom material only the grain size (in mm) and the bottom roughness (in √m/s) are entered. In future versions this could be quite extended with information about the subsoils and parameters for non-granular material.

5.5.2 The Date and Time at the closure site

Although the date and time at the closure site are not important for the calculations in the simulation program, they can serve to give the user an impression of the time range in which the closures take place. During the closure, the job date and time are exposed on the screen as well as a job day counter.

5.5.3 Information on the Basin Area

As explained in the first volume of this paper, the basin storage area is not a constant, but will vary with the level in the basin:

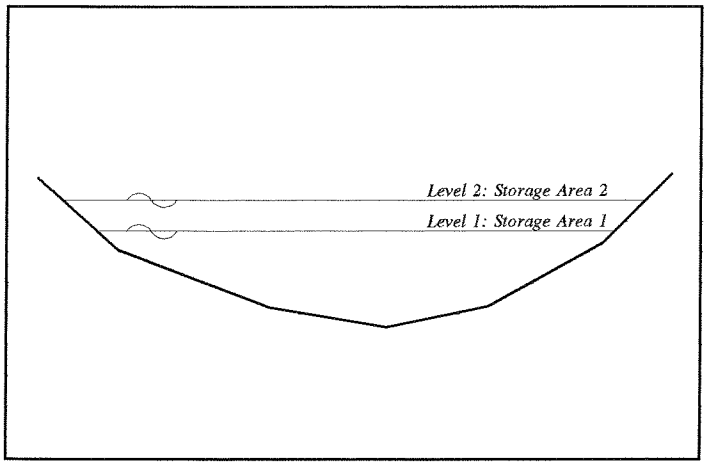


Figure 5.1: Storage Areas

Each basin has its specific storage function, which gives the relation between level and storage area. The storage function is very important for the flow calculations.

In the simulation program, this relation is schematized as an interpolation between five known combinations of level and area:

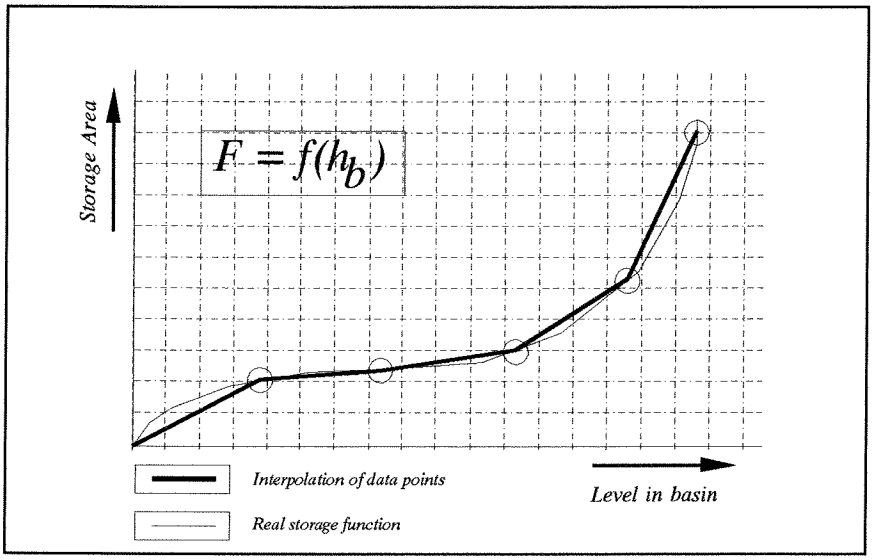


Figure 5.2: The Storage Function

When the Basin Area option is chosen, a dialog box appears on the screen in which

these five combinations can be entered. The levels are expressed in m, the areas in m2. The data should be entered in increasing order of levels and areas. The smallest and biggest values for the levels should be such, that the whole possible level range that can occur during the closure is covered. If this is not the case, a warning will appear.

5.5.4 The Depths and Widths of the gaps

When the depths and widths of the gaps are entered, the gap banks are assumed to be vertical and the bottom horizontal. During closure, the dam heads may have slopes.

The gap width (in m) is restricted to a minimum value of 15 meters. It should be mentioned, that the input of the gap depth (in m below MSL) has only a relative function: when the gap dimensions have been altered, the process is set to a stable situation, in which the bottom is ajusted so that the flow gap area will be in equilibrium. In case there are more closure gaps now, the depths of the gaps will be ajusted relative to the depth entered by the user. This adaptation of the gap depth should be kept in mind, especially when a gap dimension is entered and the tidal conditions or the basin area change, the gap depth will also change.

5.5.5 Information on the Tidal Conditions

The tidal variations of the sea level can be described by a fourier series of sine functions. The sum of the contributions of each of the components, results in the tidal characteristics as are observed in reality.

As explained in volume I of this paper, in the simulation program the tidal conditions have been decided to be described sufficiently accurate by the use of four of the main components. The contributions of the M2, S2, K1 and O1 can be entered, with which most typical tidal situations can be schematized

The periods of each of these components are entered as constants in the program (in s), and can not be changed by the user. In the dialog box that will be on the screen when the option Tidal Conditions is chosen, the amplitudes (in m) and the tidal phase at the start of the calculations (in s) can be entered. The sum of the amplitudes indicate the maximum tidal amplitude that can occur. When combinations of several tidal components are made, neap tide and spring tide can be simulated.

The tidal conditions can not be changed when the calculations have been started - when the Tidal Conditions option is chosen, the process will be initiated.

5.5.6 Information on a River Flow

In version 1.0, a river connected to the basin is simply schematized as a constant flow. During the construction, this river flow might be changed by the user.

When a river flows into the basin, the construction of a spillway should not be forgotten (chapter 5.6.7)

5.5.7 The Weather Conditions: Storm

In the actual version of the program, the only weather condition that can effect the closure is a storm. As explained in volume I, a storm is characterized by its setup, its duration and its probability of occurrance.

When the option Define Storm is chosen, these three figures can be adapted. The storm surge is restricted to a maximum value of 10 meters; the duration is expressed in hours and the probability should have a value between 0 and 1.

The option Start Storm temporally sets the storm probability to 1, causing a storm to occur. This option can also be invoked by pressing the storm button. If a storm is started when the storm data has not been defined, the dialog box will appear.

5.6 The Construction Activities Menu

5.6.1 Choosing the Construction Method

With version 1.0 of CLOSIM, two horizontal construction methods and two vertical methods can be chosen. If needed, two methods can be combined - for example when a gap is closed horizontally after a sill has been constructed. The four methods that can be chosen are:

* Construction from the dam banks, using dump trucks;
* Construction with dumping vessels (vertical);
* Construction with a cable way (vertical line dumping)
* Construction by dumping sand with pipe lines.

The two vertical methods are different, because the structure will advance in another way. When *line dumping* is applied (using a cable way), the added volume will have a *triangular* form, while the dumping from vessels will be in slices.

The construction method determines, with the diameter of the used material, the angle of the slopes of the structure. Each method has its own restriction for the grain sizes that can be used. If the chosen combiniation of material and method is not correct, the user is requested to adapt the values.

As mentioned, during construction the building method might be changed; this will not lead to resetting of the closure situation.

5.6.2 Construction of a bottom protection

At any time, a bottom protection can be constructed. If the option is chosen during closure, this means that the existing protection is repaired.

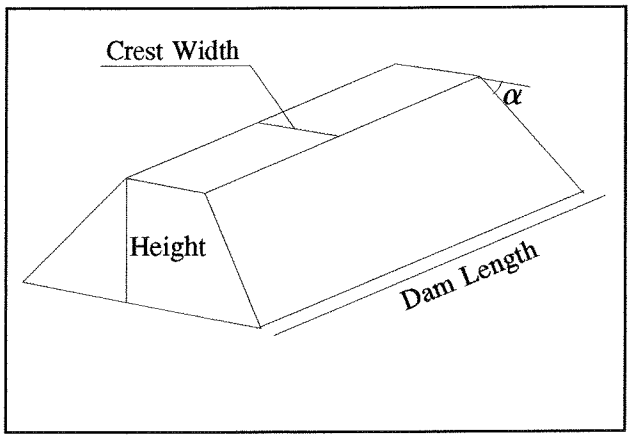
In reality, the design for the bottom protection has to be verified for various mechanisms of collapsing. In this version of the program though, the bottom protection is simply defined by its length (the structure will be built in the middle of the protection) and its quality (good or bad).

A good protection is defined as a protection, that keeps its function after being settled.

A protection can settle when the scour holes behind it collapse. A bad protection will only protect if it did not settle; hence, when a scour hole collapses, the length of the bottom protection diminuishes.

5.6.3 Defining the Dam Dimensions

Because the dam is defined as a trapezoid, four parameters are enough to schematize it: the height, the crest width, the angle of the slopes and the length of the dam:



With this option, the dam height above MSL and the crest width are chosen. The total dam height will be the sum of this height, and the depth of the original gap. The angle of the slopes will, as noted before, depend on the size of the material and the construction method. The total length of the structure will be the original width of the closure gap.

In case the dam is constructed by a cable way, the top width is restricted to a minimum of 10 meters, because the material will be dumped from one or two lines.

5.6.4 Defining the Dam Angles

The dam angles determine the volume of the total construction. When a dam is constructed with trucks, a cable way or pipe lines, the user can not influence the angle of the dam slopes; the material just falls down and settles at an angle which will depend on the material size and the construction method.

Also when vessels are used to construct the dam, the angle of dumping will be restricted by this natural angle. In that case though, the layers can be dumped precisely and a flatter angle can be chosen.

5.6.5 Material and Production

When the *gradual closure* option is chosen, the user can enter several values:

* In case of horizontal closure, the *productions* from the left side and from the right side can be chosen, in m3/h;

* In case of vertical closure, the vertical production can be chosen, in m3/h. Also the *interrupt level* can be set. When this level is reached, the closure activities stop. This can be useful when a sill is constructed up til a certain level. After the activities have been interrupted, the closure can be finished by vertical construction.

* The *material size* can be chosen. The size of the particles is restricted to minimum and maximum values, according to the chosen building method.

The *gradual closure* option can be selected from the main drop down menu, with the *F8* key or with the *gradual closure button*.

5.6.6 Constructing a Spillway

When a river flows into the basin, a *spillway* should be constructed to spill the accumulating water in the basin. In this version of the simulation program, a spillway is simply defined as a *culvert* with a certain flow area. When a culvert is made, the gap flow area will have a minimum value.

Also without a river flow, the construction of a culvert can be useful, because it will decrease the velocities in the closure gap. Because of a certain combination of tidal conditions and basin dimensions, it could be impossible to close with sand only. After constructing a culvert, a closure with sand only might succeed.

5.6.7 Entering the Draft of the Vessels

When a closure is made with vessels, the vessels dump their material floating above the structure. During construction, the depth varies, because of the raising sill level and the variations in the water levels. It is obvious, that at a certain depth the draft of the vessels will be too much.

With this option, the vessel draft can be entered.

In case a construction with vessels should be stopped because of the vessel's draft, a warning appears after which three things can be done: the user can wait, until the water level is higher; there could be used other vessels, with a smaller draft; or, the construction could be finished applying another construction method.

5.7 The Window Menu

5.7.1 Opening the Overall Window

When the overall window is opened, on the left side of the screen the graphical presentation of the calculation results will appear. The features of this window have been explained in chapter *4*.

The option *Overall Window* can be chosen from the drop down menu, or with the specific button.

5.7.2 Opening the DataBox Window

The databox window contains the numerical presentation of the closure situation. The meaning of the items in this window have been explained in paragraph ###.

The option *DataBox Window* can be selected from the drop down menu or with the *DataBox button*.

As mentioned in paragraph *5.4.4*, both the *Overall Window* as the *DataBox Window* will pop up automatically when the calculation process is started.

5.7.3 Closing a Window

With this option from the drop down menu, or with the *ALT-F3* key, the highlighted window can be closed. When one of the windows or both windows have been closed, the calculations will be a lot faster. Especially on slower computers this can be useful.

When the *DataBox* window is closed, the size of the *Overall Window* can be increased.

All window operations can be effectuated with the mouse. When the square in the upper left corner is pressed, the window closes; when the mouse touches the window right below, the size can be ajusted.

5.8 The Help Option

The simulation program contains a simple help option. When this option is selected from the main menu, a dialog box appears. After in this box has been selected *OK*, a menu option or button can be chosen. On the chosen topic a help file will show up, with information about the specific function.

6. Running some Closure Examples

6.1 Closure files on the User's Disk

On the user disk of *CLOSIM v 1.0*, three closure examples have been written to file. In this chapter the running of those files have been commented:

1. *SAND1.CTB,* a sand closure,
2. *CABLE1.CTB,* a rock fill closure from a cable way,
3. *STONE1.CTB,* a rock fill closure from dump trucks.

These closure examples can be run from after installation. To do so, go to the *CLOSIM* directory and start the program by entering *CLOSIM <enter>*. After the presentation screen the main menu is shown. From this menu, choose *load situation*, and enter the name of the closure you want to run. After the closure has been loaded, you can start it with the *stopwatch button*, the *F6* key or the *start* option from the drop-down menu. When the process is started, the two windows appear and the closure starts.

1. SAND1.CTB, a sand closure

Sand1 is a closure executed by dumping sand with pipelines from the channel banks into the gap. The closure parameters are:

D50 Bottom material	0.5 mm
D50 Construction Material	0.5 mm
Amplitude Tide M2	0.6 m
Amplitude Tide S2	0.2 m
Length of Bottom Protection	0 m
Gap Dimensions	300 m x 13.6 m
Top Width of Dam	10 m
Top Level of Dam	3 m + MSL

When the process begins, a description of the closure is:

The dumping of sand starts at day # 0. The dam heads advance, and the structure reaches its required height. Everything goes well until the ninth day, when the velocities in the gap increase. From day 10 until day 17 this causes erosion of the bottom material, and a deepening of the closure gap.
Spring tide occurs, and it gets more difficult to close the gap, but from day 14, during neap tide, the closure advances again. The waterlevels in the gap and at sea differ, and the maximum velocity gets to around 2.6 m/s.
Then spring tide occurs again and it is obvious that this way, the gap will not be closed. A different closure method or bigger material sizes should be chosen.

2. *CABLE1.CTB, a cable way closure*

The closure situation written to disk as *Cable1* is the next closure to retrieved. The data consists of:

D50 Bottom material	0.5 mm
D50 Construction Material	500 mm
Amplitude Tide M2	1.5 m
Amplitude Tide S2	0.5 m
Length of Bottom Protection	400 m, bad quality
Gap Dimensions	300 m x 13.6 m
Top Width of Dam	10 m
Top Level of Dam	3 m + MSL
Production Vertical Dumping	1000 m^3/h

During closure, the following occurrances are encountered:

The dumping of material starts. The structure advances fast. After some days, the holes behind the bottom protection start to erode. For each section of the dam this scour hole depth is the same, due to the uniform structure. At a certain stage the holes collapse, and the bottom protection is damaged. Before this damage can threaten the structure though, collapse occurs due to another failure mechanism; at the twelfth day instability of the dam crest occurs. (This phenomena has been described in chapter 5 of volume I of this paper).

3. *STONE1.CTB, a horizontal rockfill closure*

The third example is a horizontal closure with larger granular material, dumped from both channel banks, and defined by the following figures:

D50 Bottom material	0.5 mm
D50 Construction Material	100 mm
Amplitude Tide M2	1.5 m
Amplitude Tide S2	0.5 m
Length of Bottom Protection	400 m, good quality
Gap Dimensions	300 m x 13.6 m
Top Width of Dam	10 m
Top Level of Dam	3 m + MSL
Production from Left Side	1000 m^3/h
Production from Right Side	2000 m^3/h

The running of the closure *Stone1* shows the following:

The closure starts and the dam heads advance. From the fourth day, scour holes occur due to the dam head contraction. Because of the turbulence in the vortex street, locally the velocities are higher. As the dam advances, this scour continues. The bottom protection settles due to collapse of the scour holes, but before this can threaten the structure, the closure is finished.

Appendix A

Structure of Closure Input Data Files

At any moment, a closure situation can be stored on disk using the option *Save Situation*. The *closure data file*, which is written then, allways gets the extension *CTB* and contains all variables which have to do with the actual closure.

The *default file* for the simulation program contains a closure situation, which is loaded always when the program starts. It has been stored on the main disk as *DEFAULT.CTB*. This file could be exchanged for any other closure data file.

The closure data files can be edited with any text editor, although this is not recommended - it is safer to change the files using the simulation program. Viewing those files though, can provide for information on the stored closure situation.

Below, a closure file has been shown as it appears on disk after storing a closure situation. The comments, right of the values, also appear in the files. Here, they have been extended and the units of measuring have been added.

```
CTBFILE          {File identification}                    [-]
1996             {Year of closure, fictive}               [-]
10               {Month of closure, fictive}              [-]
23               {Day of closure, fictive}                [-]
02               {Hour}                                   [-]
38               {Minute}                                 [-]
1                Job Day Number                           [-]
327              Day of Year                              [-]
Sunday           Name of Week Day                         [-]
0.00000          Seconds of Process (did not start)       [s]
0.00000          River Flow (there is no river)           [m3/s]
0.00000          Culvert Area (there is no culvert)       [m2]
50.00000         Chezy Value                              [()]
0.40000          BottomD50                                [mm]
0.00000          Mean Sea Level                           [m]
0.96593          Actual Sea Level                         [m]
0.99144          Old Sea Level                            [m]
0.00000          Old Basin Level                          [m]
0.00000          Actual Basin Level                       [m]
-1947.13819      Total Flow throug Gaps                   [m3/s]
11.44886         Y factor Level Drawing                   [-]
1000.00000       Coord X Level Drawing                    [-]
    0            Kenterings Till Now                      [-]
3.00000          Maximum Storm Surge                      [m]
24.00000         Storm Duration                           [h]
0.00100          Chance of Storm                          [-]
```

```
0.00000          Draft of Vessels                          [m]
1.00000          Tidal Amplitude Factor                    [-]
16000.00000      Tidal Phase at T=0                        [s]

0.00000          Tidal Amplitude Factor                    [-]
4000.00000       Tidal Phase at T=0                        [s]
0.00000          Tidal Amplitude Factor                    [-]
8000.00000       Tidal Phase at T=0                        [s]
0.00000          Tidal Amplitude Factor                    [-]
2000.00000       Tidal Phase at T=0                        [s]
1.00000          Max Amplitude Possible                    [m]
4416.36364       Gap Area in Equilibrium                   [m2]

-19.72174        Original Maximum Depth                    [m]
300.00000        Original Total Width                      [m]
0.43385          Drawing Width Factor                      [-]
-1               Actual Flow Direction                     [-]
-1               Last Flow Direction                       [-]
-100.00000       Level in Basin                            [m]
5000000          ==> gives Area in Basin                   [m2]
-60.00000        Level in Basin                            [m]
12000000         ==> gives Area in Basin                   [m2]
-10.00000        Level in Basin                            [m]
24000000         ==> gives Area in Basin                   [m2]
10.00000         Level in Basin                            [m]
80100000         ==> gives Area in Basin                   [m2]
100.00000        Level in Basin                            [m]
90000000         ==> gives Area in Basin                   [m2]
1                Number of Gaps to Close                   [-]
0                Way of Construction                       [-]
-1               Actual Weir Situation                     [-]
1                Construction Status                       [-]
0                Once Cable Way?                           [-]
300.00000        Original Width of Gap                     [m]
300.00000        Actual Width of Gap                       [m]
-14.72174        Original Depth of Gap                     [m]
4416.52217       Original Gap Area                         [m2]
-1947.13819      Actual Gap Flow                           [m3/s]
-0.41428         Actual Velocity in Gap                    [m/s]
1.00000          Contraction in Gap                        [-]
0.00000          Volume Dumped   Left                      [m3]
0.00000                          Right                     [m3]
0.00000                          Vertically                [m3]
1                Process Left                              [-]
1                Process Right                             [-]
0.00000          Actual Volume of Dam                      [m3]
0.00000          X left of Apron                           [m]
300.00000        X right of Apron                          [m]
-14.72174        Apron Level                               [m+MSL]
-14.72174        Dam Level Left                            [m+MSL]
-14.72174        Dam Level Right                           [m+MSL]
-14.72174        Actual Gap Depth                          [m+MSL]
25.00000         Top Width of Dam                          [m]
5.00000          Required Top Level                        [m+MSL]
```

```
352858.23082     Required Dam Volume                          [m3]
19.72174         => Required Dam Height                       [m+MSL]
0.51757          Slope of Dam Head                            [rads]
0.51757          Slope of Section                             [rads]
1000.00000       D50 of Dam Material                          [mm]
0.00000          Speed Closure Left                           [m3/h]
0.00000                        Right                          [m3/h]
100.00000        Level Limit Vertical                         [m+MSL]
2.00000          Vertical Inaccuracy                          [m]
0.00000          Speed Vertical closure                       [m3/h]
10.00000         Drawing of Sections; X                       [-]
600.00000        Length of Protection                         [m]
15.00000         Length of BP section                         [m]
0.00000          Lately Skipped Left                          [m3]
0.00000          Lately Skipped Right                         [m3]
15.00000         Width between Scour SC                       [m]
0.00000          Maximum Scour Outside                        [m]
0.00000          Maximum Scour Inside                         [m]
0.00000          Place Max Scour Inside                       [m]
0.00000          Place Max Scour Outside                      [m]
20               Number of Scour Snaps                        [-]

{FOR ALL WIDTH SECTIONS: 20 in this case}

    0.00000          Scour Hole Inside                       [m]
    0.00000          Reduced Scour                           [m]
    0.31028          Last Scour Depth                        [m]
    0.00000          Scour Hole Outside                      [m]
    0.00000          Reduced Scour                           [m]
    0.43152          Last Scour Depth                        [m]
    20.00000         Last protect   OUT                      [-]
    20.00000         Last protect    IN                      [m]
    157.19704       Scour Integral                           [()]

        {FOR ALL BOTTOM PROTECTION SECTIONS: 20 x 20}

        0.00000  Protection Level InFl                       [m]
        0.00000  Protection Level OutFl                      [m]

{CONTINUED GAP INFORMATION, FOR ALL GAPS (MAX. THREE)}

4416.52217       Actual Gap Area                              [m2]
4416.52217       Original Total Area                          [m2]
72870000         Volume of Tidal Prism                        [m3]
238.53036        Square Velocity Integral (for angles)        [m2/s2]
0.00000          Velocity^5 Integral (reducing transports)[m5/s5]
4000.00000       Time since Last equal levels                 [s]

{END OF FILE}
```

Appendix B

Structure of Log Files

As explained in chapter 5, the *log files* contain the results of the calculations, which are written to disk to be retrieved by any other program. This way, a closure can be reconsidered afterwards.

The structure of the log files is quite simple. In any spreadsheet, the results can be imported as *blank- and comma separated data file*. If the output should be used in other programs, here the structure of the files is given. For every log file, a graphical spread sheet is presented as well, of one of the cases presented in chapter *6* of this manual:

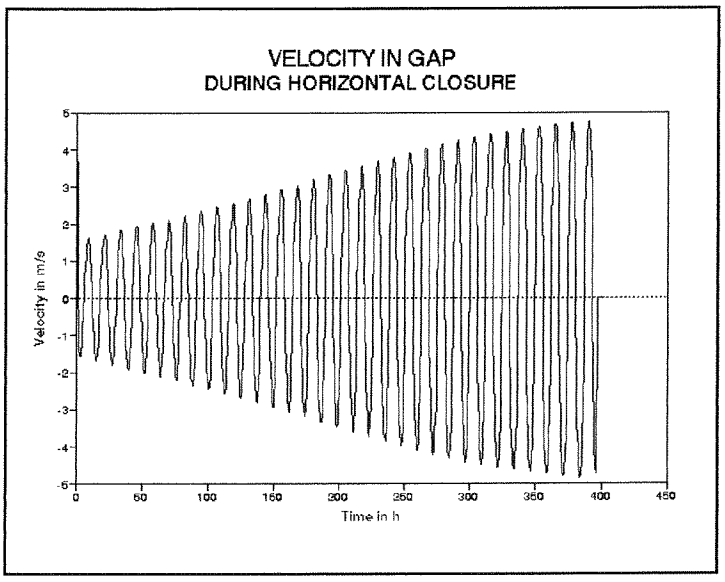
---

1. The velocity data in VELOCITY.LOG

The sequence of the data in the file is:

Time (in hours) - *Gap Surface (m^2)* - *Current Velocity (m/s)*

If this log file for a *horizontal closure* is loaded into a spread sheet, figures like the one below are obtained:

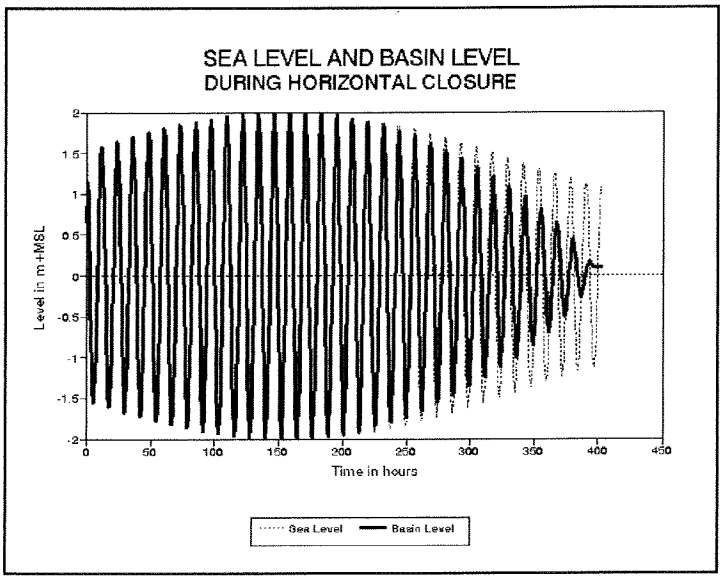


2. The level data in LEVELS.LOG

The information on the levels is written as:

Time (in hours) - Sea Level (m+MSL) - Basin Level (m+MSL)

A spread sheet turns this log file into a figure like shown below:



It can be seen, that the closure is finished after spring tide has occurred.

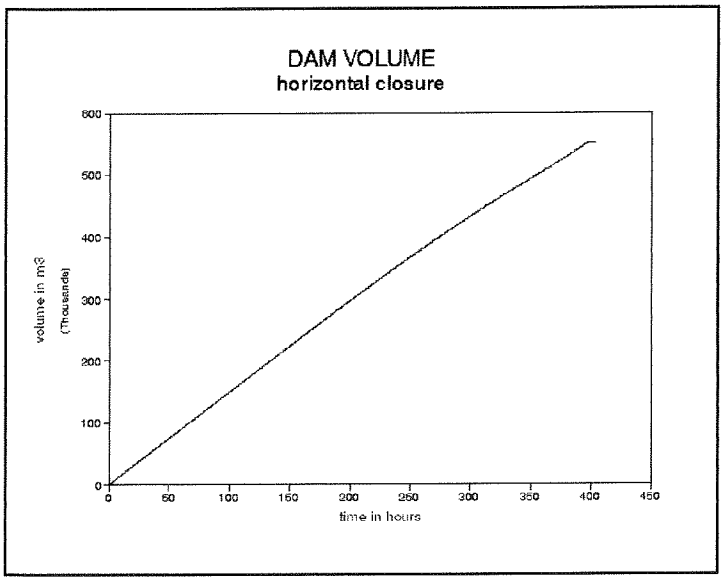
3. The Production and Erosion information in EROSPROD.LOG

The information on production and erosion is written:

Time (in hours) - Total Dumped (m^3) -

Total Eroded (m^3) - Actual Volume (m^3)

In a spreadsheet this log file has been printed for horizontal closure. Because of the big grain sizes of the material, no erosion has taken place, and the dam volume is equal to the volume of the dumped material:

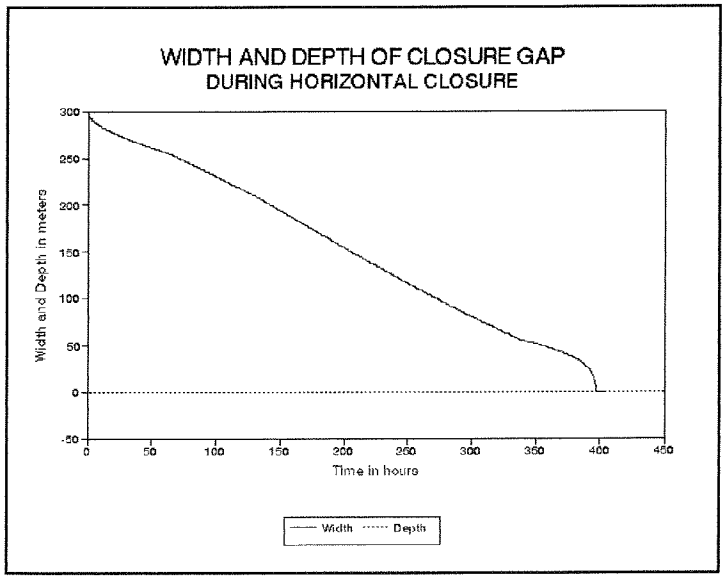


4. The Gap dimensions in GAPDIMEN.LOG

The dimensions of the gap to be closed are written as:

Time (in hours) - *Width of gap (m)* - *Depth of gap (m)*

For a horizontal closure, the spreadsheet results in the following graph:



It can be seen, that for horizontal closure the width reduction is linear with the time - big grain sizes are used and there is barely erosion. The variations that take place are in the first and last stage of construction, when the width is reduced more than linear because of geometry of the dam advance; in the full stage, there is some variation of the width with the water levels.

Appendix C

Quick User's Reference

INSTALLATION:

- *Enter Disk in drive A:*
- *type A:\INSTALL*

REQUIRED FILES:

- *CLOSIM.EXE* *All in directory CLOSIM (Optional)*
- *SIMFREE.DSK*
- *DEFAULT.CTB*
- *CLOSIM.BMP*

STARTUP:

- *Change to directory C:\CLOSIM*
- *type CLOSIM [possible with closure name + .CTB]*

USER INTRODUCTION

- *Main menu INFO / INTRO for introduction or use Question Mark Button*
- *Main menu PROCESS / START to start calculations or use F6 or Stopwatch Button*

MAIN MENU

- *INFORMATION*
  - *ABOUT [information about program and version]*
  - *INTRO [introducing the user to the program]*
- *FILE*
  - *SAVE [save a closure situation to disk]*
  - *LOAD [load a closure situation from disk]*
  - *CHANGE DIRECTORY [change main directory]*
  - *DOS SHELL [temporally to DOS]*
  - *QUIT [quit the program, go to shell]*
- *PROGRAM*
  - *TOGGLE SOUND [sound on/off]*
  - *SELECT COLORS [set color palette]*
  - *LOG FILES [select which log files]*
  - *LOAD DESKTOP [retrieve a desktop file]*
  - *SAVE DESKTOP [store a desktop file]*

- *PROCESS*
  - *LOAD DEFAULT [load default values]*
  - *NEW CLOSURE [enter data for new closure]*

  - *INIT PROCESS [set process to initial state]*
  - *START PROCESS [start the calculations]*
  - *INTERRUPT PROCESS [interrupt calculations]*

- *BASIN*
  - *GENERAL [# gaps, Chezy, D50 bottom material]*
  - *DATE AND TIME [date and time at closure]*
  - *STORAGE AREA [storage function of basin]*
  - *DEPTHS AND WIDTHS [gap dimensions]*
  - *TIDAL CONDITIONS [amplitudes / phase(0)]*
  - *RIVER FLOW [river flow in m^3/s]*
  - *DEFINE STORM [set storm conditions]*
  - *START STORM [start storm now]*

- *CONSTRUCTION*
  - *METHOD [trucks, vessels, cable way, pipe lines]*
  - *PROTECTION [length & quality of protection]*
  - *DIMENSIONS [dam height and top width]*
  - *ANGLES [side and head angles of dam]*
  - *GRADUAL CLOSURE [materials and quantities]*
  - *CULVERT [constructing a spillway]*
  - *VESSEL DRAFT [required depth for vessels]*

- *WINDOW*
  - *OVERALL WINDOW [graphical presentation]*
  - *DATABOX WINDOW [numerical presentation]*
  - *CLOSE WINDOW [closing the actual window]*

- *HELP*
  - *HELP ON OPTIONS [help on any feature]*

CONTENTS OF THE DATABOX WINDOW

Mnemo	*Parameter*	*Value / Units*
Job day #	# of days closure is going on	---
Hsea	Actual sea level	m + MSL
Qt	Total flow through gaps	m^3/s
As	Total flow area of gaps	m^2
Setup	Actual storm setup	m
Dur	Time left to storm	s
%prf	Percentage of gap closed	%
Qgap	Flow through gap	m^3/s
Vgap	Velocity in gap	m/s
Agap	Flow area of gap	m^2
Weir	Situation of flow on weir	CRITical / NORMal
Lvpr	Level of toes	m + MSL
Wgap	Width of gap	m
Hdam	Actual height of dam	m
Vreq	Required volume for dam	m^3
Vdum	Total dumped up till now	m^3
Vero	Total eroded up till now	m^3
Vprf	Actual volume of dam	m^3
Hole	Depth of greatest scour hole	m
Eqpm	Kind of equipment used in gap	TRUCks, VESSels, CABLe way, PIPE lines
Prod	Total production in gap	m^3/h
Size	Size of material particles	m