Fate of pesticides in field ditches: the TOXSWA simulation model

P.I. Adriaanse

Report 90

DLO Winand Staring Centre, Wageningen (The Netherlands), 1996

ABSTRACT

Adriaanse, P.I., 1996. *Fate of pesticides in field ditches: the TOXSWA simulation model* Wageningen (the Netherlands), DLO Winand Staring Centre. Report 90 241 pp.; 25 Figs; 3 Tables; 19 Annexes.

The TOXSWA model describes the fate of pesticides entering field ditches by spray drift, atmospheric deposition, surface runoff, drainage or leaching. It considers four processes: transport, transformation, sorption and volatilisation. A sample simulation showed that sorption to macrophytes can considerably reduce the concentration in the water phase. However, sorption to macrophytes leads to a slower transport and thus to longer residence times for the pesticide in the water layer. Sedimentation and resuspension of suspended solids were not considered, so that the model can only be applied for periods shorter than one month.

Keywords: environmental protection, pesticide concentration, surface water

Projects 7242 and 597. [REPO90.EVR / 04-96]

Contents

	page
Preface	9
Summary	11
1 Introduction	15
2 System description	17
2.1 Introduction and system definition	17
2.2 Processes and suspended solids behaviour	19
3 Mass balances for water layer and sediment	23
3.1 Water layer	23
3.2 Sediment	26
4 Basic processes in water layer and sediment	31
4.1 Water layer	31
4.2 Sediment	35
5 Boundary conditions and initial condition, other input	41
5.1 Water layer	41
5.2 Sediment	45
6 Numerical solution of the mass conservation equations	47
6.1 Introduction	47
6.2 Numerical weight factors	49
6.3 Numerical dispersion	54
6.4 Water layer	56
6.5 Boundary conditions and initial condition for water layer	66
6.5.1 Lower boundary condition	66
6.5.2 Upper boundary condition	67
6.5.3 Initial condition	68
6.6 Sediment	69
6.7 Boundary conditions and initial condition for sediment	76
6.7.1 Upper boundary condition	76
6.7.2 Lower boundary condition	77
6.7.3 Initial condition	78
6.8 Iterative calculations caused by the Freundlich equation for sorption	78
6.9 Coupling the water and sediment subsystems	81
7 Verification	83
7.1 Introduction	83
7.2 Stability condition	85
7.2.1 Description of the positivity for the water layer	85
7.2.2 Restrictions resulting from requirements for positivity for the water layer	86

7.2.3 Restrictions resulting from requirements for positivity for the sediment 91
7.3 Consistency condition 96
7.3.1 Introduction 96
7.3.2 Analytical solutions of the Convection-Dispersion Equation 97
7.3.3 Application of the analytical solutions of the Convection-Dispersion Equation to the water and sediment subsystems of TOXSWA 99
7.3.4 Comparison of the analytical solutions with the numerical solutions (i.e. TOXSWA) 102

8 Illustration of the potential use of TOXSWA: simulation of chlorpyrifos behaviour after spray drift deposition 107
8.1 Design of computation and values of parameters 107
8.2 Results and discussion 107

9 Conclusions, discussion and recommendations 113
9.1 Conclusions and discussion 113
9.2 Recommendations 114

References 117

List of symbols 121

Tables

1 Overview of variables in the conservation equations for the water and sediment subsystems, demonstrating at which location in the grid these variables are defined 49
2 Outline of weight factors introduced according to space, using the finite-difference method in TOXSWA 53
3 Total pesticide concentrations in node 1(z = 0.0005 m) of the sediment as a function of time, calculated with the aid of the analytical solution for the sediment, Eq.(7.39) 103

Figures

1 Possible entry routes of pesticides to a field ditch 16
2 Outline of model system 18
3 Definition of coordinate system 18
4 Diagram of modelled processes 20
5 Mass balance at times t and $t + \Delta t$ 23
6 Cross section of a ditch with the shape of stacked trapezia 27
7 The sediment subsystem simplified to one dimension 27
8 The sediment subsystem situated below an area of constant size, through which exchange of pesticide between water layer and sediment occurs 28
9 Mass balance for the sediment subsystem 29
10 Calculation of wetted perimeter in the sediment subsystem 37

11 Four situations in the sediment, depicting the possible combinations of the directions of the advection flow ($\ell q/P\varepsilon$) and the concentration gradient $\partial c_{lb}/\partial z$ 38
12 Dirac delta function for pulse input 43
13 Dirac delta function for point-type input 43
14 Outline defining the discretisation of the x axis (water layer) 47
15 Outline defining the discretisation of the z axis (sediment) 48
16 Linear interpolation to determine the concentration at the interface, $c_{i-½}$ with the aid of the concentrations at grid points i-1 and i, c_{i-1} and c_i respectively 51
17 The function $g_{i-½}$, with ß representing a weight factor of space and Δx_{i-1} and Δx_i the slope parameters 52
18 Detail of the TOXSWA field ditch system, demonstrating the concept of one water subsystem coupled to many sediment subsystems 82
19 Outline of the domain (u, β) for which the condition $LDD > 0$ (part(a)) or $RDD > 0$ (part(b)) result in a restriction for the time step, Δt 90
20 Outline of the domain (q, β) for which the condition $LDD > 0$ (part (a)) or $RDD > 0$ (part(b)) result in a restriction for the time step, Δt 95
21 Comparison of the total chlorpyrifos concentration in the sediment calculated with the aid of the TOXSWA model (indicated by the markers) and with the aid of the analytical solution for the mass conservation equation of the sediment, Eq.(7.39) (indicated by the drawn lines) 104
22 Comparison of the chlorpyrifos concentration in the water phase of the ditch calculated with the aid of the TOXSWA model (indicated by the markers) and with the aid of the analytical solution for the mass conservation equation of the water layer, Eq.(7.43) (indicated by the drawn lines) 105
23 Chlorpyrifos concentration in the water layer and at selected locations in the sediment after spray drift deposition of 0.03 $kg.ha^{-1}$ onto the ditch 110
24 Distribution of chlorpyrifos between the different compartments as a function of time for the entire ditch (200 m) as well as per running metre at selected locations. 111
25 Mass balances of chlorpyrifos as a function of time for the entire ditch (200 m). Separate mass balances are shown for the water and sediment subsystems 112

Annexes

1 The elements *LOD, LDD, LBD, ROD, RDD, RBD* and *RV* from Eq.(6.26) for the grid points m +1 up to ebt -1 inclusive in the end buffer of the water subsystem 125
2 The elements *LOD, LDD, LBD, ROD, RDD, RBD* and *RV* from Eq. (6.26) for the grid point $eb = ebt$ of the end buffer of the water subsystem. The flow velocity u is positive 129
3 The elements *LOD, LDD, LBD, ROD, RDD, RBD* and *RV* from Eq. (6.26) for the grid point $eb = ebt$ of the end buffer of the water subsystem. The flow velocity u is negative 133

4 The elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD*, *RBD* and *RV* from Eq. (6.26) for the grid point i = 1 in the water subsystem in the case of a positive flow direction 137
5 The elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD*, *RBD* and *RV* from Eq. (6.26) for the grid point *fb* = 2 up to *fb* = *fbt* inclusive of the front buffer of the water subsystem. The flow velocity alternates between being positive and negative 141
6 The elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD*, *RBD* and *RV* from Eq. (6.26) for the grid point *fb* = 1 in the case of a negative flow velocity in the situation of alternating positive and negative flow velocities in the water layer 145
7 The elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD*, *RBD* and *RV* from Eq. (6.26) for the grid point *fb* = 1 in the case of a positive flow velocity in the situation of alternating positive and negative flow velocities in the water layer 149
8 The elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD*, *RBD* and *RV* from Eq. (6.39) for the grid point k = 1 of the sediment subsystem. So, this is the upper boundary condition 153
9 The elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD*, and *RBD* from Eq. (6.39) for the grid points n + 1 up to *ebbt* - 1 inclusive from the end buffer at the lower end of the sediment 157
10 The lower boundary condition at the grid point *ebbt* in the case of downward water flow in the sediment 161
11 The lower boundary condition at the grid point *ebbt* in the case of upward water flow in the sediment 165
12 Structure of the TOXSWA program 169
13 Description of the TOXSWA model including subroutines 171
14 Outline proving the positivity and stability of the matrix equation $L.c^{n+1} = R.c^n$, with L and R as the two tridiagonal matrices of Eqs. (6.26) or (6.39) 175
15 Changes in the source code of TOXSWA 1.0 for comparison with the analytical solutions for the water layer and the sediment 177
16 Changes in the input files (compared to those of the example simulation for chlorpyrifos) for calculating TOXSWA output for comparison with the analytical solution for the sediment 181
17 Changes in the input files (compared to those of the example simulation for chlorpyrifos) for calculating TOXSWA output for comparison with the analytical solution for the water layer 185
18 Input files for the example simulation for chlorpyrifos 189
19 Source code of the TOXSWA program, version 1.0, including a guide to the vocabulary used 201

Preface

Predicting concentrations at which levels negative effects on aquatic organisms occur requires a methodology to predict exposure concentrations for these organisms. In 1991 the Directorate of Science and Transfer of Knowledge of the Dutch Ministry of Agriculture, Nature Management and Fisheries asked the DLO Winand Staring Centre (SC-DLO) to develop such a methodology. The project 'Modelling transport routes and fate of pesticides in soil and field ditches' started at SC-DLO at the end of 1991. The general administrative order on Environmental Admission Requirements for Pesticides ('AMvB-3a') (Ministry of Housing, Spatial Planning and Environment, 1995) mentions that the TOXSWA model is to be included in the future risk assessment procedure in the Netherlands.

In the first phase of this project P. Groenendijk and J.W.H. van der Kolk made an inventory of all relevant processes in field ditches. In August 1992 the project team was joined by P.I. Adriaanse, who focused specifically on the development of the TOXSWA simulation model (TOXic substances in Surface WAters), which describes the fate of pesticides in field ditches. On November 8, 1994 an international workshop was organised at the DLO Winand Staring Centre, Wageningen, the Netherlands, where aquatic fate modelling, the TOXSWA model and related pesticide regulations in the Netherlands were presented and discussed (Crum and Deneer (eds), 1995). About eighty specialists from eight different countries attended seven lectures and contributed to the discussion. By the end of 1995, W.H.J. Beltman joined the project team who wrote the user's manual for TOXSWA 1.0 and is in charge of defining 'standard scenarios' for the Netherlands.

From the end of 1991 up to mid-1995 the project work took place within the framework of project 7242 of the DLO programme 147 'Ecotoxicological Risks of Pesticides in Aquatic and Terrestrial Ecosystems', financed by the Dutch Ministry of Agriculture, Nature Management and Fisheries. From mid-1995 onwards the project was financed both by the SC-DLO (SEO-project 597) and the Ministry mentioned above (project 592 of DLO programme 276 'Emissions and Ecotoxicological Hazards of Pesticides').

Several persons outside the project team have contributed substantially to the development of the TOXSWA model. The author is most grateful to J.J.T.I. Boesten of the department Fate and Effects of Pesticides at SC-DLO, who provided many important ideas, reflections and criticisms all along the way. Special thanks are due to R.H. Aalderink of the department of Water Quality Management and Aquatic Ecology of the Wageningen Agricultural University; he provided ideas for the numerical solution and commented on it. J.A.P. Heesterbeek of the DLO Agricultural Mathematical Group contributed substantially to the mathematical background of Chapter 7. The 'walk through team', headed by M.J. van der Velden, the Software Quality Manager of SC-DLO, gave useful comments and ideas for improving the TOXSWA computer program.

Model development was critically monitored by a consultation group, consisting of representatives of the Department of Plant Protection (H. de Heer and A.C.P. van Montfort), of the Plant Protection Service (W.W.M. Brouwer), all from the Ministry of Agriculture, Nature Management and Fisheries, of the Toxicology Advisory Centre (J.B.H.J. Linders) and the Laboratory of Soil and Groundwater Research (A.M.A. van der Linden), both at the National Institute of Public Health and Environmental Protection, of the Institute for Inland Water Management and Waste Water Treatment (F. Wagemaker/P.C.M. van Noort) and of SC-DLO (P.E. Rijtema/P. Leeuwangh and P. Groenendijk).

The TOXSWA model development (and its verification) is described in this report. It treats the mathematical model, the numerical solution and the resulting computer model. Some first calculation results are presented as well.

Research on the TOXSWA model will continue at SC-DLO in the coming years. Sorption of a range of pesticides to three macrophyte species is being studied, as little information is available on this sorption parameter. In 1996 and 1997, the sensitivity of the model to input parameters and the initial conditions will be analysed. Model improvements will be implemented, initially in two areas: (i) inclusion of multiple or continuous pesticide applications to the water layer and (ii) inclusion of time series with varying water flow rates and water depths. Furthermore, standard scenarios for the Netherlands will be defined to facilitate the use of the model for risk assessments in the Dutch registration procedure. Model validation will start, using data sets from four experiments carried out in microcosms (1 m^3) and in outdoor ditches (60 m^3) by the DLO Winand Staring Centre (Crum and Brock, 1994; Aalderink and Crum, 1994; Crum et al., in prep.). In addition, data sets from experiments performed elsewhere will be assessed for their potential to validate the TOXSWA model. Field experiments in other EU countries are needed to assess whether the TOXSWA model is able to simulate fate in watercourses outside the Netherlands. The TOXSWA model is being assessed by the surface water fate group of the EU working party called FOCUS. Process descriptions need to be worked out in greater detail, including transformation rates and the effect of environmental parameters on them, or sorption to macrophytes. Finally, more experiments are needed to estimate the dispersion coefficient in various ditches under various conditions. Very few data exist for small watercourses, whereas the dispersion coefficient is very important for estimating the concentration curve. Results of these studies will be reported in future publications.

Summary

The TOXSWA model (TOXic substances in Surface WAters) has been developed to estimate exposure concentrations to pesticides of aquatic organisms in the field. The exposure concentrations are needed during the admission procedure of pesticides on the Dutch market, they are compared with laboratory toxicity data for selected standard organisms to evaluate the risks of agricultural use of pesticides for aquatic ecosystems.

The TOXSWA model describes the behaviour of pesticides in field ditches. The modelled field ditch system is two-dimensional and consists of two types of subsystem, water layer and sediment. In the water layer, concentrations vary only in horizontal direction, while in the sediment, concentrations vary both in horizontal and vertical directions. It can handle a variety of situations as regards hydrological conditions and entry routes of pesticides into surface water.

TOXSWA can be coupled to other models describing pesticides entering ditches via (i) drift or atmospheric deposition, (ii) surface runoff, or (iii) drainage or leaching through the soil.

In principle, TOXSWA can handle various hydrological conditions, so it simulates ditches with varying water depths h and rates of discharge Q.[1] The cross section of the ditch is trapezium-shaped; upward or downward seepage takes place through the ditch bottom and walls. Upward or downward seepage limits and enhances, respectively, the penetration depth of the substance into the sediment. No resuspension or sedimentation of suspended solids occurs in the water layer.

TOXSWA considers four processes: (i) transport, (ii) transformation, (iii) sorption and (iv) volatilisation. In the water layer, pesticides are transported by advection and dispersion, including transport of pesticides sorbed to suspended solids. In the sediment pesticides are transported by diffusion as well. The transformation rate covers the combined effects of hydrolysis, photolysis and biodegradation; metabolites are not considered. Sorption to suspended solids and to sediment is described using the non-linear Freundlich equation. Sorption to macrophytes is described using a linear isotherm. Pesticides are transported across the water-sediment interface by advection (upward or downward seepage) and by diffusion.

A pesticide mass balance is set up for an elemental volume in the water and in the sediment subsystem. The mass balances for the water and sediment subsystems account for incoming and outgoing mass fluxes. They result in two partial differential equations: the conservation equations for the water layer and the sediment.

[1] TOXSWA version 1.0 implements a constant water depth and discharge; these can be freely chosen.

Different types of input of pesticide into the water subsystem are possible: (i) distributed pulse input, e.g. spray drift or a momentary runoff, (ii) point-type pulse input, e.g. spillage of pesticide or a brief release from an individual drain, (iii) continuous point release, e.g. discharging tributary or (iv) continuous distributed release e.g. continuous release from many nearby drains. The water and sediment subsystems are coupled by assuming that the concentration in the liquid phase at the sediment surface equals the water phase concentration of the overlying water column. At the bottom of the sediment there is an inflow or outflow of water with pesticide.

The conservation equations for the water layer and the sediment are solved using a variable weight finite-difference method. This implies that spatial derivatives of the conservation equations may be approximated by e.g. a forward, central or backward difference. Temporal derivatives may be approximated in an implicit or explicit way, or as a weighted average of these two extremes. As processes in one subsystem may be more dynamic than in the other, different sizes of time steps are implemented for the water layer and the sediment.[2] The approach of the variable weight finite-difference method allows for an optimal computation time by adapting selected time and space steps to acceptable numerical dispersion. The numerical solution introduces numerical dispersion. In TOXSWA, the physical dispersion is corrected for this numerical dispersion. The approximations of the two conservation equations result in two matrix equations, which are solved for specified initial and boundary conditions. This allows the solution vector, consisting of the pesticide concentration, to be found. The TOXSWA model does not solve the conservation equations for the water and sediment subsystems simultaneously. The subsystems are linked by assuming that the concentration in the liquid phase calculated at time t can be used for calculating the flux at the sediment - water interface one time step later (at time $t+\Delta t$).

Verification is defined as the examination of the numerical technique in the computer model to ascertain that it truly represents the mathematical model and that there are no inherent numerical problems with obtaining a solution. This concept is worked out using the notions of convergence, stability and consistency. Convergence states that when the finite-difference grid is refined the truncation errors go to zero. Stability concerns the unstable growth or stable decay of errors in the arithmetic operations needed to solve the finite-difference equations. This condition is translated into a positivity condition, yielding conditions for the time and space steps resulting in stable and positive solutions. Consistency is the requirement that when the finite-difference grid is refined the truncation errors go to zero, but moreover that the finite-difference model approximates the partial differential equation desired and not some other partial differential equation. This is explained for the two conservation equations. It is also checked that the introduced mass of pesticide (100%) can be traced at every moment, so that the mass balances tally and that TOXSWA conserves well the introduced mass in the system. Furthermore, a model simulation is found to correspond very well with an analytical solution for a water subsystem, which

---

[2] In TOXSWA version 1.0 the numerical weight factors are fixed on values yielding an explicit central difference calculation scheme. The time steps for the water layer and the sediment also need to be similar.

responds to a Dirac delta function-type input. Correspondence between a model simulation and an analytical solution for the sediment subsystem responding to the same type of input was also excellent.

To illustrate the possibilities of the TOXSWA model, a computation was carried out for the insecticide chlorpyrifos. In total 2 g chlorpyrifos was deposited on a ditch of 200 m long and 3.65 m wide at its water surface. The ditch contained suspended solids and macrophytes and had a water flow rate of 100 m/d. Chlorpyrifos sorbs strongly to organic matter and to macrophytes. Simulation was run for four days. Initially, 53% of the 2 g chlorpyrifos was dissolved in the water phase, 42% was adsorbed to the macrophytes and 4% to the suspended solids. After four days, most of the pesticide had flowed out of the ditch, but of the remaining mass (0.1 g) 72% had penetrated into the sediment, 15% was dissolved in the water phase, 12% was adsorbed to the macrophytes and 1.3% to the suspended solids. The mass balance for the water layer showed that, besides outflow, volatilisation, penetration in sediment and transformation were the most important factors (in order of priority) contributing to the pesticide decrease in the water layer.

The TOXSWA model calculates the average concentration to which aquatic organisms are exposed at 3, 21 and 28 days after application, as well as immediately after application (at 0 d). They represent the average of the concentration course for pesticide dissolved in the water phase with time. The exposure concentration at time $t = 0$ d corresponds to the total pesticide concentration, immediately after application. The exposure concentration has been defined as the concentration at that position in the ditch where the longest exposure duration is expected, i.e. at the downstream end of the section of the ditch where the pesticide input took place.

The TOXSWA model checks whether the mass balances tally during the calculations. Results for the example simulation for chlorpyrifos showed that after four days the missing quantity in the mass balances for both the water layer and the sediment was less than 0.005% of the initial mass (plus, for the sediment, incoming mass of the water layer). So, this shows that TOXSWA 1.0 conserves well the mass of pesticide applied.

In the TOXSWA model the assumptions are made that the pesticide mixes instantaneously across the cross section and that sorption equilibrium is also instantaneous. Measurements for chlorpyrifos performed during earlier experiments of SC-DLO showed, however, that both assumptions do not correspond with reality. This may result in pesticide concentrations simulated in the water phase being too low, especially during the first one or two days.

The model has been developed to simulate periods of up to about one month; moreover, as no resuspension and sedimentation of suspended solids is included, this version of the model is not suitable for calculating long term exposure concentrations in the sediment or accumulation of pesticide in the sediment.

1 Introduction

The Dutch Pesticide Act of 1975 requires an evaluation of the hazards of pesticides, with regard to public health as well as the environment, before pesticides can be registered. In 1991, the Dutch Government began to implement the Multi-Year Crop Protection Plan, which aims to produce a considerable reduction in the dependence on and the use of pesticides in the Netherlands, as well as to reduce the emission of pesticides to the environment. Stricter rules for admission of pesticides are among the measures mentioned in this plan to alleviate side-effects of pesticides on the environment. In 1995 the Dutch Government issued a so-called general administrative order ('AMvB-3a') in which stricter rules concerning leaching to groundwater, persistence in the soil and toxicity for aquatic organisms were specified. Risk evaluation for aquatic organisms is based upon comparison of the estimated exposure concentration in the field with laboratory toxicity data.

At present, a relatively simple model, called SLOOT.BOX (Linders et al, 1990) is used to predict exposure concentrations in field ditches. This model is now being used by the Dutch Board for the Authorization of Pesticides to predict the exposure concentration for short term effects on aquatic organisms. Differentiation of admission of pesticides according to regional characteristics, or realistic prediction of chronic exposure of aquatic organisms (i.e. up to 28 days) requires a more detailed model. This should take all relevant processes into account and should consider not only spray drift deposition but also other entry routes of pesticides to ditches (e.g. runoff, drainage, leaching and atmospheric deposition) (Fig. 1).
The objective for the development of the TOXSWA model has been to provide the Board for the Authorization of Pesticides with an improved tool for the estimation of pesticide concentrations in field ditches. Acute (up to 4 days) as well as chronic (up to 28 days) exposure of aquatic organisms need to be predicted in a realistic way if one is to account for regional differences.

The present report provides a detailed description of the TOXSWA model development with a limited verification of the implemented computer program. The TOXSWA model includes all processes relevant for a description of the fate of pesticides in field ditches; it considers various entry routes of pesticides like spray drift deposition, seepage or point-type inputs. This makes the TOXSWA model a suitable tool for producing realistic estimations of the exposure concentrations, differentiated in time and space, for aquatic organisms in field ditches.

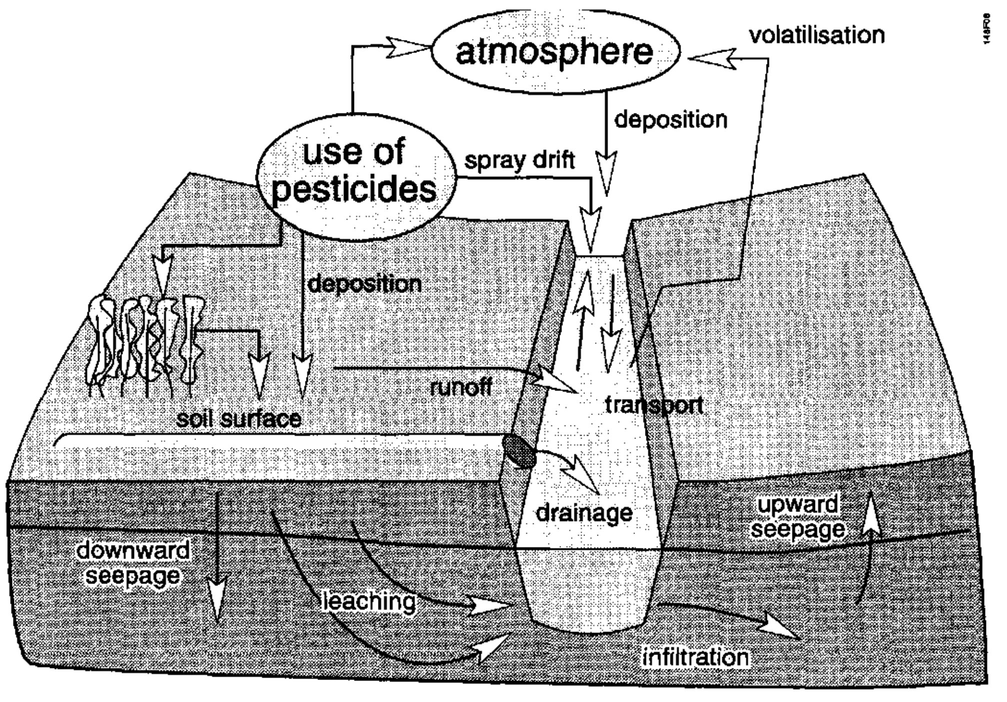


Fig. 1 Possible entry routes of pesticides to a field ditch

In Chapter 2 the modelled system and model assumptions are described, while Chapter 3 draws up mass balances for water layer and sediment. Chapter 4 discusses the processes that appear in TOXSWA and Chapter 5 specifies boundary and initial conditions. In Chapter 6 the differential equations expressing the mass balances are solved with a variable weight finite-difference method, while Chapter 7 verifies the implementation of the resulting computer program. Chapter 8 gives some early computation results and discussions. Finally the conclusions, discussion and recommendations are formulated in Chapter 9.

2 System description

2.1 Introduction and system definition

The behaviour of pesticides in field ditches, including their sediments, is described in this section. Pesticides can end up in field ditches because of spray drift, runoff from neighbouring field lots or by releases. In a ditch with a draining function, pesticides can also enter the ditch water by upward seepage through the ditch bottom. In the water, pesticides sorb to dissolved particles, to suspended solids and to macrophytes. In the sediment, pesticides sorb to the solid bottom material. In the course of time, pesticides degrade.

The field ditch system is divided into two subsystems, water layer and sediment. Mass balances are drawn up for both subsystems. These result in two partial differential equations, which are coupled to each other and in principle need to be solved simultaneously. These equations are solved with the aid of numerical calculation techniques and for specified initial and boundary conditions. These solutions form the basis for the resulting computer program.

The field ditch system is characterised in the following way (Fig. 2).
- The cross section of the ditch is trapezium-shaped and the wetted perimeter separates the water layer from the neighbouring soil.
- Discharge and water level in the ditch vary in time and space.
- Upward and downward seepage occur through the bottom and walls of the ditch.

The coordinate system is defined in Figure 3.

x axis: positive in the most frequent direction of flow in the ditch;
z axis: positive with depth, zero at the sediment surface;
y axis: perpendicular to the *x* axis and the *z* axis.

Other characteristics of the system follow below.
- There is a concentration gradient of the substance at issue in the *x* direction of the water layer; there are no concentration gradients in the *y* direction or the *z* direction.

  This means that the **assumption** has been made that the substance is ideally mixed, vertically and horizontally, in the water layer.

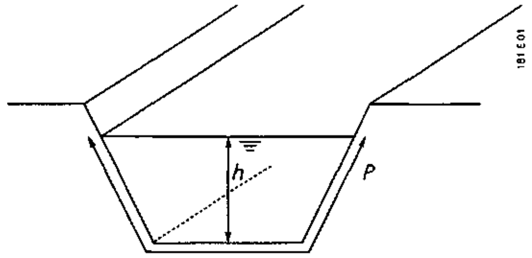


Fig. 2 Outline of model system

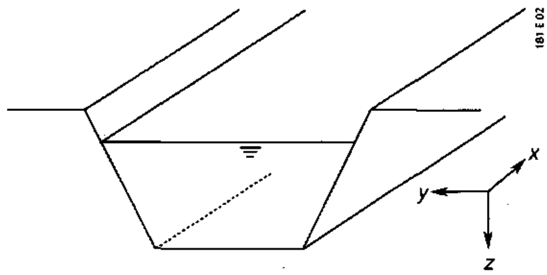


Fig. 3 Definition of coordinate system

— There is a concentration gradient of the substance in the z direction of the sediment as well as in the x direction. This implies that there is no concentration gradient in the y direction.
This means that it has been **assumed** that the substance is ideally mixed horizontally in the sediment.

— Transport of the substance occurs in the x direction in the water layer and in the z direction in the sediment.
It has been **assumed** that transport of the substance in the x direction in the sediment is so small that it can be neglected, so this flux is assumed to be zero.

(Hence, the concentration gradient in the x direction in the sediment originates in the concentration gradient in the x direction of the overlying water.)

The system thus described has been divided into two subsystems to model the substance behaviour. These are the water layer and the sediment, which together form the entire system. This has been done for the following reasons.

1. The substance behaviour in both subsystems can be described by a one-dimensional model. In the water layer this is the x direction, the direction of flow. In the sediment this is the z direction, i.e. downwards.
2. Partly different processes play a role in both subsystems.

The subsystems communicate with each other by exchanging water and substance through the wetted perimeter of the ditch.

2.2 Processes and suspended solids behaviour

In the water subsystem, the substance is subject to the following processes.
- Advection in the x direction.
- Dispersion in the x direction.
- Exchange with the atmosphere, through a diffusive flux across the water-air interface.
- Exchange with the sediment, through a combination of an advective and a diffusive flux across the wetted perimeter.
- Transformation, described as overall transformation without distinction between dissolved substance and substance sorbed to suspended solids or to macrophytes.
  It has been **assumed** that the transformation rate of the substance dissolved in the water phase equals the transformation rate of the substance sorbed to suspended solids or to macrophytes.
- Sorption to suspended solids and to macrophytes.
  It has been **assumed** that sorption to dissolved particles can be neglected in describing the fate of pesticides, so this process has not been included in the model.

Diffusion in the x direction has not been included, as dispersion prevails; the dispersion process exists even in stagnant waters, due to e.g. wind effects and inversion of the water column caused by air temperature changes between day and night.

In the sediment subsystem the substance behaviour has been described by the following processes.
- Advection in the z direction.
- Dispersion in the z direction.
- Diffusion in the z direction.
- Exchange with the water layer, being a combination of an advective and a dispersive flux across the wetted perimeter.
- Transformation, described as overall transformation without distinction between substance dissolved and substance sorbed.
  The **assumption** is that the transformation rate of the substance dissolved in pore water equals that of the substance sorbed to the solid phase of the sediment.
- Sorption to the solid phase of the sediment.

Figure 4 shows a diagram of these processes.

The system description assumes the suspended solids concentration to be constant. The suspended solids flow along with the water. This simplification, i.e. a constant concentration of suspended solids, implies that when additional water is supplied to the ditch (e.g. by drains or trenches, by seepage or by precipitation) the mass of suspended solids will increase as well. These so to say artificially added suspended solids do not contain pesticides at the moment they are added. But immediately afterwards, the total mass of pesticides in the water layer will be redistributed and the pesticides will also sorb to the additional suspended solids. Consequently, the

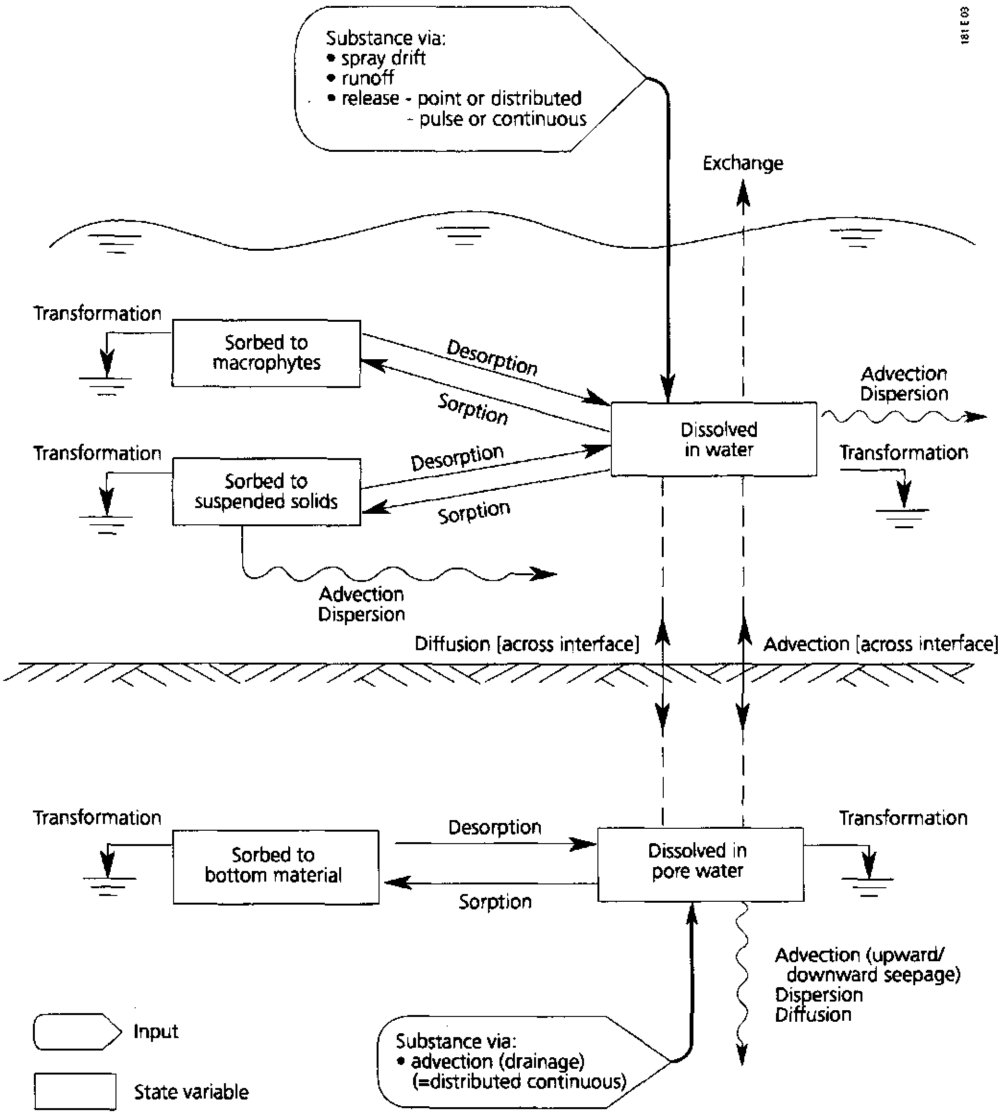


Fig. 4 Diagram of modelled processes

concentration of pesticides dissolved in the water phase will decrease (that is, not only by dilution, but also by sorption to the additional suspended solids).

> The **assumption** is therefore that the concentration of suspended solids in the water layer is constant.

Pesticides sorbed to suspended solids are subject to the same advection and dispersion as pesticides dissolved in the water phase.

> It has also been **assumed** that pesticides sorbed to suspended solids undergo the same advection and dispersion as dissolved pesticides.

No sedimentation or resuspension of suspended solids occurs. Pesticides generally sorb strongly to suspended solids. Therefore, neglecting the sedimentation flux is only acceptable if this is indeed negligibly small. This is the case e.g. over relatively short periods, or over somewhat longer periods in clear, moderately eutrophic ditches. This means that the development of turbid hypertrophic ditches for a succession of years will not be described realistically, because an essential process is disregarded. The same argumentation holds for the resuspension flux. The description is only realistic for those ditches in which the flow velocity is so low and the wind influence so small that these provoke a negligible resuspension of suspended solids. This approach is acceptable for sheltered, very slowly flowing field ditches, but whether it also is acceptable for large, full ditches in open polders or for inclined ditches is less clear.

Hence, the **assumption** has been made that sedimentation and resuspension of suspended solids are negligible.

3 Mass balances for water layer and sediment

3.1 Water layer

The water layer is assumed to be ideally mixed laterally and vertically and the substance shows a concentration gradient only in the direction of flow. This means that a one-dimensional mass balance can be drawn up for the elemental volume $A\Delta x$, in which A represents the wetted surface perpendicular to the direction of flow.

The mass balance for the substance in the water layer now becomes as follows (Fig. 5).

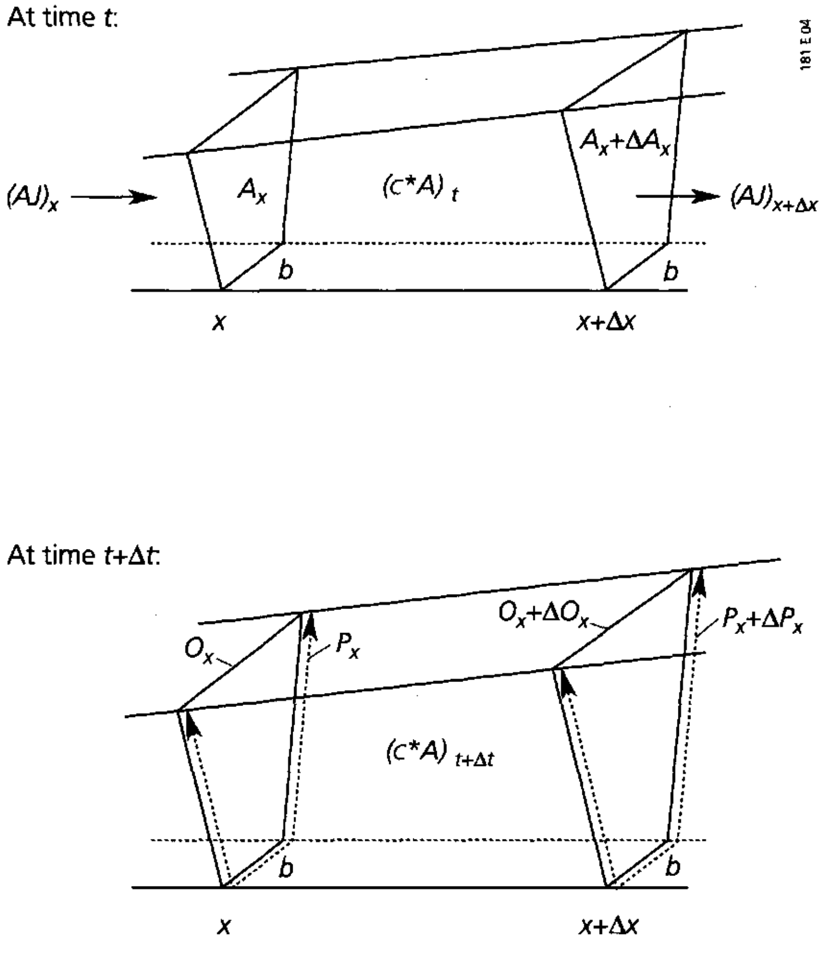


Fig. 5 Mass balance at times t and t + Δt
** (c^*A) is the mass above a unit of length, here Δx*
** A and h at $t+\Delta t$ are changed compared to those at time t*
** P_x en O_x at $t+\Delta t$ are also changed compared to those at time t*

In the above,

A_x = cross sectional area of flow at location x (L^2) (time, space)[3]

b = width of ditch bottom (L)

c^* = mass concentration of substance in water layer (this includes substance sorbed to suspended solids and to macrophytes ($M.L^{-3}$) (time, space)

h = water level above ditch bottom (L) (time, space)

J = areic[4] mass flux of substance in water layer by advection and dispersion ($M.L^{-2}.T^{-1}$) (time, space)

O_x = width of water surface at location x (L) (time, space)

P_x = wetted perimeter at location x (L) (time, space).

The width of the water surface O_x equals:

$$O_x = b + 2hs_1 \tag{3.1}$$

with

s_1 = side slope, horizontal/vertical (1).

The wetted perimeter is described as:

$$P_x = b + 2h\sqrt{s_1^2+1} \tag{3.2}$$

The change in the mass of the substance (i.e. the term c^*A) over length Δx is important for the mass balance of this elemental volume; this change is due to mass transport through the cross sectional area (i.e. the term AJ) at x and $x+\Delta x$.

The mass balance reads:
accumulation = input - output.

Notated in differentials:

$$[(c^*A)_{t+\Delta t} - (c^*A)_t]\Delta x = \Delta t[(AJ)_x - (AJ)_{x+\Delta x}] \Leftrightarrow$$
$$\frac{[(c^*A)_{t+\Delta t} - (c^*A)_t]}{\Delta t} = \frac{[(AJ)_x - (AJ)_{x+\Delta x}]}{\Delta x} \tag{3.3}$$

---

[3] The dimension of the symbol introduced is given between the first pair of brackets. L stands for length, T for time, M for mass, N for mole and Θ for temperature (Schurer and Rigg, 1980). The quantities on which the introduced variable is dependent are given between the second pair of brackets.

[4] Areic means that it is divided by the area concerned.

The limit notation for Δx and Δt reaching zero:

$$\frac{\partial(c^*A)}{\partial t} = -\frac{\partial(AJ)}{\partial x} \tag{3.4}$$

Accounting for sources and sink terms, as well as the exchange of mass across the boundaries of the subsystem, the mass balance reads as follows.

In words:
accumulation = input - output + sources - sinks ± exchange.

In differentials:

$$\begin{aligned}
[(c^*A)_{t+\Delta t} - (c^*A)_t]\Delta x = {} & \Delta t[(AJ)_x - (AJ)_{x+\Delta x}] && \text{flux} \\
& -\Delta t.k[[(c^*A)_x + \tfrac{1}{2}\Delta(c^*A)_x]\Delta x] && \text{transformation} \\
& +\Delta t.J_{wa}[(O_x + \tfrac{1}{2}\Delta O_x)\Delta x] && \text{removal to atmosphere} \\
& -\Delta t.J_{wb}[(P_x + \tfrac{1}{2}\Delta P_x)\Delta x] && \text{exchange with sediment}
\end{aligned}$$

$$\begin{aligned}
\Leftrightarrow \quad \frac{[(c^*A)_{t+\Delta t} - (c^*A)_t]}{\Delta t} = {} & \frac{[(AJ)_x - (AJ)_{x+\Delta x}]}{\Delta x} \\
& -k[(c^*A)_x + \tfrac{1}{2}\Delta(c^*A)_x] \\
& +J_{wa}(O_x + \tfrac{1}{2}\Delta O_x) \\
& -J_{wb}(P_x + \tfrac{1}{2}\Delta P_x)
\end{aligned} \tag{3.5}$$

In the above,
J_{wa} = areic mass flux of substance across the water-air interface; the flux is negative in the upward direction ($M.L^{-2}.T^{-1}$) (time, space)
J_{wb} = areic mass flux of substance across the water-sediment interface; the flux is positive in the downward direction ($M.L^{-2}.T^{-1}$) (time, space)
k = transformation rate coefficient[5] for substance in the water column (T^{-1}) (-).

The limit notation (Δx, Δt, ΔO_x, ΔP_x, and $\Delta(c^*A)$ reach zero):

$$\frac{\partial(c^*A)}{\partial t} = -\frac{\partial(AJ)}{\partial x} - k(c^*A) + J_{wa}.O_x - J_{wb}.P_x \tag{3.6}$$

This is the conservation equation for the water layer.

[5] In this report lowercase k is used to indicate rate coefficients and uppercase K to indicate equilibrium coefficients.

3.2 Sediment

Diffusion across the water-sediment interface and sorption to sediment are important processes in describing the behaviour of pesticides in surface waters. In small watercourses, the walls account for an important part of the total exchange area between water and sediment. Therefore, it is more realistic to use the wetted perimeter, instead of the bottom width, to calculate the exchange area (De Heer, 1979).

Perpendicular to the wetted perimeter, the substance is transported from the water layer to the sediment. Per unit of length in the direction of flow, this transport takes place through an area $P.1$ (L^2).

$$P_0 = b + 2h\sqrt{s_1^2+1} \tag{3.7}$$

with

P_0 = wetted perimeter (L)

h = water level above bottom of ditch (L)

At a distance d from the water-sediment interface transport in the sediment takes place through the area $P_d.1$. (Fig. 6.)

$$P_d = b + 2d.\tan(\frac{1}{2}\beta) + 2(h+d)\sqrt{s_1^2+1} \tag{3.8}$$

with

P_d = length of wetted perimeter at distance d from the water-sediment interface (L)

β = arctan ($1/s_1$) (1)

d = distance of water-sediment interface to area concerned (L).

Hence, the area through which transport takes place increases with increasing distance from the water-sediment interface.

Transport perpendicular to the wetted perimeter occurs in two dimensions, the z direction and the y direction. This two-dimensional transport has been simplified to a one-dimensional transport in the z direction. This implies that transport around corners is neglected; the ditch walls are, as it were, straightened and the flow pattern has been simplified to one-dimensional transport in a widening sediment column. (Fig. 7).

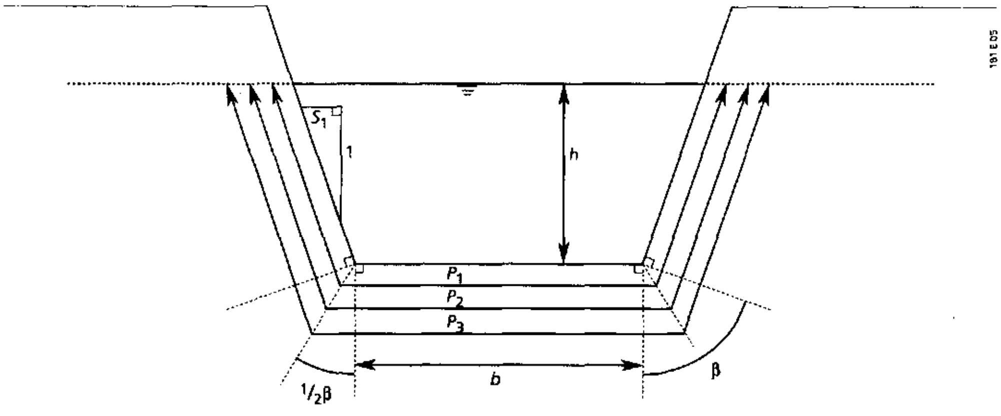


Fig. 6 Cross section of a ditch with the shape of stacked trapezia

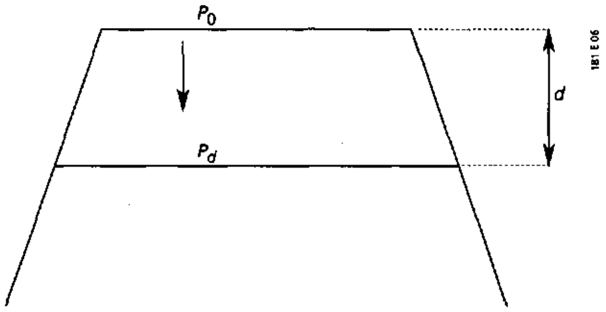


Fig. 7 The sediment subsystem simplified to one dimension

> Hence, the **assumption** is that the two-dimensional transport can be calculated by means of a one-dimensional description of mass transport in a sediment column widening with depth.

This one-dimensional description has been applied to the sediment subsystem situated below the exchange area between water layer and sediment; this area has a constant size.

> The **assumption** has been made that the size of the exchange area is constant, and does not depend on the varying water level in the ditch.

The size of the exchange area per unit of length in the x direction equals the wetted perimeter of the ditch corresponding to a particular water level h_w (Fig. 8). This can be the minimal water level occurring in the ditch. The mass transport thus takes place in a trapezium-shaped sediment column underneath the area of constant size $P_{z=0}$, per unit of length.

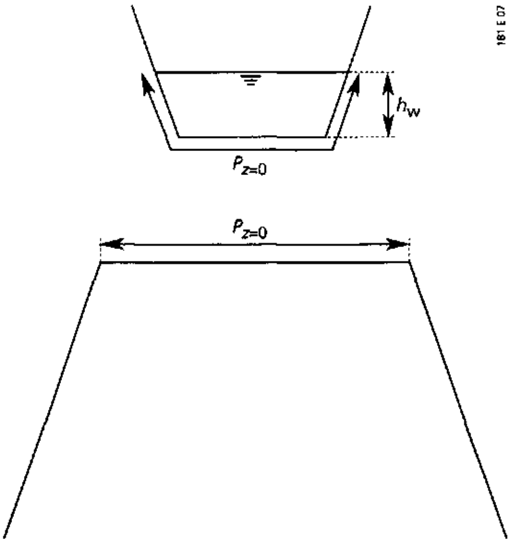


Fig. 8 The sediment subsystem situated below an area of constant size, through which exchange of pesticide between water layer and sediment occurs

$$P_{z=0} = b + 2h_w\sqrt{s_1^2 + 1} \tag{3.9}$$

The mass balance reads:
accumulation = input - output.

In Figure 9:

c_b^* = mass concentration of substance in sediment (M.L^{-3}) (time, space)
J_{lb} = areic mass flux of substance in the liquid phase of the sediment by advection, dispersion and diffusion (M.L^{-2}.T^{-1}) (time, space)
$P_{z=0}$ = length of wetted perimeter at depth z = 0 (L) (space).

In differentials, the mass balance reads:

$$(P_z + \frac{1}{2}\Delta P_z).1.\Delta z.\Delta c_b^* = \Delta t.1.(P_z.J_{lb,z} - (P_z + \Delta P_z).J_{lb,z+\Delta z}$$

$$\Leftrightarrow \qquad \frac{\Delta c_b^*}{\Delta t} = \frac{P_z.J_{lb,z} - P_z.J_{lb,z+\Delta z}}{(P_z + \frac{1}{2}\Delta P_z)\Delta z} - \frac{\Delta P_z.J_{lb,z+\Delta z}}{\Delta z.(P_z + \frac{1}{2}\Delta P_z)} \tag{3.10}$$

The limit notation (Δz, Δt and ΔP_z approach zero):

$$\frac{\partial c_b^*}{\partial t} = - \frac{\partial J_{lb}}{\partial z} - \frac{\partial P}{\partial z} \cdot \frac{J_{lb}}{P} \qquad (3.11)$$

$$(\Leftrightarrow \quad P\frac{\partial c_b^*}{\partial t} = - \frac{\partial (PJ_{lb})}{\partial z})$$

This is the conservation equation for the sediment subsystem below the exchange area of constant size.

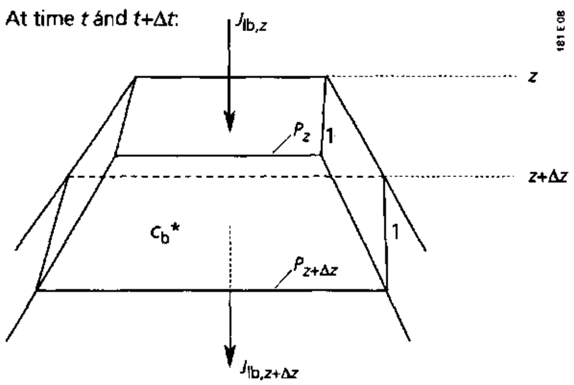


Fig. 9 Mass balance for the sediment subsystem
** area per unit of length in the x direction is considered*
** wetted areas $P_z.1$ and $P_z+\Delta P_z$ are constant in time*

Analogously to the conservation equation for the water layer (Eq. 3.6), a sink term -$k_b c_b^* P$ has been added to the conservation equation for the sediment (Eq. 3.11). There is no source term for the pesticide. The assumption of negligible fluxes in the x direction in the sediment implies that there is no exchange of mass across the boundaries of the subsystem.

Note that the exchange term between water layer and sediment in the conservation equation for the water layer (Eq. (3.6)) becomes a mathematical boundary condition here. Hence, this term is not explicitly visible in the conservation equation for the sediment.

Taking the above-mentioned sink term into account, the complete conservation equation for the sediment now reads:

$$P \frac{\partial c_b^*}{\partial t} = - \frac{\partial (PJ_{lb})}{\partial z} - k_b c_b^* P \qquad (3.12)$$

with

k_b = transformation rate coefficient for substance in the sediment (T^{-1}).

This conservation equation is coupled to the conservation equation for the water layer at the wetted perimeter P (at z=0) by the boundary condition for the flux J_{1b} at z=0. In principle, the conservation equations should be solved simultaneously.

4 Basic processes in water layer and sediment

4.1 Water layer

The conservation equation for the water layer reads:

$$\frac{\partial(c^*A)}{\partial t} = -\frac{\partial(AJ)}{\partial x} - k(c^*A) + J_{wa}.O_x - J_{wb}.P_x \qquad (3.6)$$

The substance concentration in the water layer, c^*, is described as follows.

$$c^* = c + \frac{DW.P_{z=0}}{A}.X_{mp} + ss.X_{ss} \qquad (4.1)$$

with

c^*	=	mass concentration of substance in the water layer ($M.L^{-3}$)
c	=	mass concentration of substance in the water phase ($M.L^{-3}$)
DW	=	dry weight of macrophytes per area of sediment ($M.L^{-2}$)
X_{mp}	=	content of substance sorbed to macrophytes, i.e. the ratio of the mass of substance sorbed divided by the mass of dry macrophytes ($M.M^{-1}$)
ss	=	mass concentration of suspended solids in the water layer, i.e. the ratio of the mass of dry suspended solids divided by the volume of water ($M.L^{-3}$)
X_{ss}	=	content of substance sorbed to suspended solids, i.e. the ratio of the mass of substance sorbed divided by the mass of dry suspended solids ($M.M^{-1}$)

By analogy to sorption to soil and sorption to the solid phase of sediment, the content of substance sorbed to the suspended solids equals:

$$X_{ss} = K_{F,ss}.c_{e,ss}.\left(\frac{c}{c_{e,ss}}\right)^{n_{ss}} \qquad (4.2)$$

with

$K_{F,ss}$	=	Freundlich coefficient for sorption to suspended solids ($L^3.M^{-1}$)
$c_{e,ss}$	=	concentration c at which $K_{F,ss}$ has been estimated ($M.L^{-3}$)
n_{ss}	=	Freundlich exponent for sorption to suspended solids (1)

The Freundlich coefficient for sorption to suspended solids, $K_{F,ss}$, is related to the $K_{om,ss}$ in the following way:

$$K_{F,ss} = m_{om,ss}.K_{om,ss} \qquad (4.3)$$

with

$m_{om,ss}$ = mass fraction of organic matter in the suspended solids ($M.M^{-1}$)
Hence, it has been **assumed** that sorption to suspended solids and sorption to sediment are analogous processes to sorption to soil and that both can be described with the aid of the Freundlich equation.

$K_{om,ss}$ = slope of sorption isotherm, based on the organic matter content ($L^3.M^{-1}$)

The content sorbed to macrophytes can be described as follows:

$$X_{mp} = K_{mp}.c \tag{4.4}$$

with

K_{mp} = distribution coefficient for substance between macrophytes and water, i.e. the slope of the sorption isotherm based on the mass of dry macrophytes ($L^3.M^{-1}$).

The **assumption** is that sorption of the substance to suspended solids, to macrophytes and to the solid phase of the sediment (a.o. Wauchope and Myers, 1985) is a relatively rapid process compared to other processes, so one can assume an instantaneous equilibrium.

The wetted area A of the conservation equation equals:

$$A = bh + h^2 s_1 \tag{4.5}$$

The flux J of the conservation equation describes the transport of substance, both dissolved in water and sorbed to the suspended solids. It consists of an advective and a dispersive component (which are assumed identical for the dissolved and sorbed substance).

$$J = u(c + ss.X_{ss}) - E_x \frac{\partial(c + ss.X_{ss})}{\partial x} \tag{4.6}$$

with

J = areic mass flux of substance (both dissolved in water and sorbed to suspended solids) in the water layer ($M.L^{-2}.T^{-1}$)

u = flow velocity of the water ($L.T^{-1}$)

E_x = dispersion coefficient in the direction of flow ($L^2.T^{-1}$).

The term $\partial(AJ)/\partial x$ equals

$$\frac{\partial(A[u(c + ss.X_{ss}) - E_x \frac{\partial(c + ss.X_{ss})}{\partial x}])}{\partial x} = \frac{\partial(Q(c + ss.X_{ss}) - AE_x \frac{\partial(c + ss.X_{ss})}{\partial x})}{\partial x} \tag{4.7}$$

in which

$Q = A.u$

with

Q = rate of discharge in the water layer ($L^3.T^{-1}$)

The transformation rate coefficient k is a measure of the (entire) transformation of the substance. The substance may be dissolved in water or it may be sorbed to suspended solids or to macrophytes. The three most important transformation processes in the water layer are photolysis, hydrolysis and biodegradation (Thomann and Mueller, 1987). Photolysis depends mainly on the light intensity in the water column, hydrolysis depends especially on the pH and biodegradation is mainly determined by the extent and type of the bacterial community and the temperature.

The exchange flux of the substance between water body and atmosphere, J_{wa}, is described by the film model of two laminar layers at an interface (Liss and Slater, 1974, Mackay and Leinonen, 1975, review: Mackay, 1981).

$$J_{wa} = -k_l(c - c_I) = +k_g(c_a - c_{a,I}) \tag{4.8}$$

with

J_{wa} = areic mass flux at the air-water interface ($M.L^{-2}.T^{-1}$)

k_l = exchange coefficient of substance in the liquid phase ($L.T^{-1}$)

k_g = exchange coefficient of substance in the gas phase ($L.T^{-1}$)

c_I = equilibrium mass concentration of substance at the water-gas interface in the water phase ($M.L^{-3}$)

c_a = mass concentration of substance in the air ($M.L^{-3}$)

$c_{a,I}$ = equilibrium mass concentration of substance at the water-gas interface in the gas phase ($M.L^{-3}$).

If the exchanging substance obeys Henry's law, then the equilibrium situation at the interface may be described by:

$$c_{a,I} = K_H . c_I \tag{4.9}$$

with

K_H = dimensionless Henry coefficient (1).

Henry's coefficient is estimated from the quotient of mass concentration of saturated vapour of the substance (via vapour pressure) and the solubility of the substance in water.

$$K_H = \frac{P.M_m}{R.T} \cdot \frac{1}{c_{sol}} \tag{4.10}$$

with

P = saturated vapour pressure of substance ($L^{-1}.M.T^{-2}$)

M_m = molecular mass ($M.N^{-1}$)

R = universal gas constant ($L^2.M.T^{-2}.N^{-1}.\theta^{-1}$)

T = temperature at which the saturated vapour pressure, the solubility and the exchange coefficients in the liquid and gas phases are defined (θ)
c_{sol} = solubility of substance in water ($M.L^{-3}$).

Eliminating c_I and $c_{a,I}$ from Eq. (4.8) with the aid of Eq. (4.9) flux J_{wa} becomes as follows:

$$J_{wa} = -k_{t,l}(c - \frac{c_a}{K_H}) \tag{4.11}$$

in which

$$\frac{1}{k_{t,l}} = \frac{1}{k_l} + \frac{1}{K_H . k_g} \tag{4.12}$$

with
$k_{t,l}$ = overall transfer coefficient for the air-water interface, based on the liquid phase ($L.T^{-1}$).

The exchange flux between water layer and sediment consists of an advective and a diffusive component.

$$J_{wb} = J_{wb,adv} + J_{wb,dif} \tag{4.13}$$

with
J_{wb} = areic mass flux at the water-sediment interface ($M.L^{-2}.T^{-1}$)
$J_{wb,adv}$ = areic mass flux by advection at the water-sediment interface ($M.L^{-2}.T^{-1}$)
$J_{wb,dif}$ = areic mass flux by diffusion at the water-sediment interface ($M.L^{-2}.T^{-1}$).

The description of this flux given below assumes that diffusion across the water-sediment interface is rapid compared to diffusion in sediment. This implies that the interface resistance is negligible and that the mass concentration of the substance in the water layer equals that in the outer sediment pores.

The **assumption** is thus that the resistance to transport of the substance across the water-sediment interface can be neglected.

The advective component of the exchange flux consists either of a supply of water with the substance towards the ditch because of drainage from neighbouring field lots, or of infiltrating water from the ditch into the field lots. Because of the assumption of no transport resistance across the water-sediment interface ($c = c_{lb,z=0}$), this yields:

$$J_{wb,adv} = \frac{\ell}{P_{z=0}} . q . c \tag{4.14}$$

with
ℓ = length of drained or infiltrated lot, oriented perpendicular to the ditch and extending on one or two sides of the ditch (L)

q = areic volume flux, i.e. volume of drained or supplied water divided by lot area and time ($L.T^{-1}$). The flux is positive for infiltration and negative for upward flow (drainage from the field lot).

The diffusive component of the exchange flux is described as the areic mass flux by diffusion in the sediment at the location z=0.

$$J_{wb,dif} = -(\varepsilon . D_{lb} . \frac{\partial c_{lb}}{\partial z})_{z=0} \quad (4.15)$$

with

D_{lb} = diffusion coefficient of substance in the liquid phase of the sediment ($L^2.T^{-1}$)

ε = volume fraction of pore water, i.e. volume of liquid divided by volume of bottom material (1).

c_{lb} = mass concentration of substance in the liquid phase of the sediment ($M.L^{-3}$).

The diffusion coefficient of the substance in the liquid phase of the sediment, D_{lb}, is calculated as:

$$D_{lb} = \lambda . D_w \quad (4.16)$$

with

λ = tortuosity factor, i.e. ratio of surface area of bottom material to liquid phase (1)

D_w = diffusion coefficient of substance in water ($L^2.T^{-1}$).

The conservation equation for the water layer (Eq. 3.6), is now entirely defined. All the terms of the equation have been unambiguously described. In fact, this results in one equation with only one unknown state variable (e.g. c). After solution, the other state variables (e.g. X_{ss}, X_{mp}) can be calculated. These are also determined by system parameters (like DW, $m_{om,ss}$, k, ε, D_w) and by inputs (like q and c_a). The state variables Q and h form an exception. They are not defined by the conservation equation, but partly define this themselves. They have been calculated in a water flow model and they need to be known at every chosen location and time if one is to calculate the water quality variables.

4.2 Sediment

The conservation equation for the sediment reads:

$$P \frac{\partial c_b^*}{\partial t} = - \frac{\partial (P J_{lb})}{\partial z} - k_b c_b^* P \quad (3.12)$$

The concentration of substance in the sediment, c_b^*, is defined as:

$$c_b^* = \varepsilon c_{lb} + \rho_b X_b \tag{4.17}$$

with
c_b^* = mass concentration of substance in sediment ($M.L^{-3}$)
ρ_b = bulk density of dry bottom material, i.e. volumic mass of dry bottom material ($M.L^{-3}$)
X_b = content of substance sorbed, i.e. the ratio of the mass of substance sorbed divided by the mass of dry bottom material ($M.M^{-1}$)

The content of substance sorbed to sediment, X_b, is (Jafvert, 1990):

$$X_b = K_{F,wb} \cdot c_{e,wb} \cdot \left(\frac{c_{lb}}{c_{e,wb}} \right)^{n_{wb}} \tag{4.18}$$

with
$K_{F,wb}$ = Freundlich coefficient for sorption to bottom material ($L^3.M^{-1}$)
$c_{e,wb}$ = Concentration c at which $K_{F,wb}$ has been estimated ($M.L^{-3}$)
n_{wb} = Freundlich exponent for sorption to bottom material (1)

$$K_{F,\,wb} = m_{om,\,wb}\, K_{om,\,wb} \tag{4.19}$$

with
$m_{om,wb}$ = mass fraction of organic matter of the suspended solids ($M.M^{-1}$)
$K_{om,wb}$ = slope of sorption isotherm, based on the organic matter content ($L^3.M^{-1}$).

The transformation rate coefficient for the substance in the sediment, k_b, is a measure of the entire transformation of the substance in pore water as well as in sorbed form. The main transformation processes in the sediment are hydrolysis and biodegradation.

The perimeter below the water-sediment interface, P, depends on the distance to this interface and the chosen water level h_w in the ditch. For the simplified case of one-dimensional transport in the z direction, P is defined as follows (Fig. 10).

$$P = b + 2z.\tan(\frac{1}{2}\beta) + 2(h_w + z)\sqrt{s_1^2 + 1} \tag{4.20}$$

with
β = $\arctan(1/s_1)$ (1)
z = depth below the water-sediment interface (L)

In the pore water, the substance is transported by advection, dispersion and diffusion. This flux, J_{lb}, is defined as follows[6].

$$J_{lb} = \frac{\ell}{P}.q.c_{lb} - \varepsilon(E_{lb} + D_{lb}) \frac{\partial c_{lb}}{\partial z} \tag{4.21}$$

with
E_{lb} = dispersion coefficient in pore water ($L^2.T^{-1}$).

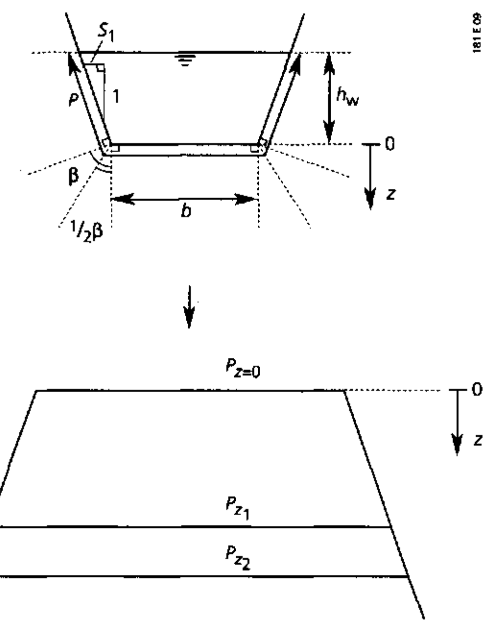


Fig. 10 Calculation of wetted perimeter in the sediment subsystem

The dispersion coefficient is defined as:

$$E_{lb} = L_{dis}.|w| \tag{4.22}$$

with
L_{dis} = dispersion length (L)
w = average flow velocity of pore water (i.e. $\ell q/P\varepsilon$) ($L.T^{-1}$).

A special problem may occur in the sediment. Very sharp concentration gradients may exist here. The magnitude (and direction) of the diffusive and dispersive fluxes are calculated with the aid of concentration gradients. (See e.g. Eq. (4.21).) Dispersion of the substance originates in the unequal flow velocities of pore water (i.e. in the velocity distribution) and is thus caused by advection. It then logically follows that the combined material flux resulting from advection and its ensuing

[6] The first term represents the advection flux, which equals: flow velocity of pore water * mass concentration in pore water * volume fraction of pore water = $(\ell q)/(P\varepsilon)$ * c_{lb} * ε = (ℓ/P) * q * c_{lb}

dispersion, should have the same direction as the advection. (Otherwise the effect would cancel out its cause) (Bolt, 1979, pp. 301 and 346).

Figure 11 shows the four possible situations, occurring in the sediment. In Situations 1 and 3, the advective and dispersive fluxes have the same direction, so the above-mentioned problem does not arise. In Situation 2 as well as in Situation 4, the advective and dispersive fluxes have opposite directions, so it is possible that the dispersive flux cancels out or even exceeds the advective flux. This is prevented as long as the following condition is fulfilled.

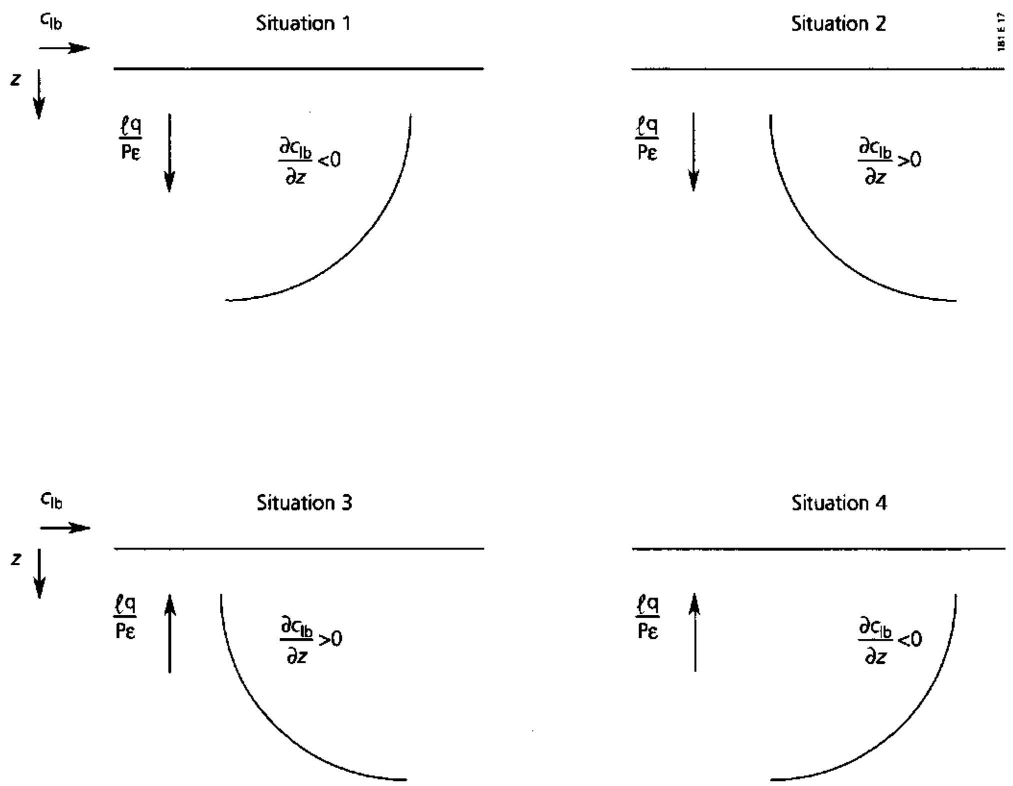


Fig. 11 Four situations in the sediment, depicting the possible combinations of the directions of the advection flow ($\ell q/P\varepsilon$)and the concentration gradient $\partial c_{lb}/\partial z$

For Situation 2:
$q > 0$ and
$\partial(c_{lb})/\partial z > 0$.

According to Eq. (4.22), with $q > 0$:

$$E_{lb} = L_{dis} \cdot \frac{\ell q}{P\varepsilon} \tag{4.22a}$$

The dispersive flux should not exceed the advective flux:

$$
\begin{aligned}
|\frac{\ell}{P} q c_{lb}| > |\varepsilon E_{lb} \frac{\partial c_{lb}}{\partial z}| \quad &\Leftrightarrow \\
q c_{lb} > L_{dis}\, q \frac{\partial c_{lb}}{\partial z} \quad &\Leftrightarrow \\
q(c_{lb} - L_{dis} \frac{\partial c_{lb}}{\partial z}) > 0 \quad &\Leftrightarrow \quad (q \text{ is positive}) \\
c_{lb} - L_{dis} \frac{\partial c_{lb}}{\partial z} > 0 &
\end{aligned}
\tag{4.23}
$$

For Situation 4:
$q < 0$ and
$\partial(c_{lb})/\partial z < 0$.

Combining Eq. (4.22) with $q < 0$ results in:

$$E_{lb} = L_{dis} . \frac{-\ell q}{P\varepsilon} \tag{4.22b}$$

The condition that the dispersive flux should not exceed the advective flux leads to:

$$
\begin{aligned}
|\frac{\ell}{P} q c_{lb}| > |\varepsilon E_{lb} \frac{\partial c_{lb}}{\partial z}| \quad &\Leftrightarrow \\
- q c_{lb} > L_{dis} \cdot -q \cdot \frac{-\partial c_{lb}}{\partial z} \quad &\Leftrightarrow \\
- q(c_{lb} + L_{dis} \frac{\partial c_{lb}}{\partial z}) > 0 \quad &\Leftrightarrow \quad (-q \text{ is positive}) \\
c_{lb} + L_{dis} \frac{\partial c_{lb}}{\partial z} > 0 &
\end{aligned}
\tag{4.24}
$$

This means that the dispersive flux is smaller than the advective flux if the following conditions are fulfilled:

$$\text{for } q > 0 \text{ ánd } \frac{\partial c_{lb}}{\partial z} > 0: \qquad c_{lb} - L_{dis} \frac{\partial c_{lb}}{\partial z} > 0 \tag{4.23}$$

and

$$\text{for } q < 0 \text{ ánd } \frac{\partial c_{lb}}{\partial z} < 0: \qquad c_{lb} + L_{dis} \frac{\partial c_{lb}}{\partial z} > 0 \tag{4.24}$$

In the case the dispersion flux would exceed the advective flux, it has been assumed that the dispersive flux will only cancel out the advective flux. Hence, only the diffusive flux is left in these cases. (See also De Heer, 1979, p.122.)

The conservation equation for the sediment is now entirely specified. All the terms of this equation have been unambiguously defined in this section. Just as for the water layer, the result is again one equation with one unknown variable (e.g. c_{lb}). This is coupled to the equation for the water layer via the boundary condition $c_{lb,z=0} = c$. In principle, the two conservation equations, for the water layer and for the sediment, need to be solved simultaneously.

5 Boundary conditions and initial condition, other input

5.1 Water layer

The conservation equation for the water layer reads:

$$\frac{\partial(c^*A)}{\partial t} = -\frac{\partial(AJ)}{\partial x} - k(c^*A) + J_{wa}.O_x - J_{wb}.P_x \qquad (3.6)$$

The boundary condition at the upper end of the subsystem states that water without pesticide flows into the ditch.

For $t \geq 0$ and $x = 0$:

$$J = 0 \qquad (5.1)$$

The boundary condition at the end of the ditch is:

For $t \geq 0$ and x = end value of ditch (e.g. 100 m):

$$J = u(c + ss.X_{ss}) - E_x \, . \, \frac{\partial(c + ss.X_{ss})}{\partial x} \qquad (5.2)$$

The initial condition is that there is no pesticide in the water layer.

For $t = 0$ and $x > 0$:

$$c^* = 0 \qquad (5.3)$$

Applications of pesticides into the ditch are not, strictly speaking, included in the boundary conditions (at $x = 0$ or x = end value) or the initial condition (at $t = 0$). These inputs to the subsystem are included as options in the conservation equation. A distinction is made between four situations.

In the first and second situations different (point-type or distributed) pulse inputs occur. In an infinitesimally small period of time, a mass of substance is released into the ditch water. The mass of substance applied per unit of volume during this period, $p(t,x)$, is described below. In both cases, a term $p(t,x)$ is added to the conservation equation.

In the first situation a distributed pulse input of the substance occurs, e.g. spray drift, a momentary runoff or a brief release from a number of nearby drains or trenches. It is assumed that the supplied lineic mass of substance M_L (dimension $M.L^{-1}$) is instantaneously and ideally mixed over the water column at the time of release. Such an pulse input may occur repeatedly.

$$p_d(t,x) = \sum_{s=1}^{m} \frac{M_L}{A} . \delta(t-t_s) \quad (5.4)$$

in which
$p_d(t,x)$ = distributed pulse input ($M.L^{-3}.T^{-1}$)
s = number of pulse input (total m inputs) (1)
and δ is the Dirac delta function (dimension T^{-1}) defined by (Fig. 12):

$\delta(t-t_s) =$ 0 for $t \neq t_s$ and

$$\int_{-\infty}^{\infty} \delta(t-t_s) dt = 1 \quad (5.5)$$

with
t_s = time of pulse input (T).

In the second situation there is a point-type pulse input, e.g. spillage of the substance, rinsing application equipment, or a brief release from individual drains or trenches. The mass of substance, released in an infinitesimally short time, is M (dimension M) and this is assumed to be in an infinitesimally thin, vertical slice of the ditch. An pulse input, occurring repeatedly, can be described by:

$$p_p(t,x) = \sum_{s_0=1}^{m_0} \frac{M.\delta(x-x_0).\delta(t-t_{s_0})}{A} \quad (5.6)$$

in which
$p_p(t,x)$ = point-type pulse input ($M.L^{-3}.T^{-1}$)
s_0 = number of pulse input at location x_0 (total m_0 inputs)
and δ is the Dirac delta function (dimension L^{-1}) defined by (Fig. 13):

$\delta(x-x_0) = 0$ for $x \neq x_0$ and

$$\int_{-\infty}^{\infty} \delta(x-x_0) dx = 1 \quad (5.7)$$

with
x_0 = location of pulse input (L).

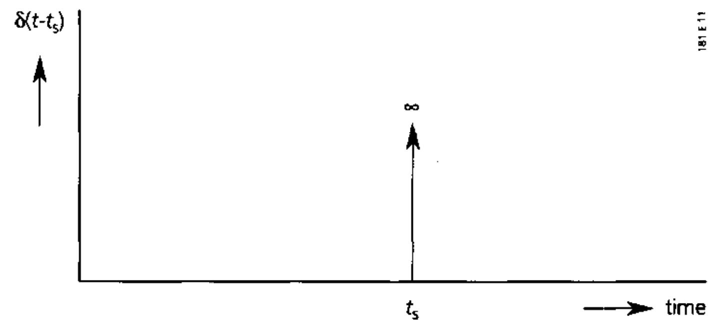


Fig. 12 Dirac delta function for pulse input

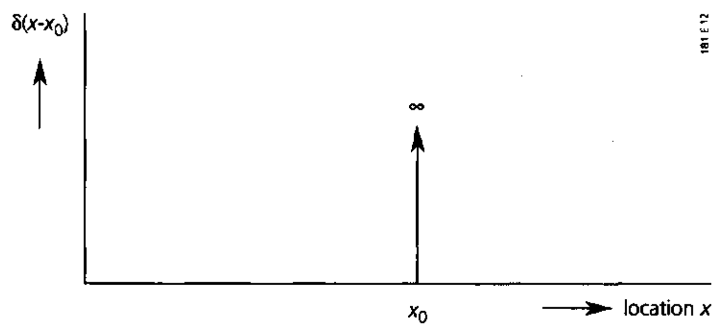


Fig. 13 Dirac delta function for point-type input

If such a pulse input occurs at a second location, the term $p_p(t,x)$ becomes as follows. (Etc.: for every subsequent location another term is added.)

$$p_p(t,x) = \sum_{s_0=1}^{m_0} \frac{M.\delta(x-x_0).\delta(t-t_{s_0})}{A} + \sum_{s_{00}=1}^{m_{00}} \frac{M.\delta(x-x_{00}).\delta(t-t_{s_{00}})}{A} \qquad (5.8)$$

in which

s_{00} = number of pulse input at location x_{00} (total m_{00} inputs).

In the third and fourth situations, there are one or more point-type or distributed sources, which start emitting from a certain moment and continue doing so for a certain period. For both situations, the mass of substance applied per unit of volume per unit of time, $b(t,x)$, is described below. A term $b(t,x)$ is added to the conservation equation.

In the third situation, a continuous point release takes place, e.g. a discharging tributary or an individual trench adding water with a constant concentration of the substance. The mass of substance released per unit of time is M_T (dimension $M.T^{-1}$, e.g. calculated as $Q_{tributary}c_{tributary}$) and this is ideally mixed over the cross sectional area A.

$$b_p(t,x) = \sum_{r=1}^{n} \frac{M_T}{A}.\zeta(t).\delta(x - x_r) \tag{5.9}$$

in which

$b_p(t,x)$ = point source with continuous input ($M.L^{-3}.T^{-1}$)

r = number of point-type input (total n inputs)

and $\zeta(t)$ is the release time function (dimension (1)) defined by:

$\zeta(t)$ = 0 for $t < t_{rs}$ or $t > t_{re}$

= 1 for $t_{rs} \leq t \leq t_{re}$

with

t_{rs} = time at which the continuous release at location x_r starts (T)

t_{re} = time at which the continuous release at location x_e ends (T).

In the fourth situation, there is a continuous, distributed release, e.g. continuing release from many nearby drains or trenches. The lineic mass of substance released per unit of time is M_{LT} (dimension $M.L^{-1}.T^{-1}$, e.g. calculated as discharge per unit of length in the flow direction x times the concentration in trench/drain water: $(q\ell).c_{dg}$, with c_{dg} = mass concentration of substance in drain/trench water).

$$b_d(t,x) = \sum_{r=1}^{n} \frac{M_{LT}}{A}.\zeta(t) \tag{5.10}$$

with

$b_d(t,x)$ = distributed source with continuous input ($M.L^{-3}.T^{-1}$).

Depending on the type of application of the substance to the subsystem, one or more terms are added to the conservation equation, Eq. (3.6). These optional terms have been defined by Eqs. (5.4), (5.6), (5.8), (5.9) and (5.10). Hence, the completed conservation equation for the water layer reads:

$$\frac{\partial(c^*A)}{\partial t} = -\frac{\partial(AJ)}{\partial x} - k(c^*A) + J_{wa}.O_x - J_{wb}.P_x$$

$$\begin{aligned} &\text{option: } +p_d.A \\ &\text{option: } +p_p.A \\ &\text{option: } +b_p.A \\ &\text{option: } +b_d.A \end{aligned} \tag{3.6a}$$

5.2 Sediment

The conservation equation for the sediment reads:

$$P \frac{\partial c_b^*}{\partial t} = - \frac{\partial (P J_{lb})}{\partial z} - k_b c_b^* P \qquad (3.12)$$

The mass flux at the water-sediment interface is composed of an advective and a diffusive component (there is no dispersive component across the interface):

$$J_{lb} = \frac{\ell}{P_0} q \cdot c_{lb} - \varepsilon D_{lb} \cdot \frac{\partial c_{lb}}{\partial z} \qquad (5.11)$$

The boundary condition at the sediment surface is:

For $t \geq 0$ and $z = 0$:

$$c_{lb} = c \qquad (5.12)$$

The boundary condition at the lower end of the sediment subsystem consists of the mass fluxes by advection, dispersion and diffusion.

For $t \geq 0$ and z = end value of sediment (e.g. 0.10 m):

$$J_{lb} = \frac{\ell}{P} q \cdot c_{lb} - \varepsilon (E_{lb} + D_{lb}) \frac{\partial c_{lb}}{\partial z} \qquad (5.13)$$

There are two options for the initial condition. One option assumes the sediment to be free of pesticide.

For $t = 0$ and $z > 0$:

$$c_b^* = 0 \qquad (5.14)$$

In the other option there is sediment with a certain mass of pesticide, specified as a function of depth.

For $t = 0$ and $z > 0$:

$$c_b^* = \mathrm{f}(z) \qquad (5.15)$$

in which

f(z) = mass concentration of substance in sediment as a function of depth at starting time $t = 0$ ($M.L^{-3}$).

6 Numerical solution of the mass conservation equations

6.1 Introduction

The conservation equation for the water subsystem Eq. (3.6) and that for the sediment subsystem Eq. (3.12) have been solved numerically with the aid of the finite-difference method. For this purpose a rectangular grid of points in the water subsystem was defined in the (x,t) plane, numbered $i = 1, 2, 3, ...$ along the x axis and $j = 1, 2, 3, ...$ along the t axis. The x axis was assumed to be positive in the direction of dominant flow. Δx_i was defined as the length of a segment around point i, while Δt was defined as the time step (Fig. 14).

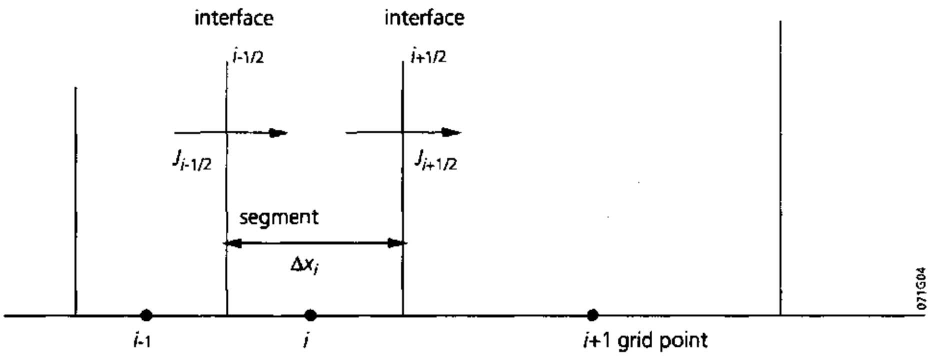


Fig. 14 Outline defining the discretisation of the x axis (water layer)

A rectangular grid of points in the sediment subsystem was defined in the (z,t) plane, numbered $k = 1, 2, 3, ...$ along the z axis and $j = 1, 2, 3, ...$ along the t axis. The z axis was assumed to be positive in the downward direction. Δz_k is the thickness of a segment around point k and Δt is again the time step (Fig. 15).[7]
Concentrations in the grid were defined at the grid points, while fluxes were defined at the interfaces, as well as the corresponding areas through which the fluxes occur (e.g. Bella and Dobbins, 1968). In the water subsystem, water flow was described with the aid of the water depth at a grid point and the flow velocity or rate of discharge through an interface.

[7] As both subsystems, water layer and sediment, are coupled to each other, it was assumed that the same time step Δt was employed in both water layer and sediment. It is, however, possible that one of the subsystems permits a much larger time step than the other and that one could make use of this to shorten the calculation time. In that case a Δt_w and a Δt_b may be defined, in which one is a multiple of the other. TOXSWA 1.0 allows only identical time steps to be applied.

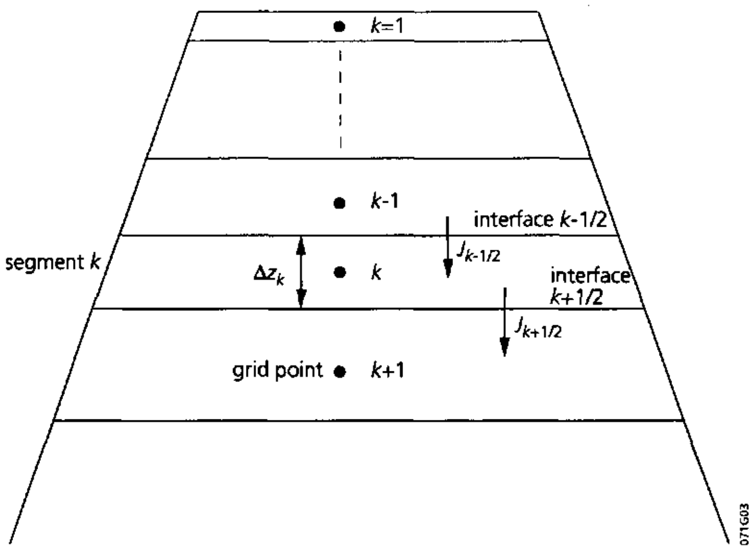


Fig. 15 Outline defining the discretisation of the z axis (sediment)

The terms upper and lower boundary have been defined here in relation to the variable of space: the upper boundary is located at $x = 0$ or $z = 0$ and the lower boundary at the end values of x and z. Table 1 shows an overview of the variables and the locations at which these were defined, for both the water and sediment subsystems.

The finite-difference method offers several possibilities. Spatial derivatives may be approximated with the aid of forward, central or backward differences. Temporal derivatives of the differential equation considered may be approximated in an explicit or an implicit manner. Selecting the calculation scheme depends on two factors, calculation time and acceptable numerical dispersion. The calculation time is mainly determined by the size of the space steps, Δx or Δz, or of the time steps, Δt. The step sizes are prescribed by the conditions which have been imposed to obtain stability, or in fact positivity of the solution. An unstable solution of the differential equations oscillates and diverges; a stable solution converges, but it can still produce negative solutions (i.e. concentrations), which is physically impossible. Therefore, one is interested in stable and positive solutions of the differential equations considered. The imposed positivity conditions lead to conditions for the allowed size of time and space steps. Hence, it is not only the dispersion coefficient and the flow velocity, but also the selected calculation scheme which determine the step sizes allowed.

Table 1 Overview of variables in the conservation equations for the water and sediment subsystems, demonstrating at which location in the grid these variables are defined

	Water layer	Sediment
Water flow		
grid point	h	-
interface	Q, u	$\dfrac{\ell\ q}{P\ \varepsilon}$
Water quality		
grid point	c, c^* A, b (to calculate volumes) k O_x, P_x DW X_{mp}, X_{ss} J_{wa}, J_{wb}	c_{lb}, $\dot{c}_b$ P (to calculate volumes) k X_b ε
interface	uA	$\left(\dfrac{\ell}{P}\ \dfrac{q}{\varepsilon}\right)P$
	$E_x.A$	$D_{lb}.P$, $E_{lb}.P$

Which size of numerical dispersion is acceptable depends on the types of issue studied. If one is studying point sources resulting in sharp concentration gradients, one does not want the selected calculation scheme to smoothen these gradients rapidly (i.e. in an artificial way). This would not agree with the actual situation. In such cases one should select a calculation scheme resulting in a small numerical dispersion. In the case of releases from diffuse origins no steep concentration gradients are present in the surface water, which makes it possible to select a calculation scheme with a relatively large numerical dispersion. Small numerical dispersions are often linked to small possible time and space steps, which results in long calculation times on the computer. If the numerical dispersion is allowed to be larger, it is generally possible to use larger steps, involving less calculation time.

Selecting the calculation scheme thus involves finding a balance between calculation time and acceptable numerical dispersion. Which size of numerical dispersion can be accepted, depends on the types of issue studied (steep concentration gradients or not).

6.2 Numerical weight factors

The finite-difference method in its most generalized form makes use of two weight factors in its calculation scheme. A weight factor θ for time indicates whether the solution scheme is implicit or explicit, or a weighted average of these two extremes. A weight factor α for space indicates whether the solution method is backward, forward or somewhere in between backward and forward. In the case of a constant segment size, the concentration used to estimate convective flow between the grid points i and i+1, $c_{i+½}$, can be expressed as:

$$c_{i+½} = (1 - \alpha)\ c_i + \alpha\ c_{i+1} \tag{6.1}$$

with

$0 \leq \alpha \leq 1.$
By analogy:

$$c_{i-½} = (1 - \alpha)\ c_{i-1} + \alpha\ c_i \tag{6.1a}$$

A backward difference is used for $\alpha = 0$, a forward one for $\alpha = 1$.

Similarly, the concentration used to approximate the derivative with respect to time between time $j\Delta t$ and $(j+1)\Delta t$ (defined as $c_{j+½}$) reads:

$$c_i^{j+½} = \theta\ c_i^j + (1 - \theta)\ c_i^{j+1} \tag{6.2}$$

with

$0 \leq \theta \leq 1.$
An explicit calculation scheme is applied for $\theta = 1$, while an implicit calculation scheme is used for $\theta = 0$.

If the segment size is not constant but differs per grid point, the situation becomes more complicated (Fig. 16). It remains possible, however, to obtain the same type of approximation for all segments for only one value of a weight factor β (comparable to the weight factor α discussed above). We will define β in such a way that:
- $\beta = 0$ corresponds with a backward difference,
- $\beta = ½$ corresponds with a central difference and
- $\beta = 1$ corresponds with a forward difference.

We did so, because it is logical to have a value 1 for a forward difference.

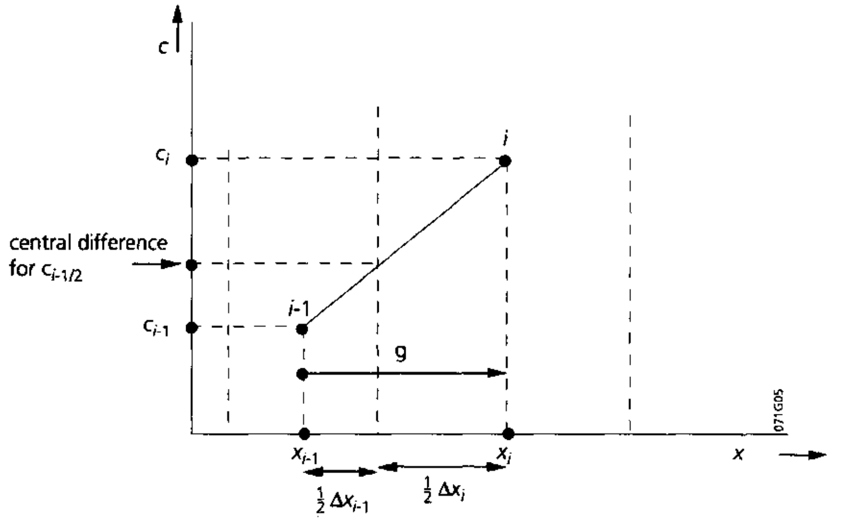


Fig. 16 Linear interpolation to determine the concentration at the interface, $c_{i-½}$ with the aid of the concentrations at grid points i-1 and i, c_{i-1} and c_i respectively

The curve between the grid points *i*-1 and *i* in Figure 16 is described by:

$$c_{i-½} = c_{i-1} + \frac{c_i - c_{i-1}}{x_i - x_{i-1}} g_{i-½} \qquad (6.3)$$

in which g is the location at which $c_{i-½}$ is calculated. We want to express g as a function of Δx_{i-1}, Δx and β with the restriction that $g = ½\Delta x_{i-1}$ if $\beta = ½$, because this is the central difference case. This results in the following function g (Fig. 17):

$$0 \leq \beta \leq ½ \; : \quad g_{i-½} = \beta \; \Delta x_{i-1}$$

$$½ < \beta \leq 1 \; : \quad g_{i-½} = ½ \; \Delta x_{i-1} + (\beta - ½) \; \Delta x_i \qquad (6.4)$$

Increasing β from 0 to 1 corresponds to going from a calculation of the concentration, $c_{i-½}$, only with the aid of c_{i-1} (fully backward) up to a calculation of $c_{i-½}$ only with the aid of c_i (fully forward). Linear interpolation takes place between these two extremes.

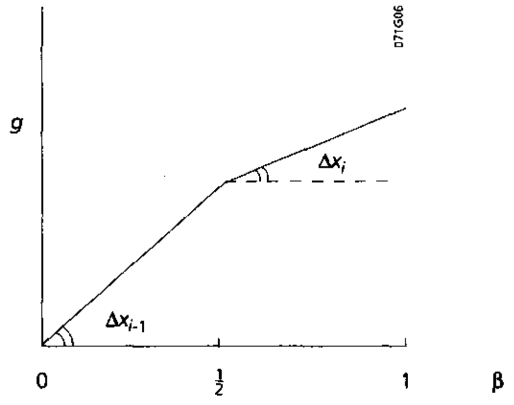


Fig. 17 The function $g_{i-½}$ with ß representing a weight factor of space and Δx_{i-1} and Δx_i the slope parameters

Eq. (6.3) can be rewritten as:

$$c_{i-½} = \left(1 - \frac{g_{i-½}}{x_i - x_{i-1}}\right) c_{i-1} + \frac{g_{i-½}}{x_i - x_{i-1}} c_i \tag{6.3a}$$

This means that the weight factor (1-α) from Eq. (6.1a) has been replaced by:

$$1 - \frac{g_{i-½}}{x_i - x_{i-1}}$$

and the weight factor α by:

$$\frac{g_{i-½}}{x_i - x_{i-1}}$$

Below, Eq. (6.3a) will be shown as:

$$c_{i-½} = (1 - NWW_{i-½})\, c_{i-1} + NWW_{i-½}\, c_i \tag{6.3b}$$

NWW, the Numerical Weight factor for the Water layer, stands for:

$$NWW_{i-½} = \frac{g_{i-½}}{x_i - x_{i-1}} \tag{6.5}$$

Similarly:

$$c_{i+½} = (1 - NWW_{i+½})\ c_i + NWW_{i+½}\ c_{i+1} \tag{6.6}$$

with

$$NWW_{i+½} = \frac{g_{i+½}}{x_{i+1} - x_i} \tag{6.5a}$$

The original weight factor α from Eqs. (6.1) and (6.1a) has now been replaced by the factor $NWW_{i-½}$ or $NWW_{i+½}$. A calculation scheme with fully forward differences according to space is described by a value of 1 for all three weight factors introduced. Table 2 shows the outline.

Table 2 Outline of weight factors introduced according to space, using the finite-difference method in TOXSWA

Difference approximation	Weight factor α	Weight factor β	*NWW*
Forward	1	1	1
Central	½	½	depending on the ratio of the segment sizes
Backward	0	0	0

By analogy:

$$c_{\text{lb}\ k-½} = (1 - NWB_{k-½})\ c_{\text{lb}\ k-1} + NWB_{k-½}\ c_{\text{lb}\ k} \tag{6.7}$$

in which the *NWB*, the Numerical Weight factor for the sediment [water Bottom], equals:

$$NWB_{k-½} = \frac{g_{i-½}}{z_k - z_{k-1}} \tag{6.8}$$

Similarly:

$$c_{\text{lb}\ k+½} = (1 - NWB_{k+½})\ c_{\text{lb}\ k} + NWB_{k+½}\ c_{\text{lb}\ k+1} \tag{6.7a}$$

with

$$NWB_{k+½} = \frac{g_{i+½}}{z_{k+1} - z_k} \tag{6.8a}$$

In the TOXSWA model, the calculation scheme can be determined by the model user by selecting the weight factors β for space and θ for time. The selected calculation scheme for the sediment does not necessarily correspond with that for the water layer, so the value of β or θ for the sediment may differ from those selected for the water layer.

6.3 Numerical dispersion

For the classical advection-diffusion equation

$$\frac{\partial c}{\partial t} = - u \frac{\partial c}{\partial x} + D \frac{\partial^2 c}{\partial x^2}, \tag{6.9}$$

so applying to a system without transformation and without external exchanges, the coefficient for numerical dispersion can be approximated by:

$$E_{\mathrm{num}} = \frac{u}{2} \left[(1 - 2\alpha) \Delta x + (1 - 2\theta) u \Delta t\right] \tag{6.10}$$

(Van Genuchten and Wieringa, 1974; Aalderink, 1993)

When differences in segment sizes are taken into account, the equation reads:

$$E_{\mathrm{num}\, i-½} = \frac{u}{2} [(1 - 2 NWW_{i-½}) (x_i - x_{i-1}) + (1 - 2\theta) u \Delta t] \tag{6.11}$$

or

$$E_{\mathrm{num}\, i+½} = \frac{u}{2} [(1 - 2 NWW_{i+½}) (x_{i+1} - x_i) + (1 - 2\theta) u \Delta t] \tag{6.11a}$$

For an explicit calculation scheme with backward differences, this results e.g. in ($\alpha = 0$ or $NWW = 0$ and $\theta = 1$):

$$E_{\mathrm{num}\, i-½} = \frac{u}{2} [(x_i - x_{i-1}) - u \Delta t] \tag{6.12}$$

In such schemes the numerical dispersion can be suppressed by appropriately choosing $(x_i - x_{i-1})$ and Δt.

The numerical dispersion in implicit schemes ($\theta = 0$) is always larger than that in explicit schemes. Note that the expression 'numerical dispersion' is somewhat misleading, because E_{num} can also be negative (e.g. if $\alpha = 1$ and $\theta = 1$), so then 'numerical steepening' occurs. An example of an implicit calculation scheme with backward differences would be ($\alpha = 0$ or $NWW = 0$ and $\theta = 0$):

$$E_{\text{num } i-½} = \frac{u}{2} \left[(x_i - x_{i-1}) + u \, \Delta t \right] \tag{6.13}$$

The approximated numerical dispersion for a Crank-Nicholson scheme ($\alpha = ½$, $\theta = ½$) with equal segment sizes results in:

$$E_{\text{num}} = 0$$

For unequal segment sizes, the factor *NWW* will not be exactly ½, which means that the approximated numerical dispersion will not equal zero. Generally, the Crank-Nicholson scheme will lead to small time and space steps, so to long calculation times.

If the numerical dispersion caused by the selected calculation scheme is approximately known, a more accurate approximation of Eq. (6.9) may be obtained, provided the appropriate difference equation uses a 'calculation' dispersion of the form:

$$E_{\text{calculation}} = E_{\text{physical}} - E_{\text{numerical}} \tag{6.14}$$

This is a better method for simulating real, measured concentration courses (Van Genuchten and Wieringa, 1974).

In the TOXSWA model, the numerical dispersion coefficients have been approximated for the water and sediment subsystems. This has been done by rewriting the advection-diffusion equations for both subsystems, expressing c^* and c_b^* as a factor multiplied by the concentration in the water and liquid phase, respectively. Next Eq. (6.10) has been applied (Cf. Wierenga and van Genuchten, 1974 and Boesten, 1986). For the water subsystem, this results in:

$$E_{\text{num},i-½} = \frac{u}{2} \left[(1 - 2NWW_{i-½}) \, (x_i - x_{i-1}) + (1 - 2\theta) \frac{u \left(1 + ssK_F n_{ss} \left(\frac{c}{c_e}\right)^{n_{ss}-1}\right)}{\left(1 + \frac{DW \, P_{z=0}}{A} K_{\text{mp}} + ssK_F n_{ss} \left(\frac{c}{c_e}\right)^{n_{ss}-1}\right)} \Delta t \right] \tag{6.15}$$

$$E_{\text{num},i+½} = \frac{u}{2} \left[(1 - 2NWW_{i+½})(x_{i+1} - x_i) + (1 - 2\theta) \frac{u \, (1 + ssK_F n_{ss} (\frac{c}{c_e})^{n_{ss}-1})}{(1 + \frac{DW \, P_{z=0}}{A} K_{\text{mp}} + ssK_F n_{ss} (\frac{c}{c_e})^{n_{ss}-1})} \Delta t \right] \tag{6.15a}$$

(Assumptions made include that ss, K_F, K_{mp} and A are constant in time.)

The numerical dispersion coefficients for the sediment subsystem are given by:

$$E_{\mathrm{num.},k-½} = \frac{\ell q}{2P}\left[(1 - 2NWB_{k-½})\,(z_k - z_{k-1}) + (1 - 2\theta)\frac{\ell q}{P\left(\epsilon + \rho_b K_F n_{wb}\left(\frac{c_{\mathrm{Ib}}}{c_e}\right)^{n_{wb}-1}\right)}\Delta t\right] \tag{6.16}$$

$$E_{\mathrm{num.},k+½} = \frac{\ell q}{2P}\left[(1 - 2NWB_{k-½})\,(z_{k+1} - z_k) + (1 - 2\theta)\frac{\ell q}{P\,(\epsilon + \rho_b K_F n_{wb}(\frac{c_{\mathrm{Ib}}}{c_e})^{n_{wb}-1})}\Delta t\right] \tag{6.16a}$$

(Assumptions made include that ε, ρ_b, K_F and P are constant in time.)

6.4 Water layer

The right-hand term of the conservation equation for the water layer, Eq. (3.6), was approximated by applying the generalised finite-difference equation. In this equation, spatial derivatives and concentrations are evaluated at time $j+½$, as defined by Eq. (6.2) which implies that θ is user defined.

$$\left(-\frac{\partial(AJ)}{\partial x}\right)_i^{j+½} - k\,(c^*\,A)_i^{j+½} + (J_{wa}O_x)_i^{j+½} - (J_{wb}P_x)_i^{j+½} \approx$$

$$\frac{(AJ)_{i-½}^{j+½} - (AJ)_{i+½}^{j+½}}{\Delta x_i} - k\,(c^*\,A)_i^{j+½} + (J_{wa}O_x)_i^{j+½} - (J_{wb}P_x)_i^{j+½} \tag{6.17}$$

The term $(AJ)_{i-½}^{j+½}$ for values of i from 2 to m is given by:

$$(AJ)_{i-½}^{j+½} = \left[uA\,(c + ssX_{ss}) - AE_x\left(\frac{\partial(c + ssX_{ss})}{\partial x}\right)\right]_{i-½}^{j+½} = \tag{6.18}$$

$$Q_{i-½}^{j+½}\,(c_{i-½}^{j+½} + ssX_{ss\,i-½}^{\;j+½}) - A_{i-½}^{j+½}\,E_{x\,i-½}^{\;j+½}\left(\frac{(c_i^{j+½} + ssX_{ss\,i}^{\;j+½}) - (c_{i-1}^{j+½} + ssX_{ss\,i-1}^{\;j+½})}{½\Delta x_i + ½\Delta x_{i-1}}\right)$$

in which

$$Q_{i-½}^{j+½} = \theta\,Q_{i-½}^{j} + (1 - \theta)\,Q_{i-½}^{j+1}$$

$$c_{i-½}^{j+½} = \theta\ c_{i-½}^{j} + (1 - \theta)\ c_{i-½}^{j+1} = \theta\ (1 - NWW_{i-½})\ c_{i-1}^{j} + \theta\ NWW_{i-½}\ c_i^{j} +$$

$$(1 - \theta)\ (1 - NWW_{i-½})\ c_{i-1}^{j+1} + (1 - \theta)\ NWW_{i-½}\ c_i^{j+1}$$

ss is constant in time and space

$$X_{ss\ i-½}^{\ j+½} = \theta\ X_{ss\ i-½}^{\ j} + (1 - \theta)\ X_{ss\ i-½}^{\ j+1} =$$

$$\theta\ (1 - NWW_{i-½})\ X_{ss\ i-1}^{\ j} + \theta\ NWW_{i-½}\ X_{ss\ i}^{\ j} +$$

$$(1 - \theta)\ (1 - NWW_{i-½})\ X_{ss\ i-1}^{\ j+1} + (1 - \theta)\ NWW_{i-½}\ X_{ss\ i}^{\ j+1}$$

$A_{i-½}^{j+½} = \theta\ A_{i-½}^{j} + (1 - \theta)\ A_{i-½}^{j+1}$ and

$$A_{i-½} = f(h_{i-½})$$

$$E_{i-½}^{j+½} = \theta\ E_{i-½}^{j} + (1 - \theta)\ E_{i-½}^{j+1}$$

$$c_i^{j+½} = \theta\ c_i^{j} + (1 - \theta)\ c_i^{j+1}$$

$$X_{ss\ i}^{\ j+½} = \theta\ X_{ss\ i}^{\ j} + (1 - \theta)\ X_{ss\ i}^{\ j+1}$$

$$c_{i-1}^{j+½} = \theta\ c_{i-1}^{j} + (1 - \theta)\ c_{i-1}^{j+1}$$

$$X_{ss\ i-1}^{\ j+½} = \theta\ X_{ss\ i-1}^{\ j} + (1 - \theta)\ X_{ss\ i-1}^{\ j+1}$$

By analogy the term $(AJ)_{i+½}^{j+½}$ for values of i from 1 to $(m$-1) reads:

$$(AJ)_{i+½}^{j+½} = [uA\ (c + ssX_{ss}) - AE_x\left(\frac{\partial(c + ssX_{ss})}{\partial x}\right)]_{i+½}^{j+½} = \tag{6.19}$$

$$Q_{i+½}^{j+½}\ (c_{i+½}^{j+½} + ssX_{ss\ i+½}^{\ j+½}) - A_{i+½}^{j+½}\ E_{x\ i+½}^{\ j+½}\left(\frac{(c_{i+1}^{j+½} + ssX_{ss\ i+1}^{\ j+½}) - (c_i^{j+½} + ssX_{ss\ i}^{\ j+½})}{½\Delta x_{i+1} + ½\Delta x_i}\right)$$

in which

$$Q_{i+½}^{j+½} = \theta\ Q_{i+½}^{j} + (1 - \theta)\ Q_{i+½}^{j+1}$$

$$c_{i+½}^{j+½} = \theta\ c_{i+½}^{j} + (1 - \theta)\ c_{i+½}^{j+1} = \theta\ (1 - NWW_{i+½})\ c_i^j + \theta\ NWW_{i+½}\ c_{i+1}^j +$$

$$(1 - \theta)\ (1 - NWW_{i+½})\ c_i^{j+1} + (1 - \theta)\ NWW_{i+½}\ c_{i+1}^{j+1}$$

$$X_{ss\ i+½}^{\ j+½} = \theta\ (1 - NWW_{i+½})\ X_{ss\ i}^{\ j} + \theta\ NWW_{i+½}\ X_{ss\ i+1}^{\ j} +$$

$$(1 - \theta)\ (1 - NWW_{i+½})\ X_{ss\ i}^{\ j+1} + (1 - \theta)\ NWW_{i+½}\ X_{ss\ i+1}^{\ j+1}$$

$A_{i+½}^{j+½} = \theta\ A_{i+½}^{j} + (1 - \theta)\ A_{i+½}^{j+1}$ with

$$A_{i+½} = f(h_{i+½})$$

$$E_{i+½}^{j+½} = \theta\ E_{i+½}^{j} + (1 - \theta)\ E_{i+½}^{j+1}$$

$$c_{i+1}^{j+½} = \theta\ c_{i+1}^{j} + (1 - \theta)\ c_{i+1}^{j+1}$$

$$X_{ss\ i+1}^{\ j+½} = \theta\ X_{ss\ i+1}^{\ j} + (1 - \theta)\ X_{ss\ i+1}^{\ j+1}$$

The term - $k(c^*A)_i^{j+½}$ equals:

$$-k\ (c^*\ A)_i^{j+½} = -k(c^{*\ j+½}_{\ i}\ A_i^{j+½}) \tag{6.20}$$

in which

$$c^{*\ j+½}_{\ i} = \theta\ c^{*\ j}_{\ i} + (1 - \theta)\ c^{*\ j+1}_{\ i}$$

$$A_i^{j+½} = \theta\ A_i^{j} + (1 - \theta)\ A_i^{j+1}$$

The term $(J_{wa}O_x)_i^{j+½}$ equals:

$$(J_{wa}O_x)_i^{j+½} = (J_{wa\ i}^{\ j+½}\ O_{x\ i}^{\ j+½}) = -k_{t,l}\left(c_i^{j+½} - \frac{c_{a\ i}^{\ j+½}}{K_H}\right) O_{x\ i}^{\ j+½} \tag{6.21}$$

in which

$$c_{a\ i}^{\ j+½} = \theta\ c_{a\ i}^{\ j} + (1 - \theta)\ c_{a\ i}^{\ j+1}$$

(it has been assumed that c_a, the pesticide background concentration in the air, is constant)

$$O_{x\ i}^{\ j+½} = \theta\ O_{x\ i}^{\ j} + (1 - \theta)\ O_{x\ i}^{\ j+1}$$

The term $-(J_{wb}P_x)_i^{j+½}$ reads:

$$-(J_{wb}P_x)_i^{j+½} = -(J_{wb,adv\ i}^{\ j+½} + J_{wb,dif\ i}^{\ j+½})\ P_{x\ i}^{\ j+½}$$

For downward water flow ($q > 0$):

$$= - \left(\frac{\ell}{P_{z=0\ i}^{\ j+½}} \ q_i^{j+½} \ c_i^{j+½} - \left(\varepsilon \ D_{\rm lb} \frac{\partial c_{\rm lb}}{\partial z} \right)_{z=0\ i}^{j+½} \right) P_{x\ i}^{\ j+½}$$

$$= - \left(\frac{\ell}{P_{z=0\ i}^{\ j+½}} \ q_i^{j+½} \ c_i^{j+½} - \varepsilon_i^{j+½} \ D_{{\rm lb}\ i}^{\ j+½} \frac{(c_{{\rm lb}\ i,\ k=1}^{\ j+½} - c_i^{j+½})}{½\ \Delta z_1} \right) P_{x\ i}^{\ j+½} \quad (6.22a)$$

For upward water flow ($q < 0$):

$$= - \left(\frac{\ell}{P_{z=0\ i}^{\ j+½}} \ q_i^{j+½} \ c_{{\rm lb}\ i,\ k=1}^{\ j+½} - \left(\varepsilon \ D_{\rm lb} \frac{\partial c_{\rm lb}}{\partial z} \right)_{z=0\ i}^{j+½} \right) P_{x\ i}^{\ j+½}$$

$$= - \left(\frac{\ell}{P_{z=0\ i}^{\ j+½}} \ q_i^{j+½} \ c_{{\rm lb}\ i,\ k=1}^{j+½} - \varepsilon_i^{j+½} \ D_{{\rm lb}\ i}^{\ j+½} \frac{(c_{{\rm lb}\ i,\ k=1}^{\ j+½} - c_i^{j+½})}{½\ \Delta z_1} \right) P_{x,\ i}^{\ j+½} \quad (6.22b)$$

in which

$P_{z=0\ i}^{\ j+½} = P_{z=0\ i}^{\ j}$ (constant in time)

$q_i^{j+½} = \theta \ q_i^j + (1 - \theta) \ q_i^{j+1}$

$\varepsilon_i^{j+½} = \varepsilon_i^j$ (contant in time)

$D_{{\rm lb}\ i}^{\ j+½} = D_{{\rm lb}\ i}^{\ j}$ (constant in time)

$P_{x\ i}^{\ j+½} = P_{x\ i}^{\ j} \ (= P_{z=0\ i}^{\ j})$ (constant in time)

$c_{\mathrm{lb}\,i,\,k=1}^{\;j+½} = c_{\mathrm{lb}\,i,\,k=1}^{\;j}$ (see also section 6.9.)

The left-hand term of the conservation equation for the water layer, Eq. (3.6), was also approximated with the aid of the finite-difference method.

$$\left(\frac{\partial c^* A}{\partial t}\right)_i^j \approx \frac{c^{*\,j+1}_{\;i} A_i^{j+1} - c^{*\,j}_{\;i} A_i^j}{\Delta t} \tag{6.23}$$

The variable c^*, the total mass concentration of the substance in the water layer, may be written as a factor multiplied by c, the mass concentration in the water phase, according to Eq. (6.24):

$$c^{*\,j}_{\;i} = \left\{1 + \frac{DW\, P_{z=0}}{A} K_{\mathrm{mp}} + ss\, K_{\mathrm{F},ss} \left(\frac{c_i^j}{c_{e,ss}}\right)^{n_{ss}-1}\right\} c_i^j \tag{6.24}$$

This factor depends on c, so c is calculated for known values of c^* in an iterative way (see section 6.8). Similarly the variable X_{ss}, the content of the substance sorbed to suspended solids, may be expressed as a function of c:

$$X_{\mathrm{ss}\,i}^{\;j} = \left\{K_{\mathrm{F},ss} \left(\frac{c_i^j}{c_{e,ss}}\right)^{n_{ss}-1}\right\} c_i^j \tag{6.25}$$

Substituting these Eqs. (6.24) and (6.25) into the numerical approximations of the conservation equation (Eqs. (6.17) up to (6.23) inclusive) and rearrangement of all terms lead to the following equations:

$$\begin{pmatrix} LDD & LBD & O & . & . & . & . \\ LOD & LDD & LBD & O & . & . & . \\ O & LOD & LDD & LBD & O & & \\ . & . & . & . & . & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & . & . & O & LOD & LDD & LBD \\ . & . & . & . & O & LOD & LDD \end{pmatrix} \begin{pmatrix} c_1^{j+1} \\ c_2^{j+1} \\ . \\ . \\ c_{i-1}^{j+1} \\ c_i^{j+1} \\ c_{i+1}^{j+1} \\ . \\ . \\ c_m^{j+1} \end{pmatrix} = \begin{pmatrix} RDD & RBD & O & . & . & . & . \\ ROD & RDD & RBD & O & . & . & . \\ O & ROD & RDD & RBD & O & & \\ . & . & . & . & . & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & . & . & O & ROD & RDD & RBD \\ . & . & . & . & O & ROD & RDD \end{pmatrix} \begin{pmatrix} c_1^{j} \\ c_2^{j} \\ . \\ . \\ c_{i-1}^{j} \\ c_i^{j} \\ c_{i+1}^{j} \\ . \\ . \\ c_m^{j} \end{pmatrix} + \begin{pmatrix} RV_1 \\ RV_2 \\ . \\ . \\ RV_{i-1} \\ RV_i \\ RV_{i+1} \\ . \\ . \\ RV_m \end{pmatrix} \tag{6.26}$$

This shows a left-hand tridiagonal matrix, a right-hand tridiagonal matrix and a separate right-hand vector (composed of constants), as well as the two concentration vectors at times j and $j+1$.

The elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD*, *RBD* and *RV* in row i are (in the case of downward water flow ($q > 0$) in the sediment subsystem):

$$LOD = -\frac{Q_{i-½}^{j+½}}{\Delta x_i}\,(1-\theta)\,(1-NWW_{i-½})\,\Delta t\left(1 + ss\,K_F\left(\frac{c_{i-1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right) -$$

$$\frac{2A_{i-½}^{j+½}\,E_{x\,i-½}^{\,j+½}}{\Delta x_i\,(\Delta x_i + \Delta x_{i-1})}\,(1-\theta)\,\Delta t\left(1 + ss\,K_F\left(\frac{c_{i-1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$LDD = -\frac{Q_{i-½}^{j+½}}{\Delta x_i}\,(1-\theta)\,NWW_{i-½}\,\Delta t\left(1 + ss\,K_F\left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right) +$$

$$\frac{Q_{i+½}^{j+½}}{\Delta x_i} (1 - \theta) (1 - NWW_{i+½}) \Delta t \left(1 + ss\ K_F \left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right) +$$

$$\frac{2A_{i-½}^{j+½}\ E_{x\ i-½}^{\ j+½}}{\Delta x_i\ (\Delta x_i + \Delta x_{i-1})} (1 - \theta) \Delta t \left(1 + ss\ K_F \left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right) +$$

$$\frac{2A_{i+½}^{j+½}\ E_{x\ i+½}^{\ j+½}}{\Delta x_i\ (\Delta x_{i+1} + \Delta x_i)} (1 - \theta) \Delta t \left(1 + ss\ K_F \left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right) +$$

$$\left(1 + \frac{DW\ P_{z=0\ i}^{\ \ j+1}}{A_i^{j+1}} K_{mp} + ss\ K_F \left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right) (A_i^{j+1} + kA_i^{j+½} (1 - \theta) \Delta t) +$$

$$k_{t,1}\ O_{x\ i}^{\ j+½} (1 - \theta) \Delta t + \frac{\ell}{P_{z=0}} q_i^{j+½}\ P_{x\ i}^{\ j} (1 - \theta) \Delta t +$$

$$\frac{2\varepsilon_i^j\ D_{lb}\ P_{x\ i}^{\ j}}{\Delta z_1} (1 - \theta) \Delta t$$

$$LBD = \frac{Q_{i+½}^{j+½}}{\Delta x_i} (1 - \theta)\ NWW_{i+½}\ \Delta t \left(1 + ss\ K_F \left(\frac{c_{i+1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right) -$$

$$\frac{2A_{i+½}^{j+½}\ E_{x\ i+½}^{\ j+½}}{\Delta x_i\ (\Delta x_{i+1} + \Delta x_i)} (1 - \theta) \Delta t \left(1 + ss\ K_F \left(\frac{c_{i+1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$ROD = \frac{Q_{i-½}^{j+½}}{\Delta x_i}\ \theta\ (1 - NWW_{i-½})\ \Delta t \left(1 + ss\ K_F \left(\frac{c_{i-1}^j}{c_e}\right)^{n_{ss}-1}\right) +$$

$$\frac{2A_{i-½}^{j+½}\ E_{x\,i-½}^{\ j+½}}{\Delta x_i\ (\Delta x_i + \Delta x_{i-1})}\ \theta\ \Delta t \left(1 + ss\ K_F \left(\frac{c_{i-1}^j}{c_e}\right)^{n_{ss}-1}\right)$$

$$RDD = \frac{Q_{i-½}^{j+½}}{\Delta x_i}\ \theta\ NWW_{i-½}\ \Delta t \left(1 + ss\ K_F \left(\frac{c_i^j}{c_e}\right)^{n_{ss}-1}\right) -$$

$$\frac{Q_{i+½}^{j+½}}{\Delta x_i}\ \theta\ (1 - NWW_{i+½})\ \Delta t \left(1 + ss\ K_F \left(\frac{c_i^j}{c_e}\right)^{n_{ss}-1}\right) -$$

$$\frac{2A_{i-½}^{j+½}\ E_{x\,i-½}^{\ j+½}}{\Delta x_i\ (\Delta x_i + \Delta x_{i-1})}\ \theta\ \Delta t \left(1 + ss\ K_F \left(\frac{c_i^j}{c_e}\right)^{n_{ss}-1}\right) -$$

$$\frac{2A_{i+½}^{j+½}\ E_{x\,i+½}^{\ j+½}}{\Delta x_i\ (\Delta x_{i+1} + \Delta x_i)}\ \theta\ \Delta t \left(1 + ss\ K_F \left(\frac{c_i^j}{c_e}\right)^{n_{ss}-1}\right) +$$

$$\left(1 + \frac{DW\ P_{z=0}}{A_i^j}K_{mp} + ss\ K_F \left(\frac{c_i^j}{c_e}\right)^{n_{ss}-1}\right)(A_i^j - kA_i^{j+½}\ \theta\ \Delta t) -$$

$$k_{t,1}\, O_{x\,i}^{\;j+½}\, \theta\, \Delta t \;-\; \frac{\ell}{P_{z=0}}\, q_i^{j+½}\, P_{x\,i}^{\;j}\, \theta\, \Delta t \;-$$

$$\frac{2\varepsilon_i^j\, D_{\mathrm{lb}}\, P_{x\,i}^{\;j}}{\Delta z_1}\, \theta\, \Delta t$$

$$RBD \;=\; -\,\frac{Q_{i+½}^{j+½}}{\Delta x_i}\, \theta\, NWW_{i+½}\, \Delta t \left(1 \;+\; ss\; K_{\mathrm{F}} \left(\frac{c_{i+1}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right) +$$

$$\frac{2A_{i+½}^{j+½}\; E_{x\;i+½}^{\;j+½}}{\Delta x_i\; (\Delta x_{i+1} \;+\; \Delta x_i)}\, \theta\, \Delta t \left(1 \;+\; ss\; K_{\mathrm{F}} \left(\frac{c_{i+1}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$RV \;=\; k_{t,\,1}\, O_{x\,i}^{\;j+½}\, \frac{c_{\mathrm{a}\,i}^{\;j+½}}{K_{\mathrm{H}}}\Delta t \;+\; 2\frac{\varepsilon_i^j\, D_{\mathrm{lb}}\, P_{x\,i}^{\;j}}{\Delta z_1} c_{\mathrm{lb}\;i,\,k=1}^{\;j}\; \Delta t$$

In the case of upward water flow ($q < 0$) in the sediment subsystem the right-hand vector RV contains an additional term:

$$-\,\frac{\ell}{P_{z=0\;i}^{\;j}}\; q_i^{j+½}\; c_{\mathrm{lb}\;i,\,k=1}^{\;j}\; P_{x\,i}^{\;j+½}\; \Delta t$$

and the penultimate terms containing $q_i^{j+½}$ in LDD and RDD disappear.
Eq. (6.26) can be written as:

$$A.c^{j+1} \;=\; B.c^{\,j} \;+\; d \tag{6.26a}$$

in which A and B are tridiagonal matrices and d is a vector.

The solution of Eq. (6.26a) is:

$$c^{j+1} = A^{-1}\,(B.c^{j} + d) \tag{6.26b}$$

in which A^{-1} is the inversed tridiagonal matrix A (see also Press at al, 1986). The value of c_i^{j+1} at time j+1 can be calculated for known values of c_i^j at time j (cf. also section 6.8).

6.5 Boundary conditions and initial condition for water layer

6.5.1 Lower boundary condition

The boundary condition at the end of the part of the ditch considered reads:

$$J_{m+\frac{1}{2}} = \left(u\,(c + ssX_{ss}) - E_x \frac{\partial (c + ssX_{ss})}{\partial x} \right)_{m+\frac{1}{2}} \tag{6.27}$$

This was approximated by defining a certain number of additional grid points, *ebt* (end buffer, total number), behind the last grid point *m* in the ditch. The mass flux leaving the water layer at grid point *ebt* is calculated as:

$$J_{ebt+\frac{1}{2}} = \max[(u\,(c + ssX_{ss}))_{ebt+\frac{1}{2}}\,,\,0] \tag{6.28}$$

This implies that an outgoing water flow (u>0) produces only advective transport leaving the water subsystem, while an ingoing water flow (u<0) causes water free of pesticide to enter the subsystem.

Eq. (6.28) implies a discontinuity from a physical point of view; changing the direction of flow at the end of the water subsystem, $u_{ebt+\frac{1}{2}}$, entails a discontinuous change of the flux $J_{ebt+\frac{1}{2}}$. This has been compensated for by extending the calculations for an additional number of grid points, *ebt*, called the end buffer, behind the last grid point in the ditch, *m*.
This buffer is a fictitious entity and serves only as a solution for the numerical problem created by the discontinuous boundary condition. Therefore, the substance can only be (re)distributed or transported in the buffer. The substance cannot originate or disappear in the buffer.
This means that certain processes can occur in the buffer behind the part of the ditch considered, while others cannot:
— advection and dispersion of substance do occur, but only advection occurs in the last half segment;
— sorption to suspended solids does not occur;
— sorption to macrophytes does not occur;
— there is no volatilisation to the atmosphere;
— there is no exchange with the sediment (so no advective or diffusive flux to or from the sediment) and

— there is no transformation.

In the end buffer the elements *LOD, LDD, LBD, ROD, RDD, RBD* and *RV* change. Annex 1 describes these elements for the grid points $m+1$ up to ebt-1 inclusive.

When the water flow in the ditch is positive, the lower boundary condition is defined by assuming an outgoing advective flux and by neglecting the dispersion in the last half segment. In this case the concentration $c^j_{ebt+½}$ cannot be calculated with the aid of the concentration c^j_{ebt+1}, so it has been assumed that $c^j_{ebt+½} = c^j_{ebt}$.
Similarly, $X_{ss}{}^j_{ebt+½} = X_{ss}{}^j_{ebt}$. Annex 2 provides the elements *LOD, LDD, LBD, ROD, RDD, RBD* and *RV* for a positive water flow at grid point *ebt*.

When the water flow is negative, the incoming substance flux is zero. (There is again no dispersion in the last half segment.) This implies that $c^j_{ebt+½} = 0$ and $X_{ss}{}^j_{ebt+½} = 0$. Annex 3 provides the elements *LOD, LDD, LBD, ROD, RDD, RBD* and *RV* for a positive water flow at grid point *ebt*.

6.5.2 Upper boundary condition

When the water flow in the ditch is positive, water free of pesticide enters the subsystem at the upper boundary. Hence, $(AJ)^j_{½} = 0$. The elements *LOD, LDD, LBD, ROD, RDD, RBD* and *RV* are presented in Annex 4.

In situations of low hydraulic gradients, and where wind influence and pumping stations are present, it is quite common to encounter alternating flow directions in watercourses. When the water flow alternates between being positive and negative, the situation becomes similar to that of the lower boundary. This means that the boundary condition at the beginning of the part of the ditch considered equals:

$$J_{fbt+½} = \left(u\ (c + ssX_{ss}) - E_x \frac{\partial(c + ssX_{ss})}{\partial x} \right)_{fbt+½} \qquad (6.29)$$

This situation was approximated by defining a certain number of additional grid points *fbt* in front of the first grid point in the ditch (front buffer, total number). The mass flux leaving the water subsystem is now calculated according to:

$$J_{½} = \min[(u\ (c + ssX_{ss}))_{½}\ ,\ 0] \qquad (6.30)$$

An outgoing, negative water flow causes only advective transport, while an ingoing, positive water flow means that water free of pesticide enters. The added front buffer with *fbt* segments compensates for this discontinuous boundary condition, analogously to the lower boundary.

For the grid points 2 up to *fbt* inclusive in the front buffer this means that:
— sorption to suspended solids does not occur;

— sorption to macrophytes does not occur;
— there is no volatilisation;
— no exchange with sediment occurs and
— there is no transformation.

For the situation of alternating positive and negative water flows, Annex 5 shows the elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD*, *RBD* and *RV* for the grid points $fb = 2$ up to $fb = fbt$ inclusive in the front buffer.

When there is a negative water flow, the upper boundary condition consists of an outgoing advective flux; dispersion is neglected in the first half segment. The concentration $c_{fb=½}$ and $X_{ss,fb=½}$ are estimated by $c_{fb=½} = c_{fb=1}$ and $X_{ss,fb=½} = X_{ss,fb=1}$. Annex 6 presents the elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD*, *RBD* and *RV* for the grid point $fb = 1$ in the front buffer in the case of a negative water flow.

When the water flow is positive, the incoming substance flux is zero. (Again there is no dispersion in the first half segment.) Hence, $c_{fb=½} = 0$ and $X_{ss,fb=½} = 0$. Annex 7 provides the elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD*, *RBD* and *RV* at grid point $fb = 1$ for a positive water flow if a front buffer is present (i.e. for alternating flow directions).

The upper and lower boundary conditions are defined in such a way that dispersion in the outer half segments is neglected. This implies that the substance cannot enter the subsystem by dispersion across the boundaries if there is a higher concentration outside the buffers than inside. Therefore, the mass balance will continue to represent the full 100% of the substance originally added to the subsystem.

The final system of numerical equations can now be composed with the aid of section 6.5 and the upper and lower boundary conditions described above. This results in a matrix equation comparable to Eq. (6.26a) for each situation. This matrix equation can be solved for a specified initial condition.

6.5.3 Initial condition

Two examples of initial conditions are worked out here: an input from diffuse origins and a point-type input, both taking place at time $t = 0$.

The first example may be air drift deposition, e.g of y kg of active ingredient per m^2 at time $t = 0$. It is assumed that the pesticide mixes instantaneously with the water layer and that sorption equilibrium with suspended solids and macrophytes is instantaneous. At time $t = 0$ none of the substance is yet sorbed to the sediment. The load per running metre of ditch is $O_x.y$ kg a.i./m' dissolved in a volume of A m^3. The substance concentration at time $t = 0$ is:

$$c^* = \frac{O_x \cdot y}{A} \tag{6.31}$$

The initial total mass concentration in the buffer(s) equals zero.

In the second example, there is a point-type input in one segment of the water subsystem, represented by e.g. y kg a.i. The total mass concentration c^* in this segment at time $t = 0$ equals:

$$c^* = \frac{y}{A \cdot \Delta x} \tag{6.32}$$

The initial mass concentration in all other segments equals zero.

Given these initial concentrations, the system of numerical equations representing the mass conservation equation of the water subsystem, Eq. (6.26), can now be solved.

6.6 Sediment

The right-hand terms of the mass conservation equation of the sediment, Eq. (3.12), are approximated by the following generalised finite-difference equation, which evaluates the spatial derivatives and the concentrations at time j+½, as defined again by Eq.(6.2).

$$\left(- \frac{\partial (PJ_{\mathrm{lb}})}{\partial z} \right)_k^{j+½} - k_{\mathrm{b}} \, (c_b^* \, P)_k^{j+½} \approx$$

$$\frac{(PJ_{\mathrm{lb}})_{k-½}^{j+½} - (PJ_{\mathrm{lb}})_{k+½}^{j+½}}{\Delta z_k} - k_{\mathrm{b}} . c_{\mathrm{b}}{}_k^{*\,j+½} . P_k^{j+½} \tag{6.33}$$

The term $(PJ_{\mathrm{lb}})_{k-½}^{j+½}$ for values of k from 2 to n is given by:

$$(PJ_{\mathrm{lb}})_{k-½}^{j+½} = \left[\frac{P.\ell}{P} . q . c_{\mathrm{lb}} - P \, \varepsilon \, (E_{\mathrm{lb}} + D_{\mathrm{lb}}) \frac{\partial c_{\mathrm{lb}}}{\partial z} \right]_{k-½}^{j+½}$$

$$= \ell . q^{j+½} . c_{\mathrm{lb}}{}_{k-½}^{j+½} - P_{k-½}^{j+½} \, \varepsilon_{k-½}^{j+½} \, (E_{\mathrm{lb}}{}_{k-½}^{j+½} + D_{\mathrm{lb}}{}_{k-½}^{j+½}) \left(\frac{c_{\mathrm{lb}}{}_k^{j+½} - c_{\mathrm{lb}}{}_{k-1}^{j+½}}{½\Delta z_k + ½\Delta z_{k-1}} \right)$$

$$= \ell . q^{j+½} . c_{\mathrm{lb}\,k-½}^{\;j+½} - P_{k-½}^{j}\, \varepsilon_{k-½}^{j} \left(E_{\mathrm{lb}\,k-½}^{\;j+½} + D_{\mathrm{lb}\,k-½}^{\;j}\right) \left(\frac{c_{\mathrm{lb}\,k}^{\;j+½} - c_{\mathrm{lb}\,k-1}^{\;j+½}}{½\Delta z_k + ½\Delta z_{k-1}} \right) \tag{6.34}$$

in which

$$q^{j+½} = \theta\; q^{j} + (1 - \theta)\; q^{j+1}$$

$$c_{\mathrm{lb}\,k-½}^{\;j+½} = \theta\; c_{\mathrm{lb}\,k-½}^{\;j} + (1 - \theta)\; c_{\mathrm{lb}\,k-½}^{\;j+1} = \theta\; (1 - NWB_{k-½})\; c_{\mathrm{lb}\,k-1}^{\;j} +$$

$$\theta\; NWB_{k-½}\; c_{\mathrm{lb}\,k}^{\;j} + (1 - \theta)\; (1 - NWB_{k-½})\; c_{\mathrm{lb}\,k-1}^{\;j+1} + (1 - \theta)\; NWB_{k-½}\; c_{\mathrm{lb}\,k}^{\;j+1}$$

$$P_{k-½}^{j} = b + 2\sum_{p=1}^{k-1} \Delta z_p . \tan(½\beta) + 2\; (h_{\mathrm{w}} + \sum_{p=1}^{k-1}\Delta z_p)\sqrt{s_1^{\,2}+1}$$

$\varepsilon_{k-½}^{j}$ is given as a function of depth

$$E_{\mathrm{lb}\,k-½}^{\;j+½} = \theta\; E_{\mathrm{lb}\,k-½}^{\;j} + (1 - \theta)\; E_{\mathrm{lb}\,k-½}^{\;j+1}$$

$$E_{\mathrm{lb}\,k-½}^{\;j} = (L_{\mathrm{dis}}\; |w|)_{k-½}^{j}$$

$$D_{\mathrm{lb}\,k-½}^{\;j} = (\lambda\; D_{\mathrm{w}})_{k-½}^{j}$$

$$c_{\mathrm{lb}\,k}^{\;j+½} = \theta\; c_{\mathrm{lb}\,k}^{\;j} + (1 - \theta)\; c_{\mathrm{lb}\,k}^{\;j+1}$$

$$c_{\mathrm{lb}\,k-1}^{\;j+½} = \theta\; c_{\mathrm{lb}\,k-1}^{\;j} + (1 - \theta)\; c_{\mathrm{lb}\,k-1}^{\;j+1}$$

P, ε and D_{lb} are constant in time, so their values at time j+½ equal their values at time j.

Similarly, $(PJ_{\mathrm{lb}})_{k+½}^{j+½}$ for values of k from 1 to n-1 reads:

$$(PJ_{lb})_{k+½}^{j+½} = \left[\frac{P \cdot \ell}{P} \cdot q \cdot c_{lb} - P\ \varepsilon\ (E_{lb} + D_{lb})\frac{\partial c_{lb}}{\partial z}\right]_{k+½}^{j+½} =$$

$$\ell \cdot q^{j+½} \cdot c_{lb\ k+½}^{\ j+½} - P_{k+½}^{j}\ \varepsilon_{k+½}^{j}\ (E_{lb\ k+½}^{\ j+½} + D_{lb\ k+½}^{\ j})\left(\frac{c_{lb\ k+1}^{\ j+½} - c_{lb\ k}^{\ j+½}}{½\Delta z_{k+1} + ½\Delta z_{k}}\right) \tag{6.35}$$

in which

$$c_{lb\ k+½}^{\ j+½} = \theta\ c_{lb\ k+½}^{\ j} + (1 - \theta)\ c_{lb\ k+½}^{\ j+1} = \theta\ (1 - NWB_{k+½})\ c_{lb\ k}^{\ j} + \theta\ NWB_{k+½}\ c_{lb\ k+1}^{\ j} +$$

$$(1 - \theta)\ (1 - NWB_{k+½})\ c_{lb\ k}^{\ j+1} + (1 - \theta)\ NWB_{k+½}\ c_{lb\ k+1}^{\ j+1}$$

$$P_{k+½}^{j} = b + 2\sum_{p=1}^{k} \Delta z_p.\tan(½\beta) + 2\ (h_w + \sum_{p=1}^{k}\Delta z_p)\sqrt{s_1^2+1}$$

$$E_{lb\ k+½}^{\ j+½} = \theta\ E_{lb\ k+½}^{\ j} + (1 - \theta)\ E_{lb\ k+½}^{\ j+1}$$

$$c_{lb\ k+1}^{\ j+½} = \theta\ c_{lb\ k+1}^{\ j} + (1 - \theta)\ c_{lb\ k+1}^{\ j+1}$$

The term $(-k_b.c_{b\ k}^{*\ j+½}.P_k^{j+½})$ equals:

$$-\ k_b\ c_{b\ k}^{*\ j+½}\ P_k^{j+½} = -\ k_b\ c_{b\ k}^{*\ j+½}\ P_k^{j} \tag{6.36}$$

with

$$c_{b\ k}^{*\ j+½} = \theta\ c_{b\ k}^{*\ j} + (1 - \theta)\ c_{b\ k}^{*\ j+1}$$

The left-hand term of the mass conservation equation for the sediment, Eq. (3.12), is also approximated by means of the finite-difference method.

$$\left(P \frac{\partial c_b^*}{\partial t} \right)_k^j \approx \frac{P_k^j (c_{b\,k}^{*\,j+1} - c_{b\,k}^{*\,j})}{\Delta t} \tag{6.37}$$

The variable c_b^* may be written as a factor multiplied by c_{lb} according to Eq. (6.38). This factor depends on c_{lb} as well, so c_{lb} is calculated for known values of c_b^* in a iterative way (see also section 6.8).

$$c_{b\,k}^{*\,j} = \left(\varepsilon + \rho_b \, K_{F,wb} \left(\frac{c_{lb\,k}^{\;j}}{c_{e,wb}} \right)^{n_{wb}-1} \right) c_{lb\,k}^{\;j} \tag{6.38}$$

Substituting Eq. (6.38) in the numerical approximations of the conservation equation, Eq. (6.33) up to (6.37) inclusive, and rearrangement of all terms lead to the following system of equations.

$$\begin{pmatrix} LDD & LBD & O & . & . & . & . \\ LOD & LDD & LBD & O & . & . & . \\ O & LOD & LDD & LBD & O & & \\ . & . & . & . & . & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & . & . & O & LOD & LDD & LBD \\ . & . & . & . & O & LOD & LDD \end{pmatrix} \begin{pmatrix} c_{lb\,1}^{\;j+1} \\ c_{lb\,2}^{\;j+1} \\ . \\ . \\ c_{lb\,k-1}^{\;j+1} \\ c_{lb\,k}^{\;j+1} \\ c_{lb\,k+1}^{\;j+1} \\ . \\ . \\ c_{lb\,n}^{\;j+1} \end{pmatrix} = \begin{pmatrix} RDD & RBD & O & . & . & . & . \\ ROD & RDD & RBD & O & . & . & . \\ O & ROD & RDD & RBD & O & & \\ . & . & . & . & . & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & . & . & O & ROD & RDD & RBD \\ . & . & . & . & O & ROD & RDD \end{pmatrix} \begin{pmatrix} c_{lb\,1}^{\;j} \\ c_{lb\,2}^{\;j} \\ . \\ . \\ c_{lb\,k-1}^{\;j} \\ c_{lb\,k}^{\;j} \\ c_{lb\,k-1}^{\;j} \\ . \\ . \\ c_{lb\,n}^{\;j} \end{pmatrix} + \begin{pmatrix} RV_1 \\ RV_2 \\ . \\ . \\ RV_{k-1} \\ RV_k \\ RV_{k+1} \\ . \\ . \\ RV_n \end{pmatrix} \tag{6.39}$$

This includes vectors for the concentrations at times j and $j+1$ as well as a right-hand and a left-hand tridiagonal matrix and a right-hand vector.

The elements LOD[8], LDD[6], LBD, ROD, RDD, RBD and RV at row k are given by:

$$LOD = - \frac{\ell \, q^{j+½}}{\Delta z_k} (1 - \theta) (1 - NWB_{k-½}) \frac{\Delta t}{P_k^j} \, sofd_{k-½} -$$

[8] The terms $sofd_{k-½}$ and $sofd_{k+½}$ go from 1 to 0 as soon as the dispersive flux in the sediment exceeds the advective flux (See section 4.2 and Figure 11).

$$\frac{P_{k-½}^{j}\ \varepsilon_{k-½}^{j}\ (E_{\mathrm{lb}}{}_{k-½}^{j+½} + D_{\mathrm{lb}}{}_{k-½}^{j})\ (1 - \theta)\Delta t}{\Delta z_k\ (½\ \Delta z_k + ½\ \Delta z_{k-1})\ P_k^j}$$

$$LDD = \frac{\ell\ q^{j+½}}{\Delta z_k}\ (1 - \theta)\ (1 - NWB_{k+½})\ \frac{\Delta t}{P_k^j}\ sofd_{k+½} +$$

$$\frac{P_{k+½}^{j}\ \varepsilon_{k+½}^{j}\ (E_{\mathrm{lb}}{}_{k+½}^{j+½} + D_{\mathrm{lb}}{}_{k+½}^{j})\ (1 - \theta)\Delta t}{\Delta z_k\ (½\ \Delta z_{k+1} + ½\ \Delta z_k)\ P_k^j} -$$

$$\frac{\ell\ q^{j+½}}{\Delta z_k}\ (1 - \theta)\ NWB_{k-½}\ \frac{\Delta t}{P_k^j}\ sofd_{k-½} +$$

$$\frac{P_{k-½}^{j}\ \varepsilon_{k-½}^{j}\ (E_{\mathrm{lb}}{}_{k-½}^{j+½} + D_{\mathrm{lb}}{}_{k-½}^{j})\ (1 - \theta)\Delta t}{\Delta z_k\ (½\ \Delta z_k + ½\ \Delta z_{k-1})\ P_k^j} +$$

$$\left(\varepsilon_k^j + \rho_{\mathrm{b}}{}_k^j\ K_{\mathrm{F}} \left(\frac{c_{\mathrm{lb}}{}_k^{j-1}}{c_{\mathrm{e}}}\right)^{n_{\mathrm{wb}}-1}\right)\left(1 + k_{\mathrm{b}}\ P_k^j\ (1 - \theta)\ \frac{\Delta t}{P_k^j}\right)$$

$$LBD = \frac{\ell\ q^{j+½}}{\Delta z_k}\ (1 - \theta)\ NWB_{k+½}\ \frac{\Delta t}{P_k^j}\ sofd_{k+½} -$$

$$\frac{P_{k+½}^{j}\ \varepsilon_{k+½}^{j}\ (E_{\mathrm{lb}}{}_{k+½}^{j+½} + D_{\mathrm{lb}}{}_{k+½}^{j})\ (1 - \theta)\Delta t}{\Delta z_k\ (½\ \Delta z_{k+1} + ½\ \Delta z_k)\ P_k^j}$$

$$ROD = \frac{\ell\ q^{j+½}}{\Delta z_k}\ \theta\ (1 - NWB_{k-½})\ \frac{\Delta t}{P_k^j}\ sofd_{k-½} +$$

$$\frac{P^j_{k-½}\ \varepsilon^j_{k-½}\ (E_{\mathrm{lb}\ k-½}^{\ j+½} + D_{\mathrm{lb}\ k-½}^{\ j})\ \theta\ \Delta t}{\Delta z_k\ (½\ \Delta z_k + ½\ \Delta z_{k-1})\ P^j_k}$$

$$RDD = -\ \frac{\ell\ q^{j+½}}{\Delta z_k}\ \theta\ (1 - NWB_{k+½})\ \frac{\Delta t}{P^j_k}\ sofd_{k+½}\ -$$

$$\frac{P^j_{k+½}\ \varepsilon^j_{k+½}\ (E_{\mathrm{lb}\ k+½}^{\ j+½} + D_{\mathrm{lb}\ k+½}^{\ j})\ \theta\ \Delta t}{\Delta z_k\ (½\ \Delta z_{k+1} + ½\ \Delta z_k)\ P^j_k}\ +$$

$$\frac{\ell\ q^{j+½}}{\Delta z_k}\ \theta\ NWB_{k-½}\ \frac{\Delta t}{P^j_k}\ sofd_{k-½}\ -$$

$$\frac{P^j_{k-½}\ \varepsilon^j_{k-½}\ (E_{\mathrm{lb}\ k-½}^{\ j+½} + D_{\mathrm{lb}\ k-½}^{\ j})\ \theta\ \Delta t}{\Delta z_k\ (½\ \Delta z_k + ½\ \Delta z_{k-1})\ P^j_k}\ +$$

$$\left(\varepsilon^j_k + \rho_{\mathrm{b}}{}^j_k\ K_{\mathrm{F}} \left(\frac{c_{\mathrm{lb}}{}^j_k}{c_{\mathrm{e}}}\right)^{n_{\mathrm{wb}}-1}\right)\left(1 - k_{\mathrm{b}}\ P^j_k\ \theta\ \frac{\Delta t}{P^j_k}\right)$$

$$RBD = -\ \frac{\ell\ q^{j+½}}{\Delta z_k}\ \theta\ NWB_{k+½}\ \frac{\Delta t}{P^j_k}\ sofd_{k+½}\ +$$

$$\frac{P^{j}_{k+½}\ \varepsilon^{j}_{k+½}\ (E_{\mathrm{lb}}\,{}^{j+½}_{k+½} + D_{\mathrm{lb}}\,{}^{j}_{k+½})\ \theta\ \Delta t}{\Delta z_k\ (½\ \Delta z_{k+1} + ½\ \Delta z_k)\ P^{j}_{k}}$$

$RV = 0$, except at the last grid point *ebbt* (see section 6.7).
As was done for the water layer, Eq. (6.39) can be solved for c_{lb}^{j+1} by inverting the left-hand tridiagonal matrix (see Press et al, 1986).

6.7 Boundary conditions and initial condition for sediment

6.7.1 Upper boundary condition

For a positive as well as for a negative advection flow q, $c_{lb} = c_i$ at the location $z = 0$ in the sediment, so the concentration of the substance in the liquid phase of the sediment equals the concentration of the substance dissolved in the water layer in segment number i. The conservation equation for the sediment subsystem, Eq. (3.12), now reads:

$$\left(P\,\frac{\partial c_b^*}{\partial t}\right)_1^j = -\left(\frac{\partial (PJ_{lb})}{\partial z}\right)_1^{j+½} - (k_b\, c_b^*\, P)_1^{j+½}$$

$$\approx \frac{(PJ_{lb})_{½}^{j+½} - (PJ_{lb})_{1½}^{j+½}}{\Delta z_1} - k_b\, c_{b\,1}^{*\,j+½}\, P_1^j \qquad (3.12a)$$

The term $(PJ_{lb})_{½}^{j+½}$ stands for:

$$(PJ_{lb})_{½}^{j+½} = \ell\, q^{j+½}\, c_{lb\,½}^{\;j+½} - P_{½}^j\, \varepsilon_{½}^j\, D_{lb\,½}^{\;j}\, \frac{c_{lb\,1}^{\;j+½} - c_{lb\,½}^{\;j+½}}{½\Delta z_1} \qquad (6.34a)$$

if $q \geq 0$:

$$= \ell\, q^{j+½}\, c_i^{j+½} - P_{½}^j\, \varepsilon_{½}^j\, D_{lb\,½}^{\;j}\, \frac{c_{lb\,1}^{\;j+½} - c_i^{j+½}}{½\Delta z_1}$$

if $q < 0$:

$$= \ell\, q^{j+½}\, c_{lb\,1}^{\;j+½} - P_{½}^j\, \varepsilon_{½}^j\, D_{lb\,½}^{\;j}\, \frac{c_{lb\,1}^{\;j+½} - c_i^{j+½}}{½\Delta z_1}$$

in which the variables are defined by Eq. (6.34). The terms $(PJ_{lb})_{1½}^{j+½}$ and $(k_b.c_b^*.P)_1^{j+½}$ were defined by the Eqs. (6.35) and (6.36). Substitution and rearrangement of the numerical approximations of the conservation equation, Eq. (3.12), for $k = 1$ results in the upper boundary condition (Annex 8).

6.7.2 Lower boundary condition

For a downward advection flow, the boundary condition at the lower end of the sediment subsystem considered is:

$$PJ_{\mathrm{lb}} = \ell\ q\ c_{\mathrm{lb}} - \varepsilon\ P\ (E_{\mathrm{lb}} + D_{\mathrm{lb}}) \frac{\partial c_{\mathrm{lb}}}{\partial z} \qquad (6.40)$$

In the case of an upward advection flow the boundary condition reads:

$$PJ_{\mathrm{lb}} = \ell\ q\ c_{\mathrm{lot}} \qquad (6.41)$$

in which c_{lot} is the concentration in the water flowing upward, i.e. seeping from the neighbouring lot into the ditch bottom. This concentration needs to be prescribed as a function of time. This boundary condition is also discontinuous, which is why an additional number of grid points *ebbt* (end buffer sediment [water bottom], total number) have been defined, constituting a buffer behind the last grid point *n* in the sediment.
The mass flux flowing out of the buffer at grid point *ebbt* equals:

if $q > 0$: $\quad PJ_{ebbt+\frac{1}{2}} = \ell\ q\ c_{\mathrm{lb}}$

(i.e. diffusion and dispersion are neglected), and

if $q < 0$: $\quad PJ_{\mathrm{lb}} = \ell\ q\ c_{\mathrm{lot}} \qquad (6.42)$

As in the case of the water layer, the buffer only serves the purpose of solving the numerical problem caused by the discontinuous boundary condition. Hence, the substance is exclusively transported in the buffer; no substance originates or disappears in the buffer. This implies that certain processes can occur in the buffer, while others cannot:
— advection, dispersion and diffusion of the substance do occur, but only advection occurs in the last half segment;
— no sorption (to the solid phase of the sediment) occurs and
— there can be no transformation of the substance.

Annex 9 presents the elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD* and *RBD* for the grid points *n*+1 up to *ebbt*-1 inclusive.

When the water flow is downward, the lower boundary condition is described by an outgoing advection flux; the diffusion and dispersion in the last half segment are neglected. (This implies that the substance cannot enter by diffusion or dispersion and the mass balance will continue to represent 100% of the mass originally added.) The concentration at the boundary, ${c_{\mathrm{lb}}}^{j}_{ebbt+\frac{1}{2}}$, is determined by:

$$c_{\mathrm{lb}\ ebbt+½}^{\ j} = c_{\mathrm{lb}\ ebbt}^{\ j} \tag{6.43}$$

Annex 10 defines the elements *LOD, LDD, LBD, ROD, RDD* and *RBD* at the grid point *ebbt*.

When the water flow is upward, the concentration in the liquid phase at the lower boundary is externally determined. So:

$$(PJ_{\mathrm{lb}})_{ebbt+½}^{j+½} = \ell\ q^{j+½}\ c_{\mathrm{lb}\ \mathrm{lot}}^{\ j+½}$$

$$= \ell\ q^{j+½}\ \theta\ c_{\mathrm{lb}\ \mathrm{lot}}^{\ j} + \ell\ q^{j+½}\ (1 - \theta)\ c_{\mathrm{lb}\ \mathrm{lot}}^{\ j+1} \tag{6.44}$$

Annex 11 shows the elements *LOD, LDD, LBD, ROD, RDD, RBD* and *RV* at the last grid point *ebbt*. The last element of the right-hand vector *RV* is found to be not equal to zero in this case.

6.7.3 Initial condition

There are two options for the initial condition of the sediment subsystem. In the first option, the sediment is free of pesticide at time $t = 0$. This implies:

$$c_{\mathrm{b}}^{*} = 0 \tag{6.45}$$

In the second option, a certain mass of pesticide is located in the sediment; this can be described as a function of depth. At time $t = 0$:

$$c_{\mathrm{b}}^{*} = \mathrm{f}(z) \tag{6.46}$$

in which

f(z) = mass concentration of the substance in sediment as a function of depth at time $t = 0$ ($\mathrm{M.L^{-3}}$).

Given the initial concentration, and selecting the right upper and lower boundary conditions, the system of numerical equations, Eq. (6.39), can now be solved.

6.8 Iterative calculations caused by the Freundlich equation for sorption

At two points, iterative calculations are necessary to solve the equations, because the non-linear Freundlich equation was used to describe sorption to the suspended solids and to the solid sediment material. The first point concerns the calculation of the initial pesticide concentration in the water phase; the second concerns the solution of the matrix equation for the water layer or the sediment.

The conservation equation for the water layer, Eq. (3.6), reads:

$$\frac{\partial(c^* A)}{\partial t} = - \frac{\partial(AJ)}{\partial x} - k\,(c^* A) + J_{wa}\,O_x - J_{wb}\,P_x \tag{3.6}$$

To approximate the fluxes of the substance in the right-hand term of Eq. (3.6), values of c have to be derived from known values of c^*. Combining Eqs. (4.1) and (4.2) results in:

$$c^* = c + \frac{DW\,P_{z=0}}{A} K_{mp}\,c + ss\,K_{F,ss}\,c_{e,ss} \left(\frac{c}{c_{e,ss}} \right)^{n_{ss}} \tag{6.47}$$

Eq. (6.47) shows that it is impossible to derive values of c from values of c^* in an explicit way. Rearranging Eq. (6.47) yields an implicit equation in c:

$$c = \frac{c^*}{1 + \frac{DW\,P_{z=0}}{A} K_{mp} + ss\,K_{F,ss} \left(\frac{c}{c_{e,ss}} \right)^{n_{ss}-1}} \tag{6.47a}$$

For many pesticides, the value of n_{ss}-1 is small (the value of n in sorption studies for soils is often near 0.9), so the right-hand term changes little with a change in c. Using an initial estimation for c in the denominator (and known values for the other variables of the right-hand term), the final value of c can be rapidly found in an iterative manner. The value of c^* at time $t = 0$ serves as the initial estimation for c, yielding a value for c in the left-hand term. This value is then used in the denominator of the right-hand term, etc. In this way, c can be approximated from known values of c^*.

An analogous calculation leads to c_{lb} from known c_b^* for the sediment subsystem. Combining Eqs. (4.17) and (4.18) yields:

$$c_b^* = \varepsilon\,c_{lb} + \rho_b\,K_{F,wb}\,c_{e,wb} \left(\frac{c_{lb}}{c_{e,wb}} \right)^{n_{wb}} \tag{6.48}$$

Rearranging Eq. (6.48) results in:

$$c_{\mathrm{lb}} = \frac{c_{\mathrm{b}}^{*}}{\varepsilon + \rho_{\mathrm{b}}\, K_{\mathrm{F,wb}} \left(\dfrac{c_{\mathrm{lb}}}{c_{\mathrm{e,wb}}} \right)^{n_{\mathrm{wb}}-1}} \qquad (6.48a)$$

For small values of n_{wb}-1, c_{lb} can be rapidly derived from known values of c_{b}^{*}, by applying iterative calculations from an initial value for c_{b}^{*}.

The numerical solution of the conservation equation for the water subsystem results in Eq. (6.26):

$$\begin{pmatrix} LDD & LBD & O & . & . & . & . \\ LOD & LDD & LBD & O & . & . & . \\ O & LOD & LDD & LBD & O & & \\ . & . & . & . & . & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & . & . & O & LOD & LDD & LBD \\ . & . & . & . & O & LOD & LDD \end{pmatrix} \begin{pmatrix} c_1^{j+1} \\ c_2^{j+1} \\ . \\ . \\ c_{i-1}^{j+1} \\ c_i^{j+1} \\ c_{i+1}^{j+1} \\ . \\ . \\ c_m^{j+1} \end{pmatrix} = \begin{pmatrix} RDD & RBD & O & . & . & . & . \\ ROD & RDD & RBD & O & . & . & . \\ O & ROD & RDD & RBD & O & & \\ . & . & . & . & . & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & . & . & O & ROD & RDD & RBD \\ . & . & . & . & O & ROD & RDD \end{pmatrix} \begin{pmatrix} c_1^{j} \\ c_2^{j} \\ . \\ . \\ c_{i-1}^{j} \\ c_i^{j} \\ c_{i+1}^{j} \\ . \\ . \\ c_m^{j} \end{pmatrix} + \begin{pmatrix} RV_1 \\ RV_2 \\ . \\ . \\ RV_{i-1} \\ RV_i \\ RV_{i+1} \\ . \\ . \\ RV_m \end{pmatrix} \qquad (6.26)$$

Eq. (6.26) has the shape of:

$$A.c^{j+1} = B.c^{j} + d \qquad (6.26a)$$

in which A and B are tridiagonal matrices and d is a vector.
The solution of Eq. (6.26a) is:

$$c^{j+1} = A^{-1}\,(B.c^{j} + d) \qquad (6.26b)$$

As the tridiagonal matrix A contains the terms

$$ss \cdot K_{\mathrm{F},ss} \left(\frac{c_{...}^{j+1}}{c_{\mathrm{e,ss}}} \right)^{n_{ss}-1}$$

which are not yet known at time $t = j$, iterative calculations are needed to solve Eq. (6.26b). The value of c^{j+1} in the denominator of this right-hand term of Eq. (6.26b) can approximated by the value of c^{j}; this leads to a first value of c^{j+1} in the left-hand term of Eq. (6.26b); this value can then be used in the denominator of the right-hand term, etc. The iterations are stopped as soon as the next value of c^{j+1} differs by less than 0.00001% from the previous value. In this way, the system of Eqs. (6.26) can be solved.

For the sediment subsystem, the system of equations approximating the conservation equation reads:

$$\begin{pmatrix} LDD & LBD & O & . & . & . & . \\ LOD & LDD & LBD & O & . & . & . \\ O & LOD & LDD & LBD & O & & \\ . & . & . & . & . & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & . & . & O & LOD & LDD & LBD \\ . & . & . & . & O & LOD & LDD \end{pmatrix} \begin{pmatrix} c_1^{j+1} \\ c_2^{j+1} \\ . \\ . \\ c_{k-1}^{j+1} \\ c_k^{j+1} \\ c_{k-1}^{j+1} \\ . \\ . \\ c_n^{j+1} \end{pmatrix} = \begin{pmatrix} RDD & RBD & O & . & . & . & . \\ ROD & RDD & RBD & O & . & . & . \\ O & ROD & RDD & RBD & O & & \\ . & . & . & . & . & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & . & . & O & ROD & RDD & RBD \\ . & . & . & . & O & ROD & RDD \end{pmatrix} \begin{pmatrix} c_1^{j} \\ c_2^{j} \\ . \\ . \\ c_{k-1}^{j} \\ c_k^{j} \\ c_{k+1}^{j} \\ . \\ . \\ c_n^{j} \end{pmatrix} + \begin{pmatrix} RV_1 \\ RV_2 \\ . \\ . \\ RV_{k-1} \\ RV_k \\ RV_{k+1} \\ . \\ . \\ RV_n \end{pmatrix} \quad (6.39)$$

The left-hand tridiagonal matrix of Eq. (6.39) contains a term

$$\rho_{b\,k}^{\,j}\, K_{F,wb} \left(\frac{c_{lb\,k}^{\,j+1}}{c_{e,wb}} \right)^{n_{wb}-1}$$

in which c_{lb}^{j+1} is not yet known at time $t = j$. With the help of iterative calculations as described for the water subsystem, the system of Eqs. (6.49), approximating the conservation equation for the sediment subsystem, can be solved.

6.9 Coupling the water and sediment subsystems

Concentration varies with distance in the watercourse; this means that the sediment at the beginning of the watercourse is influenced by a different concentration than the sediment located, e.g., halfway along the ditch. This phenomenon is accounted for by defining a sediment subsystem below each grid point in the water layer. Hence, the entire field ditch system of TOXSWA comprises one water subsystem and many sediment subsystems (Fig. 18).

The conservation equation for the water and sediment subsystems are solved separately. They are, however, linked to each other by the exchange term with the sediment in the conservation equation for the water subsystem and by the upper boundary condition in the conservation equation for the sediment subsystem. In the TOXSWA model, the linkage is done in such a way that the conservation equation for the water subsystem is approximated first. Only then is the conservation equation for the sediment subsystem approximated. This means that it is not necessary to solve both conservation equations simultaneously; instead, the simpler option of solving one equation after the other has been chosen. The next two sections explain this solution method in more detail.

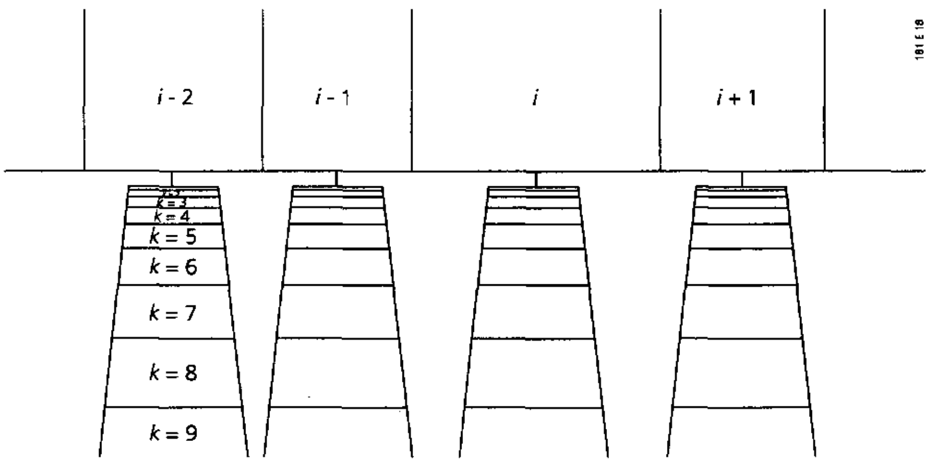


Fig. 18 Detail of the TOXSWA field ditch system, demonstrating the concept of one water subsystem coupled to many sediment subsystems

To solve the conservation equation for the water subsystem, its right-hand terms are evaluated at time $j+½$ in section 6.4 (see Eq. (6.17)). For the exchange term with the sediment, $-(J_{wb}.P_x)_i^{j+½}$, this yields a term $c_{lb}{}_{i,\,k=1}^{j+½}$ in Eq. (6.22). This term is approximated as:

$$c_{lb}{}_{i,\,k=1}^{j+½} = c_{lb}{}_{i,\,k=1}^{j} \tag{6.50}$$

The value of c_{lb} at time $j+½$ is thus approximated by its value at time j. This means that the value $c_{lb}^{j+½}$ need not yet be known when the conservation equation of the water subsystem is being solved. Hence, it is not necessary to solve simultaneously the conservation equations of the water and sediment subsystems.

Approximating the upper boundary condition of the sediment subsystem yields the terms $c_i^{j+½}$ and $c_{lb,1}^{j+½}$ in section 6.7.1 (see Eq. (6.34a)). Hence, the terms c and c_{lb} at times j and $j+1$ appear in the approximation of the upper boundary condition (see also Annex 8). Calculating c_{lb}^{j+1} at time j in the sediment subsystem requires that the value of c_i^{j+1} is known. This would imply that both conservation equations, that for the water and that for the sediment subsystem, need to be solved simultaneously. However, this is a complex procedure. It is simpler to have the solution of the conservation equation of the water subsystem precede that of the sediment subsystem by one time step. This implies that at time j in the sediment subsystem, the concentration c_i in the water layer is known at time j as well as at time $j+1$. The system of equations for the sediment subsystem, Eqs. (6.39), can now easily be solved.

7 Verification

7.1 Introduction

Annexes 12 and 13 show an outline of the computer program of the TOXSWA model. Annex 19 presents the source code of the TOXSWA program, version 1.0, including a guide to the vocabulary used. After a computer program has been written a verification is required. Verification is defined as the examination of the numerical technique in the computer model to ascertain that it truly represents the mathematical model and that there are no inherent numerical problems in obtaining a solution. This also implies a check on errors in the code (programming bugs). (Leaching Modelling Workgroup FOCUS, 1995.) For finite-difference approximations, the concept of verification may be worked out using the three notions of convergence, stability and consistency (Bear and Verruijt, 1987; Lapidus and Pinder, 1982). The condition of convergence states that when the finite-difference grid is refined the truncation errors go to zero. Stability concerns the unstable growth or stable decay of errors in the arithmetic operations needed to solve the finite-difference equations, so it has to do with the boundedness of all perturbations in a computed solution. Consistency is the requirement that when the finite-difference grid is refined, the truncation errors go to zero, but moreover that the finite-difference model approximates the partial differential equation desired and not some other partial differential equation.

The three notions of convergence, stability and consistency lead to the formulation of the following conditions for verification.
Convergence is worked out as follows:

1.[9] A solution obtained with the same numerical solution scheme but time and space steps e.g. 10 times smaller, should differ only slightly from the solution obtained with the original time and space steps. This implies that in this case the truncation errors are so small that the numerical equations provide nearly the same solution.

2.[10] The solution obtained should not vary with the combinations of the numerical weight factors selected for space β and time θ; this means that the solution should be identical whether it has been calculated with the aid of a forward, backward or central difference scheme, or with the aid of an implicit or explicit difference scheme. (The numerical dispersion in TOXSWA, which depends on

[9] This condition is not yet tested for version 1.0 of the TOXSWA model.

[10] These conditions have not been tested for version 1.0 of the TOXSWA model. Version 1.0 allows only the explicit central difference method to be selected to solve the conservation equations, and calculations are performed for a constant, rather than variable, wetted area *A*.

the numerical solution scheme selected, has been estimated and the physical dispersion has been corrected for this.)

The stability condition has been translated into the following positivity conditions:

3. The solution of the matrix equations, Eq. (6.26) for the water layer and Eq. (6.39) for the sediment, needs to be stable, i.e. the errors in the arithmatic operations needed to solve the finite-difference equations should be bounded. Moreover, as it concerns concentrations, the solution should be positive. It can be proven that for specific conditions for the *LOD, LDD, LBD, ROD, RDD* and *RBD* elements of the matrix equations and due to the specific structure of these elements (described in Annex 14 at the illustration for the three-dimensional case), the solution will be positive. Positivity can be shown to be a stricter condition than the stability condition for the solution.

The following four checks are proposed for the consistency condition:

4. Letting the finite differences approach zero should result in the numerical equations reducing to the original partial differential equations.

The numerical equations in the computer program should truly represent the mathematical model, i.e. the mass balances. This means that the consistency condition also implies the following:

5. The mass of pesticide introduced (100%) should be traceable at every moment with an error of less than 0.1% (this means that the mass balances tally and that no pesticide is lost during the calculation process by any other than those being modelled, e.g. transformation or volatilisation).

Furthermore consistency is usually checked by verifying the numerical procedure against a variety of analytical solutions (Bear and Verruijt, 1987). Two situations are proposed for this check:

6. An analytical solution of the advection-dispersion equation for constant h and A for a system with transformation, volatilisation and convective/dispersive downward seepage responding to a Dirac delta function-type input, as described in Jury and Roth (1990). (Volatilisation and convective/dispersive downward seepage are, from a mathematical point of view, equivalent to transformation.) This solution should correspond with that obtained with the numerical solution scheme. The same type of analytical solution can also be applied to the sediment subsystem. So, the water and sediment subsystem can be checked independently.

7.[11] A situation like that described under point 6., but now with a wetted surface A varying according to: $A = \varphi . e^{\gamma x + \xi t}$. This situation is relevant because the TOXSWA model can deal with water levels and wetted perimeters varying in time and space. This special description for A again results in a convection-dispersion type of equation, but the parameters of the convection and transformation terms include γ and/or ξ.

7.2 Stability condition

7.2.1 Description of the positivity for the water layer

The stability condition was worked out as stated in criterium 3 of section 7.1.

5. The solution of the matrix equations, Eq. (6.26) for the water layer and Eq. (6.39) for the sediment, needs to be stable, i.e. the errors in the arithmatic operations needed to solve the finite-difference equations should be bounded. Moreover, as it concerns concentrations, the solution should be positive. It can be proven that for specific conditions for the *LOD, LDD, LBD, ROD, RDD* and *RBD* elements of the matrix equations and due to the specific structure of these elements (described in Annex 14 at the illustration for the three-dimensional case), the solution will be positive. Positivity can be shown to be a stricter condition than the stability condition for the solution.

The solution of the system of equations, solving the conservation equation for the water subsystem reads:

$$c^{j+1} = A^{-1} (B.c^{j} + d) \tag{6.26a}$$

The vector d is zero or positive and this implies that c^{j+1} is positive if the product of the inversed matrix A^{-1} and the matrix B is positive, which is true when A^{-1} and B are both positive matrices. A positive matrix is defined as a matrix that has no negative elements. So the tridiagonal matrix B is positive if:
$RBD \geq 0$,
$RDD > 0$ and
$ROD \geq 0$.

The matrix A^{-1} is positive if:
$LBD \leq 0$,
$LDD > 0$ and
$LOD \leq 0$.

[11] These conditions have not been tested for version 1.0 of the TOXSWA model. Version 1.0 allows only the explicit central difference method to be selected to solve the conservation equations and calculations are performed for a constant, rather than variable, wetted area A.

Annex 14 gives the outline of the proof that $A^{-1}.B$ is positive and stable for the two tridiagonal matrices A and B when the elements *RBD, RDD, ROD, LBD, LDD* and *LOD* fulfill the conditions mentioned above. A positive matrix is called stable if the dominant eigenvalue of the matrix is smaller than 1. The outline is given for a system with constant segment size Δx, flow velocity u and water depth h and a linear sorption isotherm describing sorption to suspended solids. For this system the elements *LOD, LDD, LBD, ROD, RDD* and *RBD* are exactly defined in section 7.2.2.

For the sediment subsystem the positivity and its proof may be demonstrated analogously.

7.2.2 Restrictions resulting from requirements for positivity for the water layer

The conditions for which the solution of the matrix equation Eq.(6.26) is positive for the water subsystem have been worked out in an advice about the possible time and space steps for which this equation can be solved. The advice is based upon a slightly simplified situation.

When sorption to suspended solids is described with a linear isotherm and when the segment size Δx, the flow velocity u, the water depth h and wetted area A are constant, the elements *LOD, LDD, LBD, ROD, RDD, RBD* and *RV* for the water subsystem read:

$$LOD = -(1-\theta)\left(1-\beta\right)\frac{u\,\Delta t}{\Delta x}\left(1+ss\,K_{L,ss}\right) - (1-\theta)\frac{E\,\Delta t}{\Delta x^2}\left(1+ss\,K_{L,ss}\right)$$

$$LDD = (1-\theta)\,(1-2\beta)\frac{u\,\Delta t}{\Delta x}\left(1+ss\,K_{L,ss}\right) + 2(1-\theta)\frac{E\,\Delta t}{\Delta x^2}\left(1+ss\,K_{L,ss}\right)$$

$$+\left(1+k\,(1-\theta)\,\Delta t\right) + \left(1+\frac{DW\cdot P_{z=0}}{A}\,K_{mp}+ss\,K_{L,ss}\right)$$

$$+\,k_{t,1}\frac{O_x}{A}(1-\theta)\,\Delta t + \frac{\ell}{A\,P_{z=0}}\,q\,P_x(1-\theta)\,\Delta t + \frac{2\varepsilon\,D_{lb}\,P_x}{A\,\Delta z_1}(1-\theta)\,\Delta t$$

$$LBD = (1-\theta)\ \beta\ \frac{u\ \Delta t}{\Delta x}\left(1 + ss\ K_{\mathrm{L,ss}}\right) - (1-\theta)\ \frac{E\ \Delta t}{\Delta x^2}\ \left(1 + ss\ K_{\mathrm{L,ss}}\right)$$

$$ROD = \theta\ (1-\beta)\ \frac{u\ \Delta t}{\Delta x}\left(1 + ss\ K_{\mathrm{L,ss}}\right) + \theta\ \frac{E\ \Delta t}{\Delta x^2}\ \left(1 + ss\ K_{\mathrm{L,ss}}\right)$$

$$RDD = -\ \theta\ (1-2\beta)\ \frac{u\ \Delta t}{\Delta x}\left(1 + ss\ K_{\mathrm{L,ss}}\right) - 2\ \theta\ \frac{E\ \Delta t}{\Delta x^2}\ \left(1 + ss\ K_{\mathrm{L,ss}}\right)$$

$$+ \left(1 - k\ \theta\ \Delta t\right) + \left(1 + \frac{DW \cdot P_{z=0}}{A}\ K_{\mathrm{mp}} + ss\ K_{\mathrm{L,ss}}\right)$$

$$- \left(k_{\mathrm{t,1}}\ \frac{O_x}{A}\ \theta\ \Delta t + \frac{\ell}{A\ P_{z=0}}\ q\ P_x\ \theta\ \Delta t + \frac{2\varepsilon\ D_{\mathrm{lb}}\ P_x}{A\ \Delta z_1}\ \theta\ \Delta t\right)$$

$$RBD = -\ \theta\ \beta\ \frac{u\ \Delta t}{\Delta x}\left(1 + ss\ K_{\mathrm{L,ss}}\right) + \theta\ \frac{E\ \Delta t}{\Delta x^2}\ \left(1 + ss\ K_{\mathrm{L,ss}}\right)$$

$$RV = k_{\mathrm{t,1}}\ \frac{O_x}{A}\ \frac{c_{\mathrm{a}}}{K_{\mathrm{H}}}\ \Delta t + \frac{2\varepsilon\ D_{\mathrm{lb}}\ P_x}{A\ \Delta z_1}\ c_{\mathrm{lb},\,k=1}\ \Delta t \qquad (7.1)$$

with

$K_{\mathrm{L,ss}}$ = slope of (linear) sorption isotherm of suspended solids ($\mathrm{L^3.M^{-1}}$).

With

$$M = 1 + \frac{DW \cdot P_{z=0}}{A}\ K_{\mathrm{mp}} + ss\ K_{\mathrm{L,ss}} \qquad (M > 0)$$

$$F = k_{t,1} \frac{O_x}{A} + \frac{\ell}{A \, P_{z=0}} q \, P_x + \frac{2\varepsilon \, D_{lb} \, P_x}{A \, \Delta z_1} \qquad \text{(Assumption[12]: } F > 0)$$

$$G^{13} = (1 - 2\beta) \frac{u}{\Delta x} \left(1 + ss \, K_{L,ss}\right) + 2 \frac{E}{\Delta x^2} \left(1 + ss \, K_{L,ss}\right)$$

$$+ k \, M + F$$

the element *LDD* may be rewritten as:

$$LDD = G \, (1 - \theta) \, \Delta t + M \tag{7.2}$$

and the element *RDD* may be rewritten as:

$$RDD = - G \, \theta \, \Delta t + M \tag{7.3}$$

The positivity conditions for the off-diagonal elements of the two tridiagonal matrices lead to an advice about the size of the space step, Δx.

The condition $LBD \leq 0$ (or $RBD \geq 0$) leads to:

$$\Delta x \leq \frac{E}{\beta \, u} \quad \text{for } u > 0, \, E > 0 \text{ ánd } \beta \neq 0 \tag{7.4}$$

The condition $LOD \leq 0$ (or $ROD \geq 0$) leads to:

$$\Delta x \leq - \frac{E}{(1 - \beta) \, u} \quad \text{for } u < 0, \, E > 0 \text{ ánd } \beta \neq 1 \tag{7.5}$$

---

[12] In fact the term F should be such that 2 $E/\Delta x^2 \, (1+ss \, K_{L,ss})+k \, M+F$ is positive; F can become negative if q, the seepage, is negative, but this only happens for unrealistic high (negative) values of q ($|q|$ in the ordre of $m.d^{-1}$). It has been assumed that the dispersion E is positive. It may be possible, however, that the E_{cal} with which TOXSWA 1.0 calculates is negative. In that case an additional condition for Δx and Δt should be added.

[13] The sign of G depends on the values of u and β.

If $u = 0$ the conditions described above do not lead to restrictions for Δx. When a positive flow direction is combined with $\beta = 0$ or when a negative flow direction is combined with $\beta = 1$, there is also no restriction for Δx.

The positivity conditions for the diagonal elements of the two tridiagonal matrices lead to an advice about the size of the time step, Δt.

The condition $LDD > 0$ leads to:

$$G\,(1 - \theta)\,\Delta t + M > 0 \tag{7.6}$$

corresponding to:

1. $$G > 0 \;:\; \Delta t > -\frac{M}{G\,(1 - \theta)} \tag{7.7}$$

   In this case no restriction for the selection of (an always positive) Δt is found.

2. $$G < 0 \;:\; \Delta t < -\frac{M}{G\,(1 - \theta)} \tag{7.8}$$

   and $G < 0$ corresponds to:

   (i) $$\beta > ½ \;\; \text{ánd} \;\; u > H > 0 \tag{7.9}$$

   or

   (ii) $$\beta < ½ \;\; \text{ánd} \;\; u < H < 0 \tag{7.10}$$

   with

$$H = \frac{-\left\{2E\,(1 + ss\,K_{L,ss}) + k\,M\,\Delta x^2 + F\,\Delta x^2\right\}}{(1 - 2\beta)\,(1 + ss\,K_{L,ss})\,\Delta x}$$

If $\beta = ½$ or $u = 0$ $G\,(1 - \theta)$ is always positive, so the condition $LDD > 0$ does not lead to a restriction for Δt. If $\theta = 1$ $G\,(1 - \theta)$ equals zero, so the condition $LDD > 0$ is always true and only the condition $RDD > 0$ poses restrictions to Δt.

The condition $RDD > 0$ leads to:

$$-G\,\theta\,\Delta t + M > 0 \tag{7.11}$$

corresponding to:

1. $$G < 0 \ : \ \Delta t > \frac{M}{G\,\theta} \qquad (7.12)$$

In this case no restriction for the selection of Δt is found.

2. $$G > 0 \ : \ \Delta t < \frac{M}{G\,\theta} \qquad (7.13)$$

and $G > 0$ correponds to:

(i) $$\beta < ½ \ \text{ánd}\ H < u \qquad (7.14)$$

H being negative

(ii) $$\beta > ½ \ \text{ánd}\ H > u \qquad (7.15)$$

H being positive

with H as defined before.

When $\beta = ½$ or $u = 0$ $\;G\,\theta$ is always positive, so the condition RDD does lead to a restriction for Δt. Only when $\theta = 0$ $\;G\,\theta$ equals zero, the condition $RDD > 0$ is alway true but now the condition $LDD > 0$ may lead to restriction for Δt.

Figure 19 summarizes the results for the restrictions imposed on the time step Δt.

It appears that in the case $\theta = 0$ and $|H| > |u|$ (i.e. $-|H| < u < |H|$) the condition $LDD > 0$ is not fulfilled, so the solution to matrix Eq.(6.26) is not positive. So, the matrix Eq. (6.26) has only a positive solution if Δx is selected such that $|H| < |u|$.

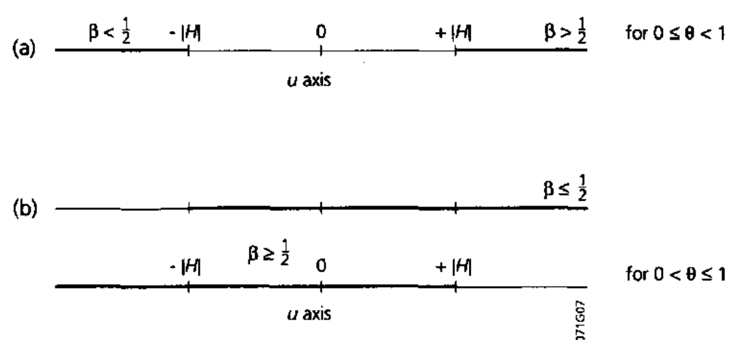


Fig. 19 Outline of the domain (u, β) for which the condition LDD > 0 (part(a)) or RDD > 0 (part(b)) result in a restriction for the time step, Δt

7.2.3 Restrictions resulting from requirements for positivity for the sediment

For the sediment subsystem an advice about the possible time and space steps is also worked out. The solution of matrix Eq.(6.39) is positive for these time and space steps.

When sorption to the bottom material is described with a linear isotherm and when the segment size Δz and the seepage q are constant, the elements *LOD, LDD, LBD, ROD, RDD* and *RBD* for the sediment subsystem read:

$$LOD = -\frac{\ell\, q}{\Delta z}\,(1-\theta)\,(1-\beta)\,\frac{\Delta t}{P_k}\,sofd$$

$$-\frac{P_{k-½}\,\varepsilon_{k-½}\,(E_{\mathrm{lb}\,k-½}+D_{\mathrm{lb}\,k-½})\,(1-\theta)\Delta t}{\Delta z^2\,P_k}$$

$$LDD = \frac{\ell\, q}{\Delta z}\,(1-\theta)\,(1-2\beta)\,\frac{\Delta t}{P_k}\,sofd$$

$$+\frac{P_{k+½}\,\varepsilon_{k+½}\,(E_{\mathrm{lb}\,k+½}+D_{\mathrm{lb}\,k+½})\,(1-\theta)\Delta t}{\Delta z^2\,P_k}$$

$$+\frac{P_{k-½}\,\varepsilon_{k-½}\,(E_{\mathrm{lb}\,k-½}+D_{\mathrm{lb}\,k-½})\,(1-\theta)\Delta t}{\Delta z^2\,P_k}$$

$$+\left(\varepsilon_k+\rho_{\mathrm{b},k}\,K_{\mathrm{L,b}}\right)\left(1+k_{\mathrm{b}}\,(1-\theta)\,\Delta t\right)$$

$$LBD = \frac{\ell\, q}{\Delta z}\,(1-\theta)\,\beta\,\frac{\Delta t}{P_k}\,sofd$$

$$- \frac{P_{k+½}\ \varepsilon_{k+½}\ (E_{\mathrm{lb}\ k+½} + D_{\mathrm{lb}\ k+½})\ (1 - \theta)\Delta t}{\Delta z^2\ P_k}$$

$$ROD = \frac{\ell\ q}{\Delta z}\ \theta\ (1 - \beta)\ \frac{\Delta t}{P_k}\ sofd + \frac{P_{k-½}\ \varepsilon_{k-½}\ (E_{\mathrm{lb}\ k-½} + D_{\mathrm{lb}\ k-½})\ \theta\ \Delta t}{\Delta z^2\ P_k}$$

$$RDD = - \frac{\ell\ q}{\Delta z}\ \theta\ (1 - 2\beta)\ \frac{\Delta t}{P_k}\ sofd - \frac{P_{k+½}\ \varepsilon_{k+½}\ (E_{\mathrm{lb}\ k+½} + D_{\mathrm{lb}\ k+½})\ \theta\ \Delta t}{\Delta z^2\ P_k}$$

$$- \frac{P_{k-½}\ \varepsilon_{k-½}\ (E_{\mathrm{lb}\ k-½} + D_{\mathrm{lb}\ k-½})\ \theta\ \Delta t}{\Delta z^2\ P_k} + \left(\varepsilon_k + \rho_{\mathrm{b},\ k}\ K_{\mathrm{L,b}}\right)\left(1 - k_{\mathrm{b}}\ \theta\ \Delta t\right)$$

$$RBD = - \frac{\ell\ q}{\Delta z}\ \theta\ \beta\ \frac{\Delta t}{P_k}\ sofd + \frac{P_{k+½}\ \varepsilon_{k+½}\ (E_{\mathrm{lb}\ k+½} + D_{\mathrm{lb}\ k+½})\ \theta\ \Delta t}{\Delta z^2\ P_k} \qquad (7.16)$$

with

$K_{\mathrm{L,b}}$ = slope of (linear) sorption isotherm of bottom material ($L^3.M^{-1}$)

sofd = factor equalling 0 or 1, preventing that the dispersion flux would exceed the advection flux in the sediment (see section 4.2)

With

$$V = \varepsilon_k + \rho_{\mathrm{b},\ k}\ K_{\mathrm{L,b}} \qquad (V > 0)$$

and

$$W^{14} = \frac{\ell\ q}{\Delta z}\ (1 - 2\beta)\ \frac{sofd}{P_k} + \frac{P_{k+½}\ \varepsilon_{k+½}\ (E_{\mathrm{lb}\ k+½} + D_{\mathrm{lb}\ k+½})}{\Delta z^2\ P_k}$$

[14] The sign of W depends on the values of q and β.

$$+ \frac{P_{k-½}\ \varepsilon_{k-½}\ (E_{\mathrm{lb}\ k-½} + D_{\mathrm{lb}\ k-½})}{\Delta z^2\ P_k} + V\ k_b$$

the element *LDD* may be rewritten as:

$$LDD = W\ (1 - \theta)\ \Delta t + V \tag{7.17}$$

and the element *RDD* may be rewritten as:

$$RDD = -\ W\ \theta\ \Delta t + V \tag{7.18}$$

The positivity conditions for the off-diagonal elements of the two tridiagonal matrices lead to an advice about the size of the space steps, Δz.

The condition $LBD \leq 0$ (or $RBD \geq 0$) leads to:

$$\Delta z \leq \frac{P_{k+½}\ \varepsilon_{k+½}\ (E_{\mathrm{lb},\ k+½} + D_{\mathrm{lb},\ k+½})}{\ell\ q\ \beta\ sofd} \tag{7.19}$$

for $q > 0$, $E + D > 0$ ánd $\beta \neq 0$ ánd $sofd \neq 0$.

The condition $LOD \leq 0$ (or $ROD \geq 0$) leads to:

$$\Delta z \leq -\ \frac{P_{k-½}\ \varepsilon_{k-½}\ (E_{\mathrm{lb},\ k-½} + D_{\mathrm{lb},\ k-½})}{\ell\ q\ (1 - \beta)\ sofd} \tag{7.20}$$

for $q < 0$, $E + D > 0$ ánd $\beta \neq 1$ ánd $sofd \neq 0$.

If $q = 0$ the conditions described above do not lead to restrictions for Δz. When a positive (downward) seepage is combined with $\beta = 0$ or when a negative (upward) seepage is combined with $\beta = 1$, there is also no restriction for Δz.

The positivity conditions for the diagonal elements of the two tridiagonal matrices lead to an advice about the size of the time step, Δt. Generally speaking, the water subsystem is more dynamic than the sediment subsystem, so one may expect that the restriction for Δt will be stricter for the water than for the sediment subsystem. In the case of a pond, however, with zero flow, the situation may be different.

The condition $LDD > 0$ leads to:

$$W\,(1-\theta)\,\Delta t + V > 0 \tag{7.21}$$

corresponding to:

1. $$W > 0 \;:\; \Delta t > \frac{-V}{W\,(1-\theta)} \tag{7.22}$$

   In this case no restriction for the selection of Δt is found.

2. $$W < 0 \;:\; \Delta t < \frac{-V}{W\,(1-\theta)} \tag{7.23}$$

   and $W < 0$ corresponds to:

   (i) $$\beta > ½ \;\; \text{ánd} \;\; Z < q \tag{7.24}$$

   Z being positive
   or

   (ii) $$\beta < ½ \;\; \text{ánd} \;\; Z > q \tag{7.25}$$

   Z being negative
   with

$$Z = \frac{-\left\{P_{k+½}\,\varepsilon_{k+½}\,(E_{\mathrm{lb},\,k+½} + D_{\mathrm{lb},\,k+½}) + P_{k-½}\,\varepsilon_{k-½}\,(E_{\mathrm{lb},\,k-½} + D_{\mathrm{lb},\,k-½}) + V\,k_b\,\Delta z^2\,P_k\right\}}{\ell\,(1-2\beta)\;sofd\;\Delta z}$$

When $\beta = ½$ or $q = 0$ $W\,(1\text{-}\theta)$ is always positive, so the condition $LDD > 0$ does not lead to a restriction for Δt. When $\theta = 1$ $W\,(1\text{-}\theta)$ equals zero, so the condition $LDD > 0$ is always true and the condition $RDD > 0$ may impose restrictions.

The condition $RDD > 0$ leads to:

$$-\,W\,\theta\,\Delta t + V > 0 \tag{7.26}$$

corresponding to:

1. $$W < 0 \;:\; \Delta t > \frac{V}{W\,\theta} \tag{7.27}$$

   In this case no restriction for the selection of Δt is found.

2. $$W > 0 \ : \ \Delta t < \frac{V}{W\,\theta} \tag{7.28}$$

and $W > 0$ corresponds to:

(i) $$\beta > ½ \text{ ánd } Z > q \tag{7.29}$$

Z being positive
or

(ii) $$\beta < ½ \text{ ánd } Z < q \tag{7.30}$$

Z being negative.

When $\beta = ½$ or $q = 0$ $W\ \theta$ is always positive so the condition $RDD > 0$ does lead to a restriction for Δt. When $\theta = 0$, $W.\theta$ equals zero, so the condition $RDD > 0$ is always true but now the condition $LDD > 0$ may impose a restriction for Δt.

Figure 20 summarizes the results for the restrictions imposed on the time step, Δt. It appears that when $\theta = 0$ and $|Z| > |q|$ (i.e. $-|Z| < q < |Z|$) the condition $LDD > 0$ is not fulfilled, so the solution to matrix Eq.(6.39) is not postive. So, the matrix Eq. (6.39) has only a positive solution if Δz is selected such that $|Z| < |q|$.

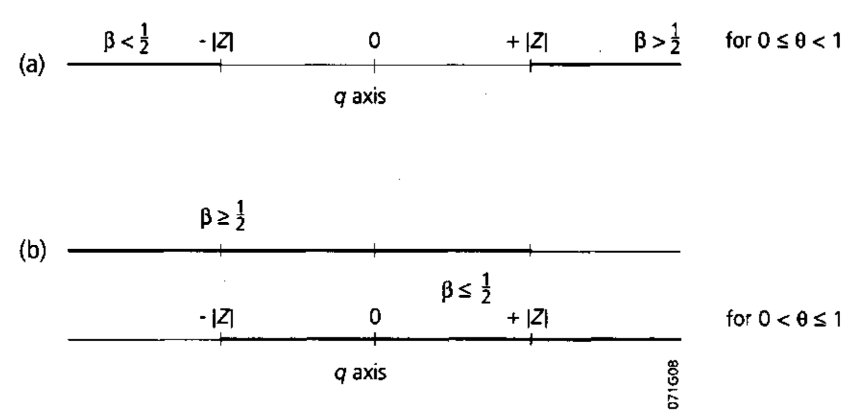


Fig. 20 Outline of the domain (q, β) for which the condition LDD > 0 (part (a)) or RDD > 0 (part(b)) result in a restriction for the time step, Δt

7.3 Consistency condition

7.3.1 Introduction

The consistency condition was tested according to three of the criteria formulated in section 7.1.

4. Letting the finite differences approach zero should result in the numerical equations reducing to the original partial differential equations.

Inspection of the approximations of the right-hand and left-hand terms of the conservation equations for the water and sediment subsystems makes it clear that all procedures are indeed consistent. (See Eqs. (6.17) and (6.23) for the water subsystem and Eqs. (6.33) and (6.37) for the sediment subsystem.)

5. The mass of pesticide introduced (100%) should be traceable at every moment with an error less than 0.1% (this means that the mass balances tally and that no pesticide is lost during the calculation process by any other process than those being modelled, e.g. transformation or volatilisation).

To check that no mass is lost in the calculation process, separate mass balances are drawn up for the water subsystem and each sediment subsystem. (Each subsystem is treated as a whole entity, meaning that it is not subdivided when the mass balances are calculated.) For each subsystem all incoming and outgoing fluxes, including the transformed pesticide mass, are summated in time and the quantity missing from the mass balance is calculated. The TOXSWA program monitors whether this missing mass does not exceed a certain percentage of the initial total mass present plus incoming mass.[15]

The numerical solution of the mass conservation equation is verified against an analytical solution, described below.

6. An analytical solution of the advection-dispersion equation for constant h and A for a system with transformation, volatilisation and convective/dispersive downward seepage responding to a Dirac delta function-type input, as described in Jury and Roth (1990). (Volatilisation and convective/dispersive downward seepage are, from a mathematical point of view, equivalent to transformation.) This solution should correspond with that obtained with the numerical solution scheme. The same type of analytical solution can also be applied to the sediment subsystem. Hence, the water and sediment subsystems can be checked independently.

The analytical solutions are worked out in the next sections.

---

[15] In the TOXSWA model version 1.0 error messages are written to a message file when the missing mass exceeds 0.1%. (There is a maximum of 20 error messages about mass balances per subsystem, water layer and sediment.)

7.3.2 Analytical solutions of the Convection-Dispersion Equation

In Jury and Roth (1990) transfer functions are used to estimate outflow concentrations from soil columns. Transfer functions are used to model such a complex system in a simple way by characterizing the output flux as a function of the input flux. Outflow concentrations may be described with the aid of a travel time probability distribution or probability density function (pdf) for outflowing solute molecules. The travel time pdf characterizes the distribution of possible travel times that a solute molecule might experience in moving from the inlet end to the outlet end. Example 3.1 of Jury and Roth provides the solution of the Convection-Dispersion Equation in case of the entry of a narrow pulse (delta function) of solute flux concentration through the inlet end $z = 0$ at $t = 0$, assuming that there is no solute present in the system initially (Eq. 3.12, divided by θV !, compare the solution of Problem 3.8 in Jury and Roth (1990).).

$$c_t^r(z, t) = \frac{1}{\sqrt{\pi\, D\, t}} \exp\left(-\frac{(z - Vt)^2}{4\, D\, t}\right) - \frac{V}{2D} \exp\left(\frac{Vz}{D}\right) \operatorname{erfc}\left(\frac{z + Vt}{\sqrt{4\, D\, t}}\right) \quad (7.31)$$

in which

c_t^r = probability density of the total resident concentration (here 1, normalised, L^{-1})

D = effective diffusion-dispersion coefficient ($L^2.T^{-1}$)

t = time (T)

z = depth (L)

V = pore water velocity ($L.T^{-1}$)

and where

$\operatorname{erfc}(x) = 1 - \operatorname{erf}(x)$, the complementary error function, defined by

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-y^2)\, dy \quad (7.32)$$

This applies to non-sorbing, non-decaying substances and normalised situations (so the integral of the total concentration with depth equals always 1.).

For substances undergoing simultaneous linear adsorption and first order decay during transport under steady state water flow conditions, and for an application of mass M the solution reads (Boesten, 1992, combining sections 4.2 and 4.8 of Jury and Roth (1990)):

$$c_l^r = \frac{c_t^r}{\theta R} = \frac{M}{\theta R} \cdot e^{-\mu t} \left\{ \frac{1}{\sqrt{\pi D \frac{t}{R}}} \exp\left(- \frac{\left(z - V \frac{t}{R}\right)^2}{4 D \frac{t}{R}} \right) - \frac{V}{2D} \exp\left(\frac{Vz}{D}\right) \cdot \operatorname{erfc}\left(\frac{z + V \frac{t}{R}}{\sqrt{4 D \frac{t}{R}}} \right) \right\} \tag{7.33}$$

in which

c_l^r = mass of dissolved solute per volume of fluid ($M.L^{-3}$)
c_t^r = total resident concentration ($M.L^{-3}$)
M = applied area-averaged mass ($M.L^{-2}$)
θ = volume fraction of water (1)
R = retardation factor (1)
μ = first-order transformation rate constant (T^{-1})

So, Eq.(7.33) provides the analytical solution for the generalised form of the Convection-Dispersion Equation in case of the entry of a narrow pulse (delta function) of solute flux concentration through the inlet end $z = 0$ at $t = 0$, assuming that there is no solute present in the system initially and for a substance undergoing simultaneous linear sorption and first-order decay during transport with a steady state water flow. The generalised form of the Convection-Dispersion Equation reads:

$$(\theta + \rho_b K_{L,b}) \frac{\partial c_l^r}{\partial t} = - \theta V \frac{\partial c_l^r}{\partial x} + \theta D \frac{\partial^2 c_l^r}{\partial x^2} - \mu (\theta + \rho_b K_{L,b}) c_l^r$$

$$\Leftrightarrow R \frac{\partial c_l^r}{\partial t} = - V \frac{\partial c_l^r}{\partial x} + D \frac{\partial^2 c_l^r}{\partial x^2} - \mu R c_l^r \tag{7.34}$$

with

$K_{L,b}$ = slope of (linear) sorption isotherm of bottom material ($L^3.M^{-1}$)

and

$$R = 1 + \frac{\rho_b K_{L,b}}{\theta} \qquad (1)$$

where R is called the retardation factor (Bolt, 1979). The retardation factor R is defined as the ratio of the mean flow velocity of a non-sorbing substance divided by the mean flow velocity of a sorbing substance in a system with exclusively convective transport.

For the water subsystem the more simple analytical solution of the Convection-Dispersion Equation in case of an instantaneous source, e.g. a sudden drain discharge has been chosen. This solution applies for an infinite and not a semi-finite system as described here above. This solution is given in Leistra (1973) and reads for a soil system (in case of decay combined with adsorption):

$$c_l^r = \frac{M}{2\theta\, R} \frac{1}{\sqrt{\pi\, D\, \frac{t}{R}}} e^{-\mu t} \exp\left(- \frac{(x - V\frac{t}{R})^2}{4\, D\, \frac{t}{R}} \right) \tag{7.35}$$

7.3.3 Application of the analytical solutions of the Convection-Dispersion Equation to the water and sediment subsystems of TOXSWA

We now apply the solutions described above to the two subsystems of the TOXSWA model.

The conservation equation for the sediment, Eq. (3.12) reads:

$$P \frac{\partial c_b^*}{\partial t} = - \frac{\partial (P J_{lb})}{\partial z} - k_b c_b^* P \tag{3.12}$$

Assuming sorption to sediment is a linear process, so

$$X_b = K_{L,b} \cdot c_{lb} \tag{7.36}$$

with

$K_{L,b}$ = slope of (linear) sorption isotherm of bottom material ($L^3.M^{-1}$)

and using Eq. (4.17):

$$c_b^* = \varepsilon c_{lb} + \rho_b K_{L,b} c_{lb} \tag{7.37}$$

for constant perimeters P (so for a rectangular-shaped cross section), and substituting Eq. (4.21), Eq. (3.12) now reads:

$$\left(\varepsilon + \rho_b K_{L,b}\right) \frac{\partial c_{lb}}{\partial t} = - \frac{\ell q}{P} \frac{\partial c_{lb}}{\partial z} + \varepsilon\, (E_{lb} + D_{lb}) \frac{\partial^2 c_{lb}}{\partial z^2} - k \left(\varepsilon + \rho_b K_{L,b}\right) c_{lb}$$

$$\Leftrightarrow R_{wb} \frac{\partial c_{lb}}{\partial t} = - w \frac{\partial c_{lb}}{\partial z} + (E_{lb} + D_{lb}) \frac{\partial^2 c_{lb}}{\partial z^2} - k R_{wb} c_{lb} \tag{7.38}$$

in which
R_{wb} = retardation factor for the sediment subsystem (1)

$$R_{wb} = 1 + \frac{\rho_b K_{L,b}}{\varepsilon}$$

This is the generalised form of the Convection-Dispersion Equation model for a substance undergoing simultaneous transport, linear equilibrium sorption and first order decay. For a semi-infinite system its solution for a pulse input at $z = 0$ and $t = 0$ reads (in the notation used for the TOXSWA model description) (compare Eq. (7.33)):

$$c_b^* = M e^{-kt} \left\{ \frac{1}{\sqrt{\pi (E_{lb} + D_{lb}) \frac{t}{R_{wb}}}} \exp \left(- \frac{\left(z - w \frac{t}{R_{wb}} \right)^2}{4 (E_{lb} + D_{lb}) \frac{t}{R_{wb}}} \right) \right.$$

$$\left. - \frac{w}{2(E_{lb} + D_{lb})} \exp \left(\frac{wz}{(E_{lb} + D_{lb})} \right) \operatorname{erfc} \left(\frac{z + w \frac{t}{R}}{\sqrt{4 (E_{lb} + D_{lb}) \frac{t}{R}}} \right) \right\} \tag{7.39}$$

The conservation equation for the water layer, Eq.(3.6) reads:

$$\frac{\partial (c^* A)}{\partial t} = - \frac{\partial (AJ)}{\partial x} - k(c^* A) + J_{wa} . O_x - J_{wb} . P_x \tag{3.6}$$

Assuming sorption to suspended solids is a linear process, so

$$X_{ss} = K_{L,ss} \cdot c \tag{7.40}$$

with
$K_{L,ss}$ = slope of (linear) sorption isotherm of suspended solids ($L^3.M^{-1}$)

and combining Eqs. (4.1) and (4.4):

$$c^* = \left(1 + \frac{DW\,P_{z=0}}{A} K_{mp} + ss\,K_{L,ss}\right) c \qquad (7.41)$$

For constant wetted areas A, assuming exclusively convective, downward seepage, and substituting Eqs. (4.6), (4.11) and (4.14), the conservation equation, Eq. (3.6) now reads:

$$\left(1 + \frac{DW\,P_{z=0}}{A} K_{mp} + ss\,K_{L,ss}\right) \frac{\partial c}{\partial t} = -u\,(1 + ss\,K_{L,ss}) \frac{\partial c}{\partial x} + E_x(1 + ss\,K_{L,ss}) \frac{\partial^2 c}{\partial x^2}$$

$$- k \left(1 + \frac{DW\,P_{z=0}}{A} K_{mp} + ss\,K_{L,ss}\right) c - k_{t,l} \left(c - \frac{c_a}{K_H}\right) \frac{O_x}{A} - \frac{q}{A} c \Leftrightarrow$$

$$R_{wl} \frac{\partial c}{\partial t} = -u \frac{\partial c}{\partial x} + E_x \frac{\partial^2 c}{\partial x^2} - k\,R_{wl}\,c - \frac{k_{t,l}\,O_x}{(1 + ss\,K_{L,ss})\,A} c - \frac{q}{A\,(1 + ss\,K_{L,ss})} c \qquad (7.42)$$

(assume c_a, mass concentration of substance in the air, equals zero)

in which
R_{wl} = retardation factor for the water subsystem (1)

$$R_{wl} = \frac{\left(1 + \frac{DW\,P_{z=0}}{A} K_{mp} + ss\,K_{L,ss}\right)}{1 + ss\,K_{L,ss}}$$

For a finite system with $\varepsilon = 1.0$ its solution for a pulse input at $t = 0$ reads (in the notation used for the TOXSWA model description) (compare Eq.(7.35)):

$$c = \frac{M}{2R_{wl}\,(1 + ss\,K_{L,ss})} \frac{1}{\sqrt{\pi\,E_x\,\frac{t}{R_{wl}}}}$$

$$\cdot \exp\left\{-\left(k + \frac{k_{t,l}\,O_x}{(1 + ss\,K_{L,ss})\,A\,R_{wl}} + \frac{\ell q}{A\,(1 + ss\,K_{L,ss})\,R_{wl}}\right) t\right\}$$

$$\cdot \exp\left\{-\frac{\left(x - u\,\frac{t}{R_{wl}}\right)^2}{4E_x\frac{t}{R_{wl}}}\right\} \qquad (7.43)$$

7.3.4 Comparison of the analytical solutions with the numerical solutions (i.e. TOXSWA)

To be able to compare the analytical solutions of section 7.4.2 with TOXSWA model calculations the source code of TOXSWA has been changed in such a way that there was no diffusion across the water-sediment interface (See Annex 15). So, only advective transport took place across the water-sediment interface.

In the sediment subsystem a pulse input was applied at $t = 0$ in the first node, i.e. at $z = 0.0005$ m, in case of the TOXSWA calculation. The applied pulse corresponded to an initial total concentration c_b^* of 10 g/m^3 in the first segment of 1 mm thickness. For the analytical solution (Eq. (7.39) the same mass (corresponding to 10 mg/m^2) was applied at the sediment surface. To be able to compare the TOXSWA results with the analytical solution a rectangular-shaped cross section of the ditch (so constant perimeter P with depth) was selected and bulk density, porosity and organic matter content were kept constant with depth. Sorption to solid bottom material was described with a linear sorption isotherm and there was a constant downward seepage of 2 mm/(m^2.d), expressed as an infiltration flux per m^2 neighbouring field lot. All other input parameters were kept identical to the ones of the example simulation of chlorpyrifos of Chapter 8. Annex 16 summarizes the differences in the input files between the example simulation of Chapter 8 and the simulation carried out here to compare the numerical with the analytical solution for the sediment subsystem.

Figure 21 shows the concentration profiles in the upper centimeters of the sediment, calculated by the TOXSWA model and by the analytical solution, Eq. (7.39). Figure 21 shows an excellent correspondence between the numerical and the analytical solution. For the analytical solution Table 3 details the situation at the first node of 0.0005 m for the period of 0.0 to 0.5 day after application of the pulse load at the sediment surface. Table 3 demonstrates that it takes 0.0625 day for this pulse to arrive at 0.0005 m depth in the sediment. Due to dispersion and diffusion the concentration peak already lowered and therefore this peak is lower than the concentration peak that the TOXSWA model calculated for the first node in the sediment directly after application (t = 0.001 d). TOXSWA calculated a concentration peak of 9.97 g/m^3 at t = 0.001 d (i.e. after sorption to bottom material and some transport to the second node) (see also Fig. 21).

In the water subsystem a pulse input was applied at $t = 0$ in node 8 at $x = 60$ m in the ditch for the TOXSWA simulation. The pulse corresponded to an initial total concentration c^* of 8.264 g/m^3 in the eighth segment of 8 m long. For the analytical solution, Eq. (7.43), the same total mass (corresponding to 6.6112E-02 g/m^2, expressed per m^2 wetted cross section area A) was applied at 60 m in the ditch. Total ditch length was 400 m in this TOXSWA simulation, the ditch cross section was again rectangular and there was a constant downward seepage of 2 mm/(m^2.d), expressed as an infiltration flux per m^2 neighbouring field lot. Sorption to suspended solids was described with a linear sorption isotherm and all other input parameters were identical to the ones of the example simulation for chlorpyrifos of Chapter 8. Annex 17 gives an overview of the differences in the input files used here and those used in the example simulation of Chapter 8.

Table 3 Total pesticide concentrations in node 1(z = 0.0005 m) of the sediment as a function of time, calculated with the aid of the analytical solution for the sediment, Eq.(7.39)

Time (d)	Total concentration (g.m^{-3})
0.0125	2.96
0.0250	7.30
0.0375	9.03
0.0500	9.62
0.0625	9.73
0.0750	9.64
0.0875	9.46
0.1	9.24
0.2	7.56
0.3	6.45
0.4	5.70
0.5	5.14

Figure 22 shows the concentration profiles in the water layer of the ditch, calculated by the TOXSWA model and by the analytical solution, Eq. (7.43). Correspondence is again excellent. So, TOXSWA 1.0 has been verified successfully both for the water and sediment subsystems.

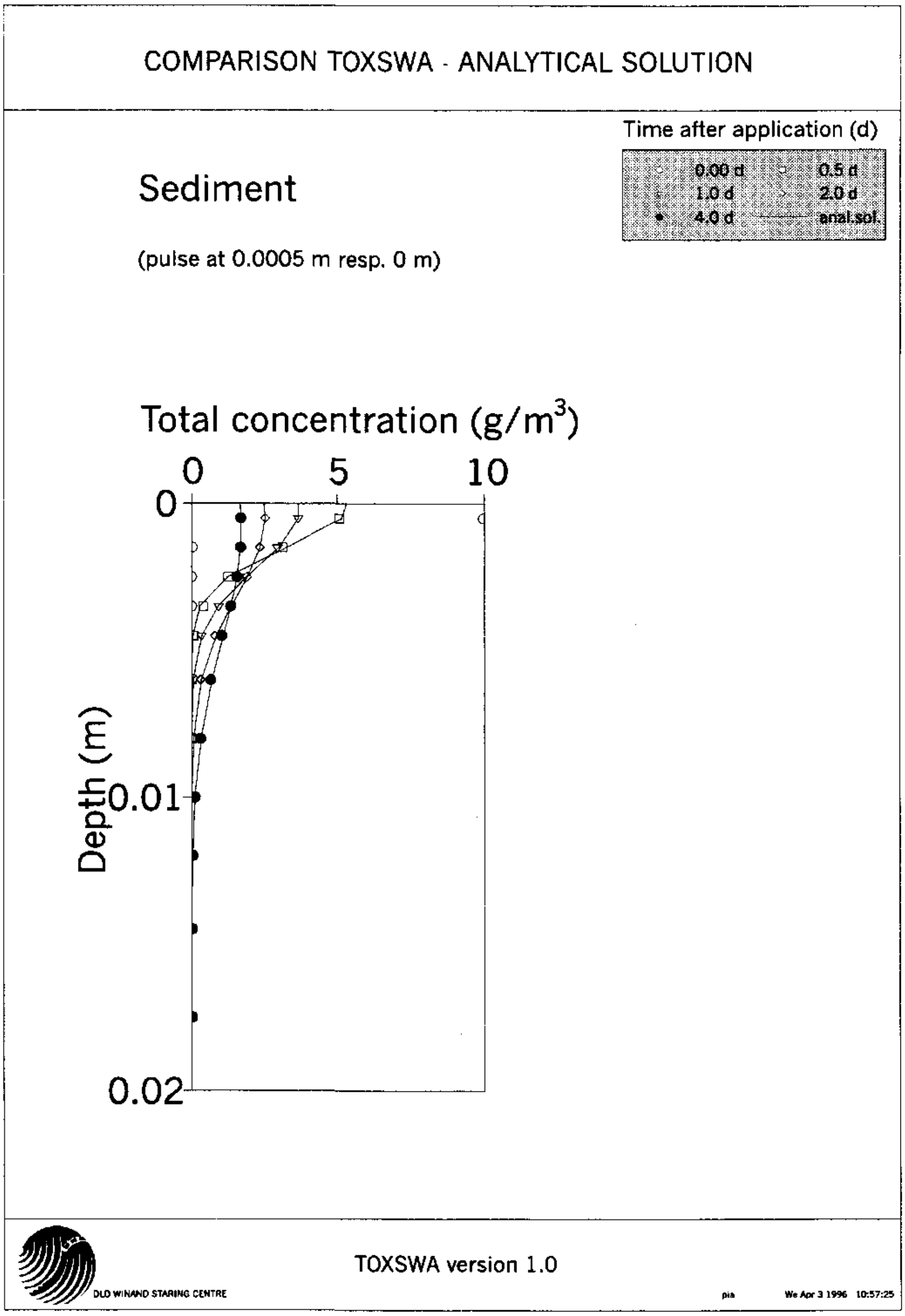


Fig. 21 Comparison of the total chlorpyrifos concentration in the sediment calculated with the aid of the TOXSWA model (indicated by the markers) and with the aid of the analytical solution for the mass conservation equation of the sediment, Eq.(7.39) (indicated by the drawn lines)

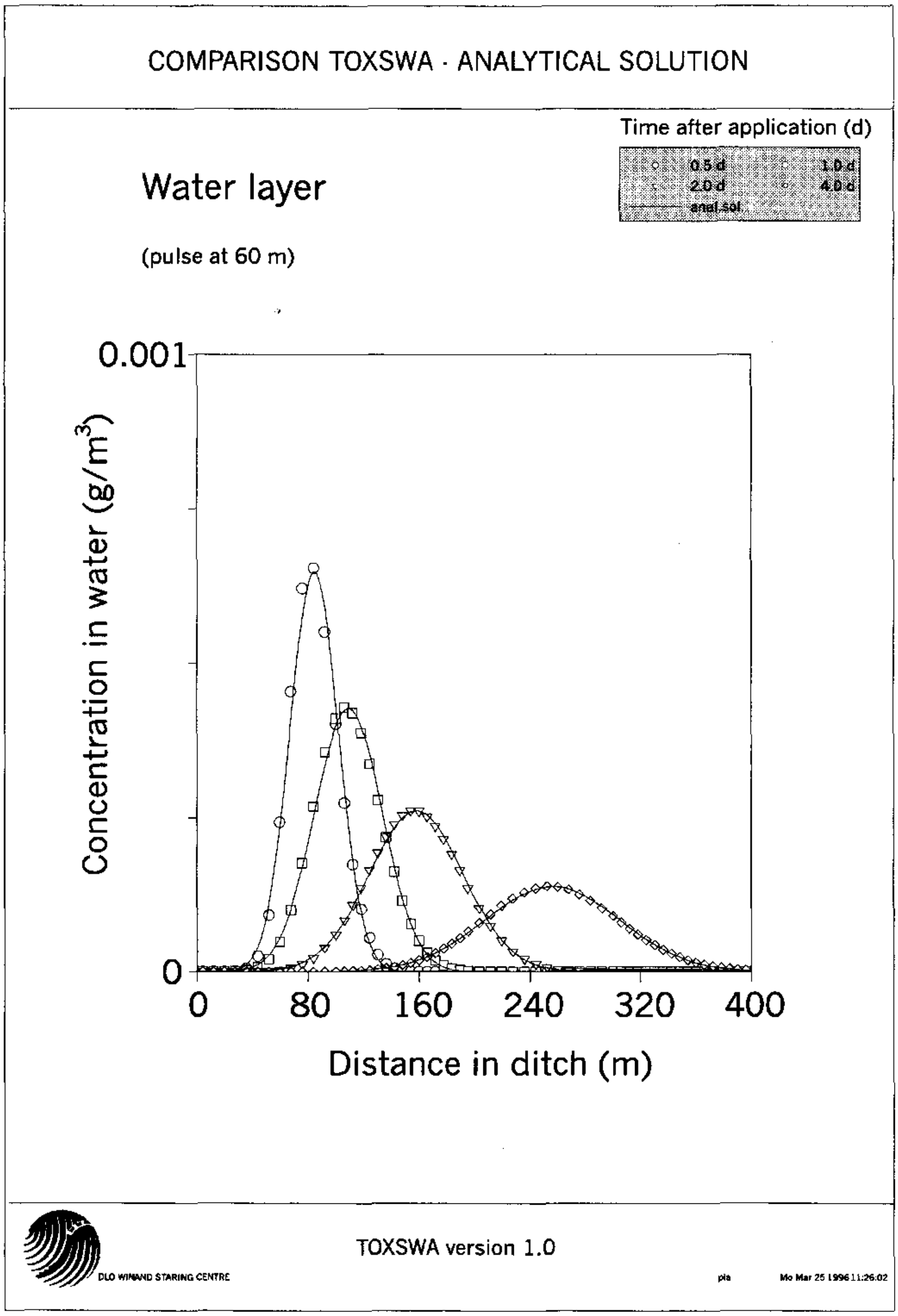


Fig. 22 Comparison of the chlorpyrifos concentration in the water phase of the ditch calculated with the aid of the TOXSWA model (indicated by the markers) and with the aid of the analytical solution for the mass conservation equation of the water layer, Eq.(7.43) (indicated by the drawn lines)

8 Illustration of the potential use of TOXSWA: simulation of chlorpyrifos behaviour after spray drift deposition

8.1 Design of computation and values of parameters

To illustrate the possibilities of the TOXSWA model a computation was carried out for the insecticide chlorpyrifos. The following situation was simulated. Of an application rate of 1 kg a.i. per ha 3% is deposited by spray drift onto a neighbouring ditch. The ditch is 200 m long and deposition of chlorpyrifos occurs over the total length of the ditch except the first 24 m. Water flow rate in the ditch is 100 m/d, the water depth is 50 cm and the cross section of the ditch has a trapezoidal shape. An amount of 250 g of dry macrophyte biomass is present per m^2 ditch bottom, corresponding to a moderately-grown ditch in summertime; the suspended solids concentration is 50 g/m^3. Initially, the water does not contain chlorpyrifos.

The sediment layer in the ditch is 10 cm thick and porosity varies from 80% in the top mm to 40% in the bottom cm. The organic matter content varies from 8% to 0.5%. There is no upward or downward seepage in the sediment. Initially, the sediment is free of pesticide.

The explicit central difference calculation scheme was selected to solve the mass conservation equations. Distances between the nodes in the water and sediment subsystems were 6 to 8 m and 1 to 10 mm, respectively. The time step was 100 s and the run time on a Pentium 90 MHz was 32 minutes.

Chlorpyrifos transformation is characterised by a half-life time of 75 and 175 d in the water layer and the sediment, respectively; these values have been determined at the DLO Winand Staring Centre in earlier experiments. Sorption to sediment is very strong (K_{om} = 16.4 m^3/kg and n = 0.984) (Pers. comm. Crum) as is sorption to macrophytes (K_{mp} = 2.0 m^3/kg) (Van Huffelen, 1993).

The input files for this example simulation are presented in Annex 18.

8.2 Results and discussion

Figure 23 shows the chlorpyrifos concentration in the water phase as well as the total mass concentration in the sediment as a function of time and depth. The upper graph shows that, due to dispersion, the concentration front flattens as it moves out of the ditch. The spray drift deposition of 0.03 kg/ha on the ditch results in an initial concentration of 4.4 μg/l in the water (after the instantaneous sorption to the macrophytes and suspended solids). Without any sorption the initial concentration would have been 8.3 μg/l. The lower graphs show the penetration of chlorpyrifos into the top 2 cm of the sediment. After 4 days, total concentrations built up vary from negligible (< 0.3 $\mu g/dm^3$) at 4 m along the ditch up to 415 $\mu g/dm^3$ at 196 m along the ditch. After this time, back-diffusion from the sediment into the water layer has already started, a process which is driven by the difference in pesticide concentration between the liquid phase (i.e. not the total mass concentration) of the sediment and the water in the ditch. The graph for the situation at 100 m clearly shows this phenomenon, as the total concentration is already decreasing. It can be derived from Figure 18 that the average rate of displacement of the pesticide in the water layer is about 60 m/d, so about 60% of the water flow rate.

Figure 23 also presents the average concentration to which aquatic organisms are exposed at 3, 21 and 28 days after application, as well as immediately after application (at 0 d). The exposure concentrations have been determined by calculating the average of the concentration course for pesticide dissolved in the water phase with time. (This implies that the exposure concentrations are calculated according to the assumptions of instantaneous mixing and sorption.) The concentration presented at time t = 0 d corresponds to the total pesticide concentration, immediately after application, so this means before any adsorption has taken place. The exposure concentration has been defined as the concentration at that position in the ditch where the longest exposure duration is expected, i.e. at the downstream end of the section of the ditch where the pesticide input took place. In this example the exposure concentrations were 8.3 μg/l immediately after application and 3.5 μg/l at t = 3 d.

Figure 24 presents the distribution of chlorpyrifos between the different compartments as a function of the time since application. The upper graph indicates that a total of nearly 2 g of the insecticide has been deposited in the ditch. Initially, 53% is dissolved in the water phase, while 42% and 4% are adsorbed to the macrophytes and suspended solids, respectively. After 4 days, the largest part (72%) of the remaining mass (0.1 g) is found in the sediment, 15% is dissolved in the water phase, 12% has been adsorbed to the macrophytes and 1.3% to the suspended solids. The lower graphs present the chlorpyrifos mass per running metre at selected locations in the ditch. Roughly speaking, they present the same pattern as that shown above, albeit with a more rapid decline in pesticide mass, due to the passage of the concentration front through the water layer. The longer chlorpyrifos is present in the overlying water layer, the more it penetrates into the sediment. In the first part of the ditch, a negligible amount (< 0.6 μg/m') (not visible) has penetrated via diffusion into the sediment, due to 'back dispersion' of the insecticide into this part of the overlying water layer.

Figure 25 shows the mass balances of chlorpyrifos as a function of time for the entire ditch. Separate mass balances are presented for the two subsystems, water layer and sediment. The total mass in the water layer decreases steadily, mainly due to outflow. Other factors contributing to the decrease are, in order of priority, volatilisation, penetration into sediment and transformation. The total mass in the sediment increases up to about 3 days after application. From ca. day 2 onwards, back-diffusion to the water layer becomes significant, as well as transformation, and the total mass in the sediment gradually starts to decrease. Initially, nearly 2 g of chlorpyrifos is present in the water layer, whereas a maximum of about 0.1 g can be found in the sediment later.

The TOXSWA model calculates how much mass is missing in the mass balances for each subsystem, i.e the (only) water subsystem and, in this case, 29 sediment subsystems. This missing quantity is expressed as a percentage of the initial mass plus the incoming additional mass for each subsystem. In the example simulation for chlorpyrifos the missing quantity in the mass balance for the entire water layer (subdivided into 29 nodes) after 3456 time steps of 100 s (corresponding to 4.00 d) was 0.0037 % ($0.72.10^{-4}$ g) of the dose. For the entire sediment, so the summation of the 29 sediment subsystems, the missing mass was 0.0042% ($0.43.10^{-5}$ g) of the initial mass plus the incoming mass from the water layer.

The three types of graph in Figures 23, 24 and 25 give an overview of the fate of the pesticide. Concentrations in water layer and sediment are shown and the mass balance graphs explain how these concentrations have developed. The distribution between the different compartments indicates where the insecticide can be found at different points in time. This example of a pesticide with a high sorption capacity shows that considerable amounts of the pesticide can be found in the macrophyte and sediment compartments. Instantaneous sorption equilibrium is assumed in the model. Measurements during past experiments with chlorpyrifos have indicated, however, that it takes about one day before chlorpyrifos deposited onto a 50 cm deep ditch is mixed over the entire depth of the water layer (Crum and Brock, 1994). The sorption equilibrium with macrophytes and with sediment may also take up to about 24 hours (Van Huffelen, 1993). This means that the simulated pesticide concentrations in the water phase may be too low and that the simulated mass concentration in the macrophyte compartment especially may be too high during the first one or two days.

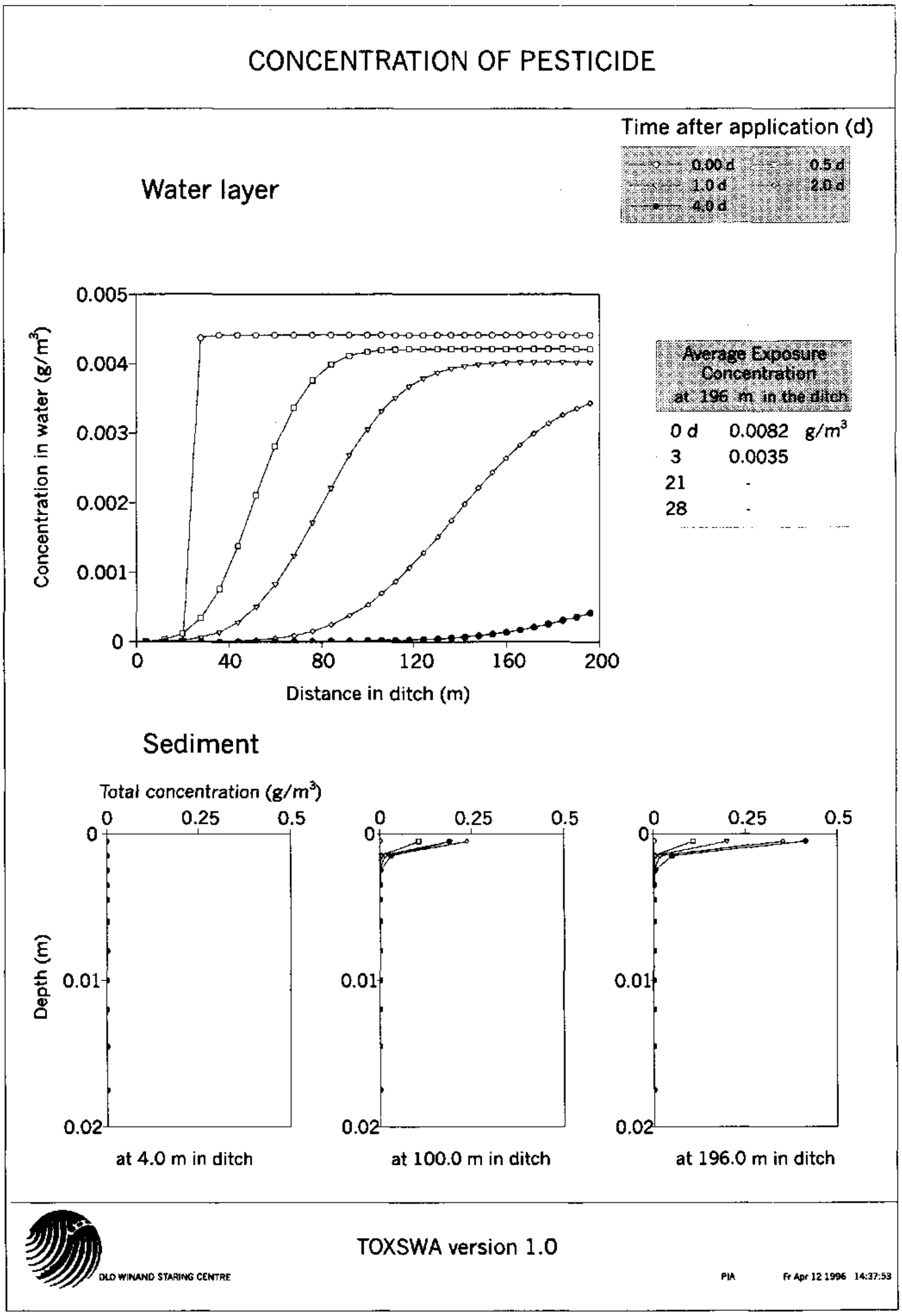


Fig. 23 Chlorpyrifos concentration in the water layer and at selected locations in the sediment after spray drift deposition of 0.03 kg.ha^{-1} onto the ditch

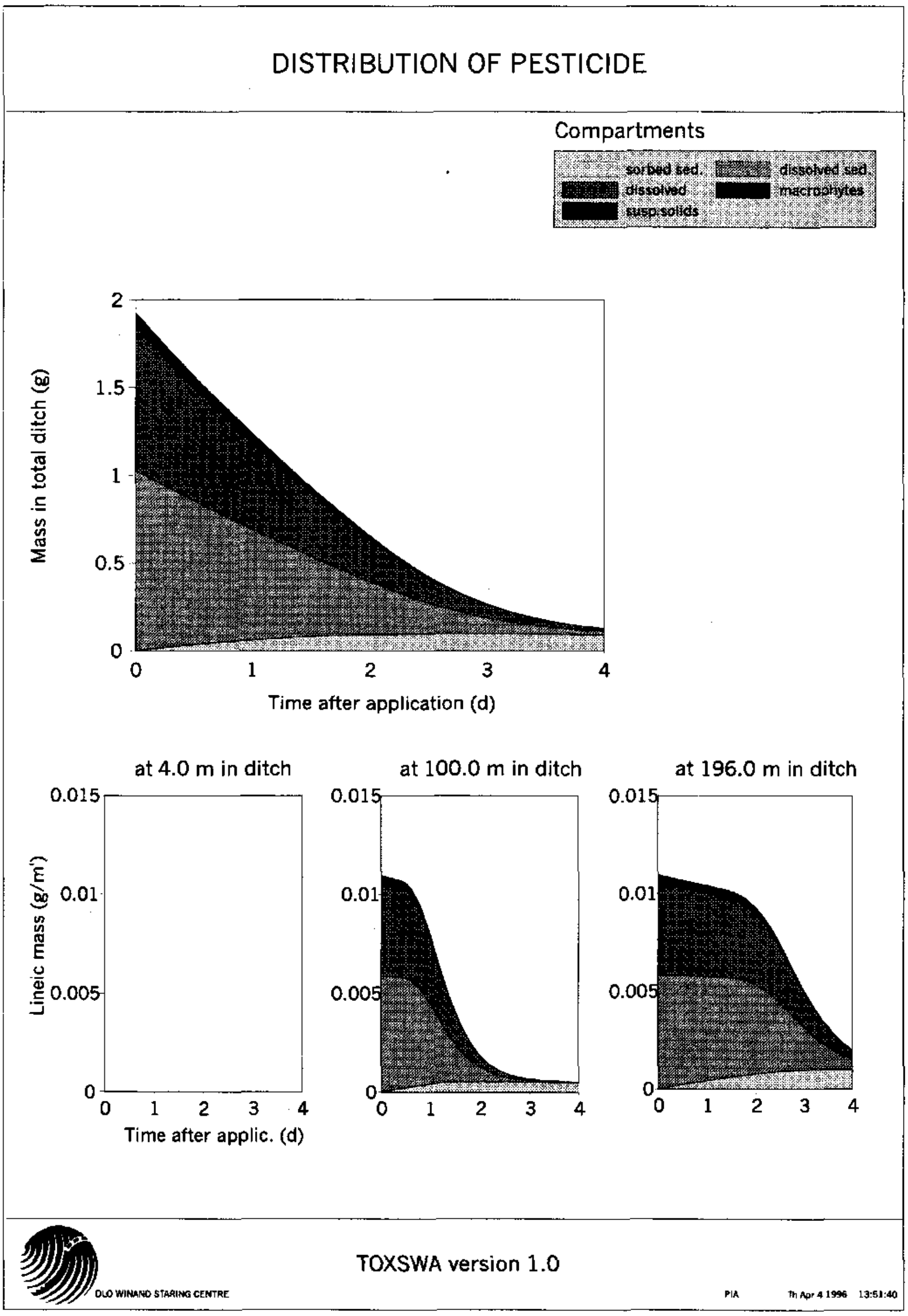


Fig. 24 Distribution of chlorpyrifos between the different compartments as a function of time for the entire ditch (200 m) as well as per running metre at selected locations. (The mass dissolved in the liquid phase of the sediment is too small to be visible)

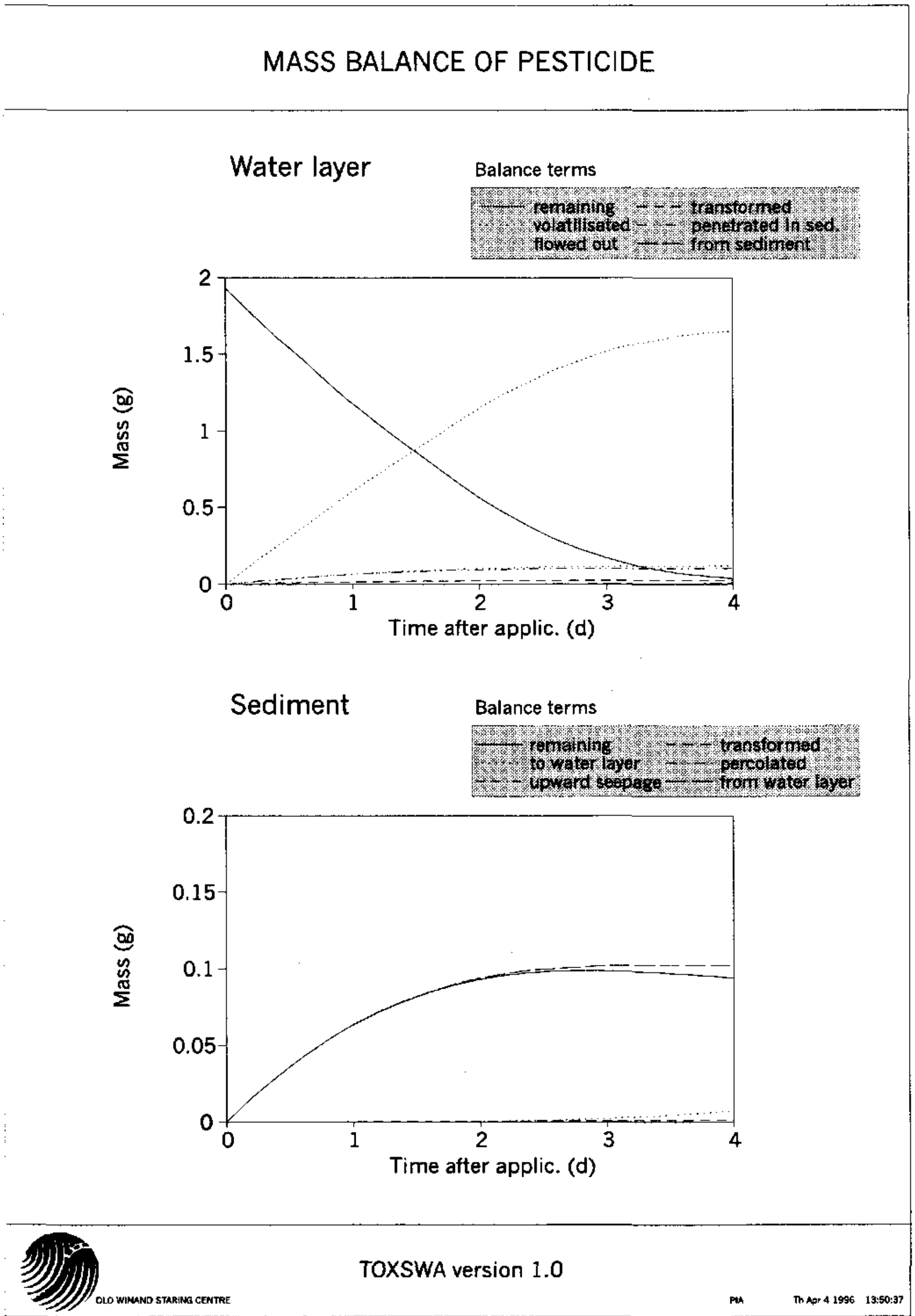


Fig. 25 Mass balances of chlorpyrifos as a function of time for the entire ditch (200 m). Separate mass balances are shown for the water and sediment subsystems

9 Conclusions, discussion and recommendations

9.1 Conclusions and discussion

The TOXSWA model has been developed to describe the fate of pesticides in field ditches. It calculates pesticide concentrations in the horizontal direction in the water layer and in the vertical plus horizontal directions in the sediment. It can handle a variety of situations as regards hydrological conditions and entry routes of pesticides into surface water.

A limited verification of the model has taken place by comparing model output with an analytical solution for the sediment subsystem, as well as for the water subsystem. In both cases the concentration profiles calculated by the model showed an excellent correspondence with those calculated according to the analytical solution. So, TOXSWA 1.0 has been verified successfully both for the water and the sediment subsystems.

An example simulation decribed the behaviour of the insecticide chlorpyrifos after spray drift deposition in a ditch. Results showed that initially, nearly half of the insecticide was adsorbed to the macrophytes and this provokes a retardation of the concentration front of about 60% compared to the water flow rate. After four days about 70% of the remaining pesticide mass was found in the sediment.

The TOXSWA model checks whether the mass balances tally during the calculations. Results for the example simulation for chlorpyrifos showed that after four days the missing quantity in the mass balance for the water layer was less than 0.005% of the dose applied. The missing mass for the entire sediment layer was also less than 0.005% of the initial mass plus incoming mass of the water layer after four days, so this shows that conservation of pesticide mass applied is well done in TOXSWA 1.0.

TOXSWA has been developed to serve as a tool in the pesticide registration procedure in the Netherlands. It has been designed to estimate chronic exposure of aquatic organisms to pesticides, so it simulates periods of up to about one month. Of course, the TOXSWA model also estimates acute exposure, but the simulated acute concentrations might differ from those encountered in the field, due to assumptions made in the model. In reality, pesticides, especially those with low solubilities, may need 24 hours before they are thoroughly mixed over the entire cross section of the ditch; subsequently, it takes time before sorption equilibrium is reached. The TOXSWA model, however, assumes instantaneous mixing over the entire cross section, as well as instantaneous sorption equilibrium with suspended solids, macrophytes and with the solid bottom material (once the pesticide has entered the sediment subsystem).

In accordance with the aim of its development (exposure concentrations for up to about one month) it is assumed that no sedimentation or resuspension occurs. This means that the TOXSWA model cannot be used to estimate long-term exposure concentrations or accumulation of pesticides in the sediment.

9.2 Recommendations

If one is to make full use of all the possibilities the TOXSWA model offers, model improvements are needed:
- inclusion of multiple or continuous pesticide applications to the water layer;
- implementation of varying water depths and rates of discharge in the water layer, coupled to incoming water via entry routes like upward seepage and surface runoff; and
- making the presentation of the model output more user-friendly.

The sensitivity of the model to input parameters, initial and boundary conditions needs to be studied. This can guide further experimental work as well as future use of the TOXSWA model, e.g. when standard scenarios are defined.

Standard scenarios need to be defined to facilitate the use of the model for risk assessments in the Dutch registration procedure. The standard scenarios should comprise the environmental conditions, such as water depth, flow velocity, mass of macrophytes, as well as the pesticide masses entering the ditch by one or several entry routes.

It should be demonstrated that the TOXSWA model simulates correctly real field situations, which means that the model needs to go through a validation process. The validation process is defined as a comparison of the model output with data independently derived from experiments or observations of the environment; this implies that none of the input parameters is obtained via calibration. Validation should be done for a specified range of validity, i.e. that part of reality to which the validation of a model applies (Leaching Modelling Workgroup FOCUS, 1995). Four experiments performed at the DLO Winand Staring Centre provide data sets for stagnant water bodies and one slowly moving water body, and these will be used first. Subsequently, data sets from elsewhere need to be assessed as to their potential use for the validation process. More experiments are probably needed to validate the model, especially in systems with a certain flow velocity.

Preliminary results of a sensitivity analysis for the water subsystem showed that the dispersion coefficient is a very important coefficient to describe well the (longitudinal) mixing of the substance in the watercourse. More experiments need to be performed to estimate the dispersion coefficient in various ditches under various conditions. Very few data exist, especially for small watercourses.

The process descriptions need to be worked out in greater detail. Transformation rates in water and sediment need to be studied more closely, as well as the influence on these rates of pH, light intensity, bioactivity and temperature conditions. Furthermore, the effect of aerobic and anaerobic conditions on the rate of transformation of the substance in the upper few millimetres below the water-sediment interface needs further study. Very little information is available on sorption to macrophytes, although this process clearly affects the level and duration of the pesticide concentration. More experimental work is needed to determine this sorption coefficient. A study on the sorption of a range of pesticides to three plant species is currently being carried out at the DLO Winand Staring Centre.

In order to assess whether the TOXSWA model is able to simulate the fate of pesticides in watercourses in other countries of the EU, field experiments need to be executed outside the Netherlands. At the moment, the TOXSWA model is being assessed by the surface water fate group of the EU working party called FOCUS, FOrum for the Co-ordination of pesticide fate models and their USe.

References

Bear, J. and A. Verruijt, 1987. *Modeling Groundwater Flow and Pollution. (Theory and applications of transport in porous media).* Holland, Kluwer.

Bella, D.A. and W.E. Dobbins, 1968. 'Difference modeling of stream pollution'. *Journal of the Sanitary Engineering*, Am. Soc. Civ. Engrs. 94: 995-1016.

Beltman, W.H.J., P.I. Adriaanse and M.J.B. van Elswijk, 1996. *TOXSWA 1.0. User's manual.* Wageningen, DLO Winand Staring Centre. Technical Document 33.

Boesten, J.J.T.I., 1986. *Behaviour of herbicides in soil: simulation and experimental assessment.* Wageningen, Institute for Pesticide Research. Doctoral thesis.

Bolt, G.H. (ed), 1979. *Soil chemistry. B. Physico-chemical models.* Amsterdam, Elsevier.

Crum, S.J.H. and T.C.M. Brock, 1994. *Fate of chlorpyrifos in indoor microcosms and outdoor ditches.* In: Freshwater Field Tests for Hazard Assessment of Chemicals, eds. I.A. Hill, F. Heimbach, P. Leeuwangh & P. Matthiesen. Michigan, USA, Lewis Publishers.

Crum, S.J.H. and J.W. Deneer (eds.), 1995. *Development of the TOXSWA model for predicting the behaviour of pesticides in surface water.* Proceedings of a workshop, held on November 8, 1994 at Wageningen, the Netherlands. Wageningen, DLO Winand Staring Centre. Report 105.

Crum, S.J.H., G.H. Aalderink, H.W. Koopman and T.C.M. Brock, in prep. *The fate of the herbicide linuron in outdoor experimental ditches.*

Genuchten, M.Th. van, and P.J. Wieringa, 1974. *Simulation of one-dimensional solute transfer in porous media.* Las Cruces, New Mexico. New Mexico State University's Agricultural Experiment Station Bulletin 628.

Heer, H. de, 1979. *Measurements and computations on the behaviour of the insecticides azinphos-methyl and dimethoate in ditches.* Agric. Res. Rep. 884. Wageningen, Pudoc.

Jafvert, C.T., 1990. 'Sorption of organic acid compounds to sediments: initial model development'. *Environ. Toxicology and Chemistry* 9: 1259- 1268.

Jury, W.A. and K. Roth, 1990. *Transfer functions and solute movements through soil. Theory and applications.* Birkhäuser Verlag Basel.

Kroes, J.G. en J.J.T.I. Boesten, 1993. *Vergelijking van de uitspoeling berekend met de modellen TRANSOL en PESTLA*. DLO-Staring Centrum, Rapport 238, Wageningen.

Lapidus, L. and G.F. Pinder, 1982. *Numerical solution of partial differential equations in science and engineering*. Chichester, Wiley.

Leaching Modelling Workgroup FOCUS (FOrum for the Co-ordination of pesticide fate models and their USe), 1995. *Leaching models and EU registration, final report*. Brussels, Document 4952/VI/95.

Linders, J.B.H.J., R. Luttik, J.M. Knoop en D. van der Meent, 1990. *Beoordeling van het gedrag van bestrijdingsmiddelen in oppervlaktewater in relatie tot expositie van waterorganismen*. RIVM, Rapport nr. 678611002, Bilthoven.

Liss, P.S. and P.G. Slater, 1974. 'Flux of gasses across the air-sea interface'. *Nature* 247: 181-184.

Mackay, D. and P.J. Leinonen, 1975. 'Rate of evaporation of low-solubility contaminants from water bodies to atmosphere'. *Environ. Sci. Technol.* 9: 1178-1180.

Mackay, D., 1981. *Environmental and laboratory rates of volatilization of toxic chemicals from water in hazard assessment of chemicals, current developments*. New York, Vol. 1, J. Saxena and F. Fischer, Eds., Academic Press.

Ministry of Housing, Spatial Planning and Environment, 1995. *Besluit van 23 januari 1995, houdende regelen als bedoeld in artikel 3a, eerste lid, van de bestrijdingsmiddelenwet 1962 (Besluit milieutoelatingseisen bestrijdingsmiddelen)*. 's-Gravenhage, The Netherlands, Staatsblad 37 (1995).

Schurer, K. en J.C. Rigg, 1980. *Grootheden en eenheden in de landbouw en de biologie*. Wageningen, Pudoc.

Thomann, R.V. and J.A. Mueller, 1987. *Principles of surface water quality modelling and control*. New York, Pub. Harper and Row.

Varga, R.B., 1962. *Matrix Iterative Analysis*. Prentice Hall, New Jersey.

Wauchope, R.D. and R.S. Myers, 1985. 'Adsorption -desorption kinetics of atrazine and linuron in freshwater-sediment aqueous slurries'. *J. Environ. Qual.* 14: 132-136.

Unpublished sources

Aalderink, R.H., 1993. *Systeemanalyse in waterkwaliteitsbeheer II*. Wageningen, Wageningen Agricultural University. Lecture notes (in prep.)

Aalderink, G.H en S.J.H. Crum, 1994. *Het gedrag van het herbicide linuron in aquatische model-ecosystemen.* Wageningen, DLO-Staring Centrum. Interne mededeling 330.

Boesten, J.J.T.I., 1992. *Source code for the function CRSEMI, calculating the resident concentration in the liquid phase for the semi-infinite Convection-Dispersion Equation.*

Crum, S.J.H., 1996, personal communication. *Calculation of K_F and n for sorption of chlorpyrifos to sediment material based at an organic matter content of 4%, on basis of a decrease in water phase concentration and with a reference concentration of 0.001 mg/dm^3.*

Huffelen, E.A. van, 1993. *Sorptie van chloorpyrifos aan de waterplanten Chara globularis (kranswier), Elodea nuttallii (waterpest) en Lemna gibba (kroos).* DLO-Staring Centrum, Wageningen. Stageverslag Rijkshogeschool IJselland.

List of symbols

A_x	=	cross-sectional area of flow at location x	(L^2)
b	=	width of ditch bottom	(L)
$b_d(t,x)$	=	distributed source with continuous input	$(M.L^{-3}.T^{-1})$
$b_p(t,x)$	=	point source with continuous input	$(M.L^{-3}.T^{-1})$
c	=	mass concentration of substance in the water phase	$(M.L^{-3})$
c^*	=	mass concentration substance in water layer (this includes substance sorbed to suspended solids and to macrophytes)	$(M.L^{-3})$
c_a	=	mass concentration of substance in the air	$(M.L^{-3})$
$c_{a,I}$	=	equilibrium mass concentration of substance at the water-gas interface in the gas phase	$(M.L^{-3})$
c_b^*	=	mass concentration of substance in sediment	$(M.L^{-3})$
$c_{e,ss}$	=	concentration c, at which $K_{F,ss}$ has been estimated	$(M.L^{-3})$
$c_{e,wb}$	=	concentration c, at which $K_{F,wb}$ has been estimated	$(M.L^{-3})$
c_I	=	equilibrium mass concentration of substance at the water-gas interface in the water phase	$(M.L^{-3})$
c_{lb}	=	mass concentration of substance in the liquid phase of sediment	$(M.L^{-3})$
c_{sol}	=	solubility of substance in water	$(M.L^{-3})$
d	=	distance of water-sediment interface to concerned area	(L)
D_{lb}	=	diffusion coefficient of substance in the liquid phase of sediment	$(L^2.T^{-1})$
D_w	=	diffusion coefficient of substance in water	$(L^2.T^{-1})$
DW	=	dry weight of macrophytes per area of sediment	$(M.L^{-2})$
E_{lb}	=	dispersion coefficient in pore water	$(L^2.T^{-1})$
E_x	=	dispersion coefficient in direction of flow	$(L^2.T^{-1})$
h	=	water level above ditch bottom	(L)
h_w	=	water level above ditch bottom, defining the exchanging perimeter $P_{z=0}$	(L)
i	=	index of grid points (segments) in space in x direction	(1)
j	=	index of grid points in time	(1)
J	=	areic mass flux of substance in water layer by advection and dispersion	$(M.L^{-2}.T^{-1})$
J_{lb}	=	areic mass flux of substance in the liquid phase of the sediment by advection, dispersion and diffusion	$(M.L^{-2}.T^{-1})$
J_{wa}	=	areic mass flux of substance across the water-air interface, the flux is negative in upward direction	$(M.L^{-2}.T^{-1})$
J_{wb}	=	areic mass flux of substance across the water-sediment interface, the flux is positive in downward direction	$(M.L^{-2}.T^{-1})$
$J_{wb,adv}$	=	areic mass flux by advection at the water-sediment interface	$(M.L^{-2}.T^{-1})$
$J_{wb,dif}$	=	areic mass flux by diffusion at the water-sediment interface	$(M.L^{-2}.T^{-1})$
k	=	index of grid points (segments) in space in z direction	(1)

k	=	transformation rate coefficient for substance in the water column	(T^{-1})
k_b	=	transformation rate coefficient for substance in the sediment	(T^{-1})
k_g	=	exchange coefficient of substance in the gas phase	$(L.T^{-1})$
k_l	=	exchange coefficient of substance in the liquid phase	$(L.T^{-1})$
$k_{t,l}$	=	overall transfer coefficient for the air-water interface, based at the liquid phase	$(L.T^{-1})$
K_{mp}	=	distribution coefficient for substance between macrophytes and water, i.e. slope of sorption isotherm based at the mass of dry macrophytes	$(L^3.M^{-1})$
$K_{om,ss}$	=	slope of sorption isotherm for suspended solids, based at the organic matter content	$(L^3.M^{-1})$
$K_{om,wb}$	=	slope of sorption isotherm for sediment, based at the organic matter content	$(L^3.M^{-1})$
$K_{F,ss}$	=	Freundlich coefficient for sorption to suspended solids	$(L^3.M^{-1})$
$K_{F,wb}$	=	Freundlich coefficient for sorption to bottom material	$(L^3.M^{-1})$
K_H	=	dimensionless Henry coefficient	(1)
$K_{L,b}$	=	slope of (linear) sorption isotherm of bottom material	$(L^3.M^{-1})$
$K_{L,ss}$	=	slope of (linear) sorption isotherm of suspended solids	$(L^3.M^{-1})$
L_{dis}	=	dispersion length	(L)
LBD	=	band above the diagonal of the left-hand tridiagonal matrix	(1)
LDD	=	diagonal of the left-hand tridiagonal matrix matrix	(1)
LOD	=	band under the diagonal of the left-hand tridiagonal matrix	(1)
ℓ	=	length of drained or infiltrated lot, oriented perpendicular to the ditch and extended on one or two sides of the ditch	(L)
$m_{om,ss}$	=	mass fraction of organic matter of the suspended solids	$(M.M^{-1})$
$m_{om,wb}$	=	mass fraction of organic matter of the sediment material	$(M.M^{-1})$
M	=	applied area-averaged mass of substance	$(M.L^{-2})$
M_m	=	molecular mass	$(M.N^{-1})$
n_{ss}	=	Freundlich exponent for sorption to suspended solids	(1)
n_{wb}	=	Freundlich exponent for sorption to bottom material	(1)
NWB	=	Numerical Weight factor sediment [water Bottom]	(1)
NWW	=	Numerical Weight factor Water layer	(1)
O_x	=	width water surface at location x	(L)
$p_d(t,x)$	=	distributed pulse input	$(M.L^{-3}.T^{-1})$
$p_p(t,x)$	=	point-type pulse input	$(M.L^{-3}.T^{-1})$
P	=	saturated vapour pressure of substance	$(L^{-1}.M.T^{-2})$
P_0	=	wetted perimeter	(L)
P_d	=	length wetted perimeter at distance d from the water-sediment interface	(L)
P_x	=	wetted perimeter at location x	(L)
$P_{z=0}$	=	length wetted perimeter at depth $z = 0$	(L)

q	=	areic volume flux, i.e. volume drained or supplied water divided by lot area and time. The flux is positive with infiltration and negative with upward flow (drainage from the field lot)	$(L^3.L^{-2}.T^{-1})$
Q	=	rate of discharge in water layer	$(L^3.T^{-1})$
r	=	number of point-type inputs (total n inputs)	
R	=	universal gas constant	$(L^2.M.T^{-2}.N^{-1}.\theta^{-1})$
R_{wb}	=	retardation factor for the sediment subsystem	(1)
R_{wl}	=	retardation factor for the water subsystem	(1)
RBD	=	band above the diagonal of the right-hand tridiagonal matrix	(1)
RDD	=	diagonal of the right-hand tridiagonal matrix	(1)
ROD	=	band under the diagonal of the right-hand tridiagonal matrix	(1)
RV	=	right-hand vector	(1)
s	=	number pulse input (total m inputs)	
s_0	=	number pulse input at location x_0 (total m_0 inputs)	
s_{00}	=	number pulse input at location x_{00} (total m_{00} inputs)	
s_1	=	side slope, horizontal/vertical	(1)
ss	=	mass concentration of suspended solids in the water layer, i.e. the ratio of the mass of dry suspended solids divided by the volume of water	$(M.L^{-3})$
t	=	time	(T)
t_{rs}	=	time at which the continuous release at location x_r starts	(T)
t_{re}	=	time at which the continuous release at location x_e ends	(T)
t_s	=	time of pulse input	(T)
T	=	temperature at which the saturated vapour pressure, the solubility and the exchange coefficients in the liquid and gas phases are defined	(θ)
u	=	flow velocity water	$(L.T^{-1})$
w	=	average flow velocity of pore water (i.e. $\ell q/P\varepsilon$)	$(L.T^{-1})$
x	=	downstream distance along ditch axis	(L)
x_0	=	location pulse input	(L)
X_b	=	content of substance sorbed, i.e. the ratio of the mass of substance sorbed divided by the mass of dry bottom material	$(M.M^{-1})$
X_{mp}	=	content of substance sorbed to macrophytes, i.e. the ratio of the mass of substance sorbed divided by the mass of dry macrophytes	$(M.M^{-1})$
X_{ss}	=	content of substance sorbed to suspended solids i.e. the ratio of the mass of substance sorbed divided by the mass of dry suspended solids	$(M.M^{-1})$
y	=	direction perpendicular to the x and z axis	(L)
z	=	depth under the water-sediment interface	(L)

α	=	weight factor for space, applied in the numerical solution method	
β	=	weight factor for space, applied in the numerical solution method	(1)
β	=	arctan $(1/s_1)$	(1)
δ	=	Dirac delta function	(T^{-1} or L^{-1})
ε	=	volume fraction of pore water, i.e. volume of liquid divided by volume of bottom material	(1)
$\zeta(t)$	=	release time function defined by: 0 for $t < t_{rs}$ or $t > t_{re}$ 1 for $t_{rs} \leq t \leq t_{re}$	(1)
θ	=	weight factor for time, applied in the numerical solution method	(1)
λ	=	tortuosity factor, i.e. ratio of surface area of bottom material to liquid phase	(1)
ρ_b	=	bulk density of dry bottom material, i.e. volumic mass of dry bottom material	($M.L^{-3}$)

Annex 1 The elements *LOD, LDD, LBD, ROD, RDD, RBD* and *RV* from Eq.(6.26) for the grid points *m* +1 up to *ebt* -1 inclusive in the end buffer of the water subsystem

$$LOD = -\frac{Q_{i-1/2}^{j+1/2}}{\Delta x_i}\,(1-\theta)\,\left(1-NWW_{i-1/2}\right)\,\Delta t\,\left(1 \cancel{+ ss\,K_F \left(\frac{c_{i-1}^{j+1}}{c_e}\right)^{n_s-1}}\right)$$

$$-\frac{2A_{i-1/2}^{j+1/2}\,E_{x\,i-1/2}^{\;j+1/2}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}\,(1-\theta)\,\Delta t\,\left(1 \cancel{+ ss\,K_F \left(\frac{c_{i-1}^{j+1}}{c_e}\right)^{n_s-1}}\right)$$

$$LDD = -\frac{Q_{i-1/2}^{j+1/2}}{\Delta x_i}\,(1-\theta)\,NWW_{i-1/2}\,\Delta t\,\left(1 \cancel{+ ss\,K_F \left(\frac{c_{i}^{j+1}}{c_e}\right)^{n_s-1}}\right)$$

$$+\frac{Q_{i+1/2}^{j+1/2}}{\Delta x_i}\,(1-\theta)\,\left(1-NWW_{i+1/2}\right)\,\Delta t\,\left(1 \cancel{+ ss\,K_F \left(\frac{c_{i}^{j+1}}{c_e}\right)^{n_s-1}}\right)$$

$$+\frac{2A_{i-1/2}^{j+1/2}\,E_{x\,i-1/2}^{\;j+1/2}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}\,(1-\theta)\,\Delta t\,\left(1 \cancel{+ ss\,K_F \left(\frac{c_{i}^{j+1}}{c_e}\right)^{n_s-1}}\right)$$

$$+\frac{2A_{i+1/2}^{j+1/2}\,E_{x\,i+1/2}^{\;j+1/2}}{\Delta x_i\left(\Delta x_{i+1}+\Delta x_i\right)}\,(1-\theta)\,\Delta t\,\left(1 \cancel{+ ss\,K_F \left(\frac{c_{i}^{j+1}}{c_e}\right)^{n_s-1}}\right)\;\cdot$$

$$+\left(1 + \frac{DW \cdot P_{z=0}}{A_i^{j+1}} K_{\mathrm{mp}} + ss\, K_{\mathrm{F}} \left(\frac{c_i^{j+1}}{c_{\mathrm{e}}}\right)^{n_m - 1}\right)\left(A_i^{j+1} + k\, A_i^{j+½} (1-\theta)\, \Delta t\right)$$

$$+ k_{\mathrm{t},1}\, O_{x\,i}^{\,j+½} (1-\theta)\, \Delta t + \frac{\rho}{P_{z=0}}\, q_i^{j+½}\, P_{x\,i}^{\,j} (1-\theta)\, \Delta t$$

$$+ \frac{2\varepsilon_i^j\, D_{\mathrm{lb}}\, P_{x\,i}^{\,j}}{\Delta z_1} (1-\theta)\, \Delta t$$

$$LBD = \frac{Q_{i+½}^{j+½}}{\Delta x_i} (1-\theta)\, NWW_{i+½}\, \Delta t \left(1 + ss\, K_{\mathrm{F}} \left(\frac{c_{i+1}^{j+1}}{c_{\mathrm{e}}}\right)^{n_m - 1}\right)$$

$$- \frac{2A_{i+½}^{j+½}\, E_{x\,i+½}^{\;j+½}}{\Delta x_i \left(\Delta x_{i+1} + \Delta x_i\right)} (1-\theta)\, \Delta t \left(1 + ss\, K_{\mathrm{F}} \left(\frac{c_{i+1}^{j+1}}{c_{\mathrm{e}}}\right)^{n_m - 1}\right)$$

$$ROD = \frac{Q_{i-½}^{j+½}}{\Delta x_i}\, \theta \left(1 - NWW_{i-½}\right) \Delta t \left(1 + ss\, K_{\mathrm{F}} \left(\frac{c_{i-1}^{j}}{c_{\mathrm{e}}}\right)^{n_m - 1}\right)$$

$$+ \frac{2A_{i-½}^{j+½}\, E_{x\,i-½}^{\;j+½}}{\Delta x_i \left(\Delta x_i + \Delta x_{i-1}\right)}\, \theta \cdot \Delta t \left(1 + ss\, K_{\mathrm{F}} \left(\frac{c_{i-1}^{j}}{c_{\mathrm{e}}}\right)^{n_m - 1}\right)$$

$$RDD = \frac{Q_{i-½}^{j+½}}{\Delta x_i}\ \theta \cdot NWW_{i-½}\ \Delta t \left(1 + ss\ K_F \left(\frac{c_i^j}{c_e}\right)^{n_{ss}-1}\right)$$

$$- \frac{Q_{i+½}^{j+½}}{\Delta x_i}\ \theta \cdot \left(1 - NWW_{i+½}\right)\ \Delta t \left(1 + ss\ K_F \left(\frac{c_i^j}{c_e}\right)^{n_{ss}-1}\right)$$

$$- \frac{2A_{i-½}^{j+½}\ E_{x\ i-½}^{\ j+½}}{\Delta x_i\ \left(\Delta x_i + \Delta x_{i-1}\right)}\ \theta \cdot \Delta t \left(1 + ss\ K_F \left(\frac{c_i^j}{c_e}\right)^{n_{ss}-1}\right)$$

$$- \frac{2A_{i+½}^{j+½}\ E_{x\ i+½}^{\ j+½}}{\Delta x_i\ \left(\Delta x_{i+1} + \Delta x_i\right)}\ \theta \cdot \Delta t \left(1 + ss\ K_F \left(\frac{c_i^j}{c_e}\right)^{n_{ss}-1}\right)$$

$$+ \left(1 + \frac{DW\ P_{z=0}}{A_i^j}\ K_{mp} + ss\ K_F \left(\frac{c_i^j}{c_e}\right)^{n_{ss}-1}\right) \left(A_i^j + kA_i^{j+½}\ \theta\ \Delta t\right)$$

$$+ k_{t,1}\ Q_{x\ i}^{\ j+½}\ \theta\ \Delta t + \frac{\ell}{P_{z=0}}\ q_i^{j+½}\ P_{x\ i}^{\ j}\ \theta\ \Delta t$$

$$+ \frac{2\varepsilon_i^j\ D_{lb}\ P_{x\ i}^{\ j}}{\Delta z_1}\ \theta\ \Delta t$$

$$RBD = -\frac{Q_{i+\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i}\,\theta \cdot NWW_{i+\frac{1}{2}}\,\Delta t \left(1 \cancel{+ SS\, K_F \left(\frac{c_{i+1}^{j}}{c_e}\right)^{n_a - 1}}\right)$$

$$+\frac{2A_{i+\frac{1}{2}}^{j+\frac{1}{2}}\,E_{x\,i+\frac{1}{2}}^{\;j+\frac{1}{2}}}{\Delta x_i\left(\Delta x_{i+1} + \Delta x_i\right)}\,\theta \cdot \Delta t \left(1 \cancel{+ SS\, K_F \left(\frac{c_{i+1}^{j}}{c_e}\right)^{n_a - 1}}\right)$$

$$RV = \cancel{k_{t,1}\, Q_{x\,i}^{\;j+\frac{1}{2}}\,\frac{c_{a\,i}^{\;j+\frac{1}{2}}}{K_H}\,\Delta t + 2\,\frac{\varepsilon_i^j\, D_{lb\,i}^{\;j+\frac{1}{2}}\, P_{x\,i}^{\;j}}{\Delta z_1}\, c_{lb\,i,\,k=1}^{\;j}}\,\Delta t$$

So, the right-hand vector RV does not appear anymore.

Annex 2 The elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD*, *RBD* and *RV* from Eq. (6.26) for the grid point *eb* = *ebt* of the end buffer of the water subsystem. The flow velocity *u* is positive

$$LOD = -\frac{Q_{i-\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i}(1-\theta)\left(1-NWW_{i-\frac{1}{2}}\right)\Delta t\left(1+ss\,K_F\left(\frac{C_{i-1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$-\frac{2A_{i-\frac{1}{2}}^{j+\frac{1}{2}}\,E_{x\,i-\frac{1}{2}}^{\;j+\frac{1}{2}}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}(1-\theta)\,\Delta t\left(1+ss\,K_F\left(\frac{C_{i-1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$LDD = -\frac{Q_{i-\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i}(1-\theta)\,NWW_{i-\frac{1}{2}}\,\Delta t\left(1+ss\,K_F\left(\frac{C_{i}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+\frac{Q_{i+\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i}(1-\theta)\left(1-NWW_{i+\frac{1}{2}}\right)\Delta t\left(1+ss\,K_F\left(\frac{C_{i}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+\frac{2A_{i-\frac{1}{2}}^{j+\frac{1}{2}}\,E_{x\,i-\frac{1}{2}}^{\;j+\frac{1}{2}}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}(1-\theta)\,\Delta t\left(1+ss\,K_F\left(\frac{C_{i}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+\frac{2A_{i+½}^{j+½}\,E_{x\;i+½}^{\;j+½}}{\Delta x_i\left(\Delta x_{i+1}+\Delta x_i\right)}(1-\theta)\,\Delta t\left(1+ss\;K_F\left(\frac{c_i^{j+1}}{c_e}\right)^{n_a-1}\right)$$

$$+\left(1+\frac{DW\cdot P_{z=0}}{A_i^{j+1}}K_{mp}+ss\;K_F\left(\frac{c_i^{j+1}}{c_e}\right)^{n_a-1}\right)\left(A_i^{j+1}+k\;A_i^{j+½}(1-\theta)\,\Delta t\right)$$

$$+k_{t,1}\,O_{x\;i}^{\;j+½}(1-\theta)\,\Delta t+\frac{\ell}{P_{z=0}}q_i^{j+½}\,P_{x\;i}^{\;j}(1-\theta)\,\Delta t$$

$$+\frac{2\varepsilon_i^j\,D_{lb}\,P_{x\;i}^{\;j}}{\Delta z_1}(1-\theta)\,\Delta t$$

$$LBD=\frac{Q_{i+½}^{j+½}}{\Delta x_i}(1-\theta)\,NWW_{i+½}\,\Delta t\left(1+ss\;K_F\left(\frac{c_{i+1}^{j+1}}{c_e}\right)^{n_a-1}\right)$$

$$-\frac{2A_{i+½}^{j+½}\,E_{x\;i+½}^{\;j+½}}{\Delta x_i\left(\Delta x_{i+1}+\Delta x_i\right)}(1-\theta)\,\Delta t\left(1+ss\;K_F\left(\frac{c_{i+1}^{j+1}}{c_e}\right)^{n_a-1}\right)$$

$$ROD=\frac{Q_{i-½}^{j+½}}{\Delta x_i}\;\theta\left(1-NWW_{i-½}\right)\Delta t\left(1+ss\;K_F\left(\frac{c_{i-1}^{j}}{c_e}\right)^{n_a-1}\right)$$

$$+ \frac{2A_{i-\frac{1}{2}}^{j+\frac{1}{2}} \, E_{x\;i-\frac{1}{2}}^{\;j+\frac{1}{2}}}{\Delta x_i \left(\Delta x_i + \Delta x_{i-1}\right)} \, \theta \cdot \Delta t \left(1 + ss \; K_{\mathrm{F}} \left(\frac{c_{i-1}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$RDD = \frac{Q_{i-\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i} \, \theta \cdot NWW_{i-\frac{1}{2}} \; \Delta t \left(1 + ss \; K_{\mathrm{F}} \left(\frac{c_{i}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$- \frac{Q_{i+\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i} \, \theta \cdot \left(1 - NWW_{i+\frac{1}{2}}\right) \; \Delta t \left(1 + ss \; K_{\mathrm{F}} \left(\frac{c_{i}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$- \frac{2A_{i-\frac{1}{2}}^{j+\frac{1}{2}} \, E_{x\;i-\frac{1}{2}}^{\;j+\frac{1}{2}}}{\Delta x_i \left(\Delta x_i + \Delta x_{i-1}\right)} \, \theta \cdot \Delta t \left(1 + ss \; K_{\mathrm{F}} \left(\frac{c_{i}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$- \frac{2A_{i+\frac{1}{2}}^{j+\frac{1}{2}} \, E_{x\;i+\frac{1}{2}}^{\;j+\frac{1}{2}}}{\Delta x_i \left(\Delta x_{i+1} + \Delta x_{i}\right)} \, \theta \cdot \Delta t \left(1 + ss \; K_{\mathrm{F}} \left(\frac{c_{i}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$+ \left(1 + \frac{DW \; P_{z=0}}{A_i^{\,j}} \, K_{\mathrm{mp}} + ss \; K_{\mathrm{F}} \left(\frac{c_{i}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right) \left(A_i^{\,j} - kA_i^{\,j+\frac{1}{2}} \; \theta \; \Delta t\right)$$

$$- k_{\mathrm{t},1} \; O_{x\;i}^{\;j+\frac{1}{2}} \; \theta \; \Delta t - \frac{\ell}{P_{z=0}} \; q_i^{\,j+\frac{1}{2}} \; P_{x\;i}^{\;j} \; \theta \; \Delta t$$

$$- \frac{2\varepsilon_i^j D_{lb} P_{x\,i}^{\;j}}{\Delta z_1} \theta \, \Delta t$$

$$RBD = - \frac{Q_{i+½}^{j+½}}{\Delta x_i} \theta \cdot NWW_{i+½} \, \Delta t \left(1 + ss \, K_F \left(\frac{c_{i+1}^j}{c_e} \right)^{n_w - 1} \right)$$

$$+ \frac{2A_{i+½}^{j+½} \, E_{x\,i+½}^{\;j+½}}{\Delta x_i \left(\Delta x_{i+1} + \Delta x_i \right)} \theta \cdot \Delta t \left(1 + ss \, K_F \left(\frac{c_{i+1}^j}{c_e} \right)^{n_w - 1} \right)$$

$$RV = k_{t,1} \, O_{x\,i}^{\;j+½} \frac{c_{a\,i}^{\;j+½}}{K_H} \Delta t + 2 \frac{\varepsilon_i^j D_{lb} P_{x\,i}^{\;j}}{\Delta z_1} c_{lb\;i,k=1}^{\;j} \, \Delta t$$

So, the elements *LBD*, *RBD* and *RV* have disappeared in the segment *ebt* in the case of a positive flow velocity.

Annex 3 The elements *LOD, LDD, LBD, ROD, RDD, RBD* and *RV* from Eq. (6.26) for the grid point *eb* = *ebt* of the end buffer of the water subsystem. The flow velocity *u* is negative

$$LOD = -\frac{Q_{i-\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i}(1-\theta)\left(1-NWW_{i-\frac{1}{2}}\right)\Delta t\left(1+ss\,K_F\left(\frac{c_{i-1}^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$-\frac{2A_{i-\frac{1}{2}}^{j+\frac{1}{2}}\,E_{x\,i-\frac{1}{2}}^{\;j+\frac{1}{2}}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}(1-\theta)\,\Delta t\left(1+ss\,K_F\left(\frac{c_{i-1}^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$LDD = -\frac{Q_{i-\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i}(1-\theta)\,NWW_{i-\frac{1}{2}}\,\Delta t\left(1+ss\,K_F\left(\frac{c_i^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$+\frac{Q_{i+\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i}(1-\theta)\left(1-NWW_{i+\frac{1}{2}}\right)\Delta t\left(1+ss\,K_F\left(\frac{c_i^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$+\frac{2A_{i-\frac{1}{2}}^{j+\frac{1}{2}}\,E_{x\,i-\frac{1}{2}}^{\;j+\frac{1}{2}}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}(1-\theta)\,\Delta t\left(1+ss\,K_F\left(\frac{c_i^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$+\frac{2A_{i+½}^{j+½}\,E_{x\,i+½}^{\;j+½}}{\Delta x_i\left(\Delta x_{i+1}+\Delta x_i\right)}(1-\theta)\,\Delta t\left(1+ss\,K_F\left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+\left(1+\frac{DW\cdot P_{z=0}}{A_i^{j+1}}K_{mp}+ss\,K_F\left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right)\left(A_i^{j+1}+k\,A_i^{j+½}(1-\theta)\,\Delta t\right)$$

$$+k_{t,1}\,O_{x\,i}^{\;j+½}(1-\theta)\,\Delta t+\frac{\ell}{P_{z=0}}\,q_i^{j+½}\,P_{x\,i}^{\;j}(1-\theta)\,\Delta t$$

$$+\frac{2\varepsilon_i^j\,D_{lb}\,P_{x\,1}^{\;j}}{\Delta z_1}(1-\theta)\,\Delta t$$

$$LBD=\frac{Q_{i+½}^{j+½}}{\Delta x_i}(1-\theta)\,NWW_{i+½}\,\Delta t\left(1+ss\,K_F\left(\frac{c_{i+1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+\frac{2A_{i+½}^{j+½}\,E_{x\,i+½}^{\;j+½}}{\Delta x_i\left(\Delta x_{i+1}+\Delta x_i\right)}(1-\theta)\,\Delta t\left(1+ss\,K_F\left(\frac{c_{i+1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$ROD=\frac{Q_{i-½}^{j+½}}{\Delta x_i}\,\theta\left(1-NWW_{i-½}\right)\Delta t\left(1+ss\,K_F\left(\frac{c_{i-1}^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+\frac{2A^{j+\frac{1}{2}}_{i-\frac{1}{2}}\,E_x{}^{j+\frac{1}{2}}_{i-\frac{1}{2}}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}\,\theta\cdot\Delta t\left(1+ss\,K_F\left(\frac{c^j_{i-1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$RDD=\frac{Q^{j+\frac{1}{2}}_{i-\frac{1}{2}}}{\Delta x_i}\,\theta\cdot NWW_{i-\frac{1}{2}}\,\Delta t\left(1+ss\,K_F\left(\frac{c^j_i}{c_e}\right)^{n_{ss}-1}\right)$$

$$-\frac{Q^{j+\frac{1}{2}}_{i+\frac{1}{2}}}{\Delta x_i}\,\theta\cdot\left(1-NWW_{i+\frac{1}{2}}\right)\,\Delta t\left(1+ss\,K_F\left(\frac{c^j_i}{c_e}\right)^{n_{ss}-1}\right)$$

$$-\frac{2A^{j+\frac{1}{2}}_{i-\frac{1}{2}}\,E_x{}^{j+\frac{1}{2}}_{i-\frac{1}{2}}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}\,\theta\cdot\Delta t\left(1+ss\,K_F\left(\frac{c^j_i}{c_e}\right)^{n_{ss}-1}\right)$$

$$-\frac{2A^{j+\frac{1}{2}}_{i+\frac{1}{2}}\,E_x{}^{j+\frac{1}{2}}_{i+\frac{1}{2}}}{\Delta x_i\left(\Delta x_{i+1}+\Delta x_i\right)}\,\theta\cdot\Delta t\left(1+ss\,K_F\left(\frac{c^j_i}{c_e}\right)^{n_{ss}-1}\right)$$

$$+\left(1+\frac{DW\,P_{z=0}}{A^j_i}\,K_{mp}+ss\,K_F\left(\frac{c^j_i}{c_e}\right)^{n_{ss}-1}\right)\left(A^j_i-kA^{j+\frac{1}{2}}_i\,\theta\,\Delta t\right)$$

$$-k_{t,1}\,O_{x\,i}{}^{j+\frac{1}{2}}\,\theta\,\Delta t-\frac{\ell}{P_{z=0}}\,q^{j+\frac{1}{2}}_i\,P_{x\,i}{}^{j}\,\theta\,\Delta t$$

$$\frac{2\varepsilon_i^j D_{lb} P_{x\,i}^{\,j}}{\Delta z_1} \theta \, \Delta t$$

$$RBD = \frac{Q_{i+½}^{j+½}}{\Delta x_i} \theta \cdot NWW_{i+½} \, \Delta t \left(1 + ss \, K_F \left(\frac{c_{i+1}^{j}}{c_e} \right)^{n_w - 1} \right)$$

$$+ \frac{2 A_{i+½}^{j+½} E_{x\,i+½}^{\,j+½}}{\Delta x_i \left(\Delta x_{i+1} + \Delta x_i \right)} \theta \cdot \Delta t \left(1 + ss \, K_F \left(\frac{c_{i+1}^{j}}{c_e} \right)^{n_w - 1} \right)$$

$$RV = k_{l,1} \, O_{x\,i}^{\,j+½} \frac{c_{a\,i}^{\,j+½}}{K_H} \Delta t + 2 \frac{\varepsilon_i^j D_{lb\,i}^{\,j+½} P_{x\,i}^{\,j}}{\Delta z_1} c_{lb\,i,\,k=1}^{\,j} \, \Delta t$$

So, the elements *LBD*, *RBD* and *RV* have disappeared in the segment *ebt* in the case of a negative flow velocity.

Annex 4 The elements *LOD, LDD, LBD, ROD, RDD, RBD* and *RV* from Eq. (6.26) for the grid point $i = 1$ in the water subsystem in the case of a positive flow direction

In the case of downward waterflow in the sediment subsystem:

$$LOD = \frac{Q_{i-½}^{j+½}}{\Delta x_i} (1-\theta) \left(1 - NWW_{i-½}\right) \Delta t \left(1 + ss\, K_F \left(\frac{c_{i-1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$\frac{2A_{i-½}^{j+½}\, E_{x\, i-½}^{\, j+½}}{\Delta x_i \left(\Delta x_i + \Delta x_{i-1}\right)} (1-\theta)\, \Delta t \left(1 + ss\, K_F \left(\frac{c_{i-1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$LDD = \frac{Q_{i-½}^{j+½}}{\Delta x_i} (1-\theta)\, NWW_{i-½}\, \Delta t \left(1 + ss\, K_F \left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+ \frac{Q_{i+½}^{j+½}}{\Delta x_i} (1-\theta) \left(1 - NWW_{i+½}\right) \Delta t \left(1 + ss\, K_F \left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+ \frac{2A_{i-½}^{j+½}\, E_{x\, i-½}^{\, j+½}}{\Delta x_i \left(\Delta x_i + \Delta x_{i-1}\right)} (1-\theta)\, \Delta t \left(1 + ss\, K_F \left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+ \frac{2A_{i+½}^{j+½}\, E_{x\, i+½}^{\, j+½}}{\Delta x_i \left(\Delta x_{i+1} + \Delta x_i\right)} (1-\theta)\, \Delta t \left(1 + ss\, K_F \left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+ \left(1 + \frac{DW \cdot P_{z=0}}{A_i^{j+1}} K_{mp} + ss\, K_F \left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right)\left(A_i^{j+1} + k\, A_i^{j+½}(1 - \theta)\, \Delta t\right)$$

$$+ k_{t,1}\, O_{x\,i}^{\,j+½}\, (1 - \theta)\, \Delta t + \frac{\ell}{P_{z=0}}\, q_i^{j+½}\, P_{x\,i}^{\,j}(1 - \theta)\, \Delta t$$

$$+ \frac{2\varepsilon_i^j\, D_{lb}\, P_{x\,i}^{\,j}}{\Delta z_1} (1 - \theta)\, \Delta t$$

$$LBD = \frac{Q_{i+½}^{j+½}}{\Delta x_i} (1-\theta)\, NWW_{i+½}\, \Delta t \left(1 + ss\, K_F \left(\frac{c_{i+1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$- \frac{2A_{i+½}^{j+½}\, E_{x\,i+½}^{\,j+½}}{\Delta x_i\, \left(\Delta x_{i+1} + \Delta x_i\right)} (1-\theta)\, \Delta t \left(1 + ss\, K_F \left(\frac{c_{i+1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$ROD = \frac{Q_{i-½}^{j+½}}{\Delta x_i}\, \theta\, (1 - NWW_{i-½}\,)\, \Delta t \left(1 + ss\, K_F \left(\frac{c_{i-1}^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+ \frac{2A_{i-½}^{j+½}\, E_{x\,i-½}^{\,j+½}}{\Delta x_i\, \left(\Delta x_i + \Delta x_{i-1}\right)}\, \theta \cdot \Delta t \left(1 + ss\, K_F \left(\frac{c_{i-1}^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$RDD = \frac{Q^{j+½}_{i-½}}{\Delta x_i}\,\theta \cdot NWW_{i-½}\,\Delta t\left(1 + ss\,K_{\mathrm{F}}\left(\frac{c_i^j}{c_{\mathrm{e}}}\right)^{n_{m}-1}\right)$$

$$-\,\frac{Q^{j+½}_{i+½}}{\Delta x_i}\,\theta \cdot \left(1 - NWW_{i+½}\right)\Delta t\left(1 + ss\,K_{\mathrm{F}}\left(\frac{c_i^j}{c_{\mathrm{e}}}\right)^{n_{m}-1}\right)$$

$$-\,\frac{2A^{j+½}_{i-½}\,E_{x\,i-½}^{\,j+½}}{\Delta x_i\left(\Delta x_i + \Delta x_{i-1}\right)}\,\theta \cdot \Delta t\left(1 + ss\,K_{\mathrm{F}}\left(\frac{c_i^j}{c_{\mathrm{e}}}\right)^{n_{m}-1}\right)$$

$$-\,\frac{2A^{j+½}_{i+½}\,E_{x\,i+½}^{\,j+½}}{\Delta x_i\left(\Delta x_{i+1} + \Delta x_{i}\right)}\,\theta \cdot \Delta t\left(1 + ss\,K_{\mathrm{F}}\left(\frac{c_i^j}{c_{\mathrm{e}}}\right)^{n_{m}-1}\right)$$

$$+\left(1 + \frac{DW\,P_{z=0\,i}^{\;j}}{A_i^j}\,K_{\mathrm{mp}} + ss\,K_{\mathrm{F}}\left(\frac{c_i^j}{c_{\mathrm{e}}}\right)^{n_{m}-1}\right)\left(A_i^j - kA_i^{j+½}\,\theta\,\Delta t\right)$$

$$-\,k_{\mathrm{t},1}\,O_{x\,i}^{\;j+½}\,\theta\,\Delta t \;-\; \frac{\ell}{P_{z=0}}\,q_i^{j+½}\,P_{x\,i}^{\;j}\,\theta\,\Delta t$$

$$-\,\frac{2\varepsilon_i^j\,D_{\mathrm{lb}}\,P_{x\,i}^{\;j}}{\Delta z_1}\,\theta\,\Delta t$$

$$RBD = -\frac{Q_{i+\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i}\,\theta \cdot NWW_{i+\frac{1}{2}}\,\Delta t\left(1 + ss\,K_F\left(\frac{c_{i+1}^{j}}{c_e}\right)^{n_w - 1}\right)$$

$$+\frac{2A_{i+\frac{1}{2}}^{j+\frac{1}{2}}\,E_{x\,i+\frac{1}{2}}^{\;j+\frac{1}{2}}}{\Delta x_i\,(\Delta x_{i+1} + \Delta x_i)}\,\theta \cdot \Delta t\left(1 + ss\,K_F\left(\frac{c_{i+1}^{j}}{c_e}\right)^{n_w - 1}\right)$$

$$RV = k_{t,1}\,O_{x\,i}^{\;j+\frac{1}{2}}\,\frac{c_{a\,i}^{\;j+\frac{1}{2}}}{K_H}\Delta t + 2\frac{\varepsilon_i^j\,D_{lb\,i}^{\;j+\frac{1}{2}}\,P_{x\,i}^{\;j}}{\Delta z_1}c_{lb\,i,k=1}^{\;j}\,\Delta t$$

In the case of upward water flow in the sediment subsystem the right-hand vector contains an additional term:

$$-\frac{\ell}{P_{z=0}}\,q_i^{j+\frac{1}{2}}\,c_{lb\,i,k=1}^{\;j}\,P_{x\,i}^{\;j+\frac{1}{2}}\,\Delta t$$

and the penultimate terms containing $q_i^{j+\frac{1}{2}}$ in the *LDD* and *RDD* elements disappear.

So in all cases ($q < 0$, $q > 0$) the elements *LOD* and *ROD* have disappeared in the first segment; *LBD*, *RBD* and *RV* remain unchanged in this case of a positive flow velocity and an incoming flux of zero in the water subsystem.

Annex 5 The elements *LOD, LDD, LBD, ROD, RDD, RBD* and *RV* from Eq. (6.26) for the grid point *fb* = 2 up to *fb* = *fbt* inclusive of the front buffer of the water subsystem. The flow velocity alternates between being positive and negative

$$LOD = -\frac{Q_{i-\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i}(1-\theta)\left(1-NWW_{i-\frac{1}{2}}\right)\Delta t\left(1+ss\,K_F\left(\frac{c_{i-1}^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$-\frac{2A_{i-\frac{1}{2}}^{j+\frac{1}{2}}\,E_{x\,i-\frac{1}{2}}^{\;j+\frac{1}{2}}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}(1-\theta)\,\Delta t\left(1+ss\,K_F\left(\frac{c_{i-1}^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$LDD = -\frac{Q_{i-\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i}(1-\theta)\,NWW_{i-\frac{1}{2}}\,\Delta t\left(1+ss\,K_F\left(\frac{c_{i}^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$+\frac{Q_{i+\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i}(1-\theta)\left(1-NWW_{i+\frac{1}{2}}\right)\Delta t\left(1+ss\,K_F\left(\frac{c_{i}^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$+\frac{2A_{i-\frac{1}{2}}^{j+\frac{1}{2}}\,E_{x\,i-\frac{1}{2}}^{\;j+\frac{1}{2}}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}(1-\theta)\,\Delta t\left(1+ss\,K_F\left(\frac{c_{i}^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$+\ \frac{2A_{i+½}^{j+½}\ E_{x\ i+½}^{\ j+½}}{\Delta x_i\left(\Delta x_{i+1}+\Delta x_i\right)}\ (1-\theta)\ \Delta t\left(1+ss\ K_{\mathrm{F}}\left(\frac{c_i^{j+1}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$+\left(1+\frac{DW\cdot P_{z=0}}{A_i^{j+1}}\ K_{\mathrm{mp}}+ss\ K_{\mathrm{F}}\left(\frac{c_i^{j+1}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)\left(A_i^{j+1}+k\ A_i^{j+½}\ (1-\theta)\ \Delta t\right)$$

$$+\ k_{t,1}\ O_{x\ i}^{\ j+½}\ (1-\theta)\ \Delta t+\frac{\ell}{P_{z=0}}\ q_i^{j+½}\ P_{x\ i}^{\ j}(1-\theta)\ \Delta t$$

$$+\ \frac{2\varepsilon_i^j\ D_{½}\ P_{x\ i}^{\ j}}{\Delta z_1}\ (1-\theta)\ \Delta t$$

$$LBD=\frac{Q_{i+½}^{j+½}}{\Delta x_i}\ (1-\theta)\ NWW_{i+½}\ \Delta t\left(1+ss\ K_{\mathrm{F}}\left(\frac{c_{i+1}^{j+1}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$-\ \frac{2A_{i+½}^{j+½}\ E_{x\ i+½}^{\ j+½}}{\Delta x_i\left(\Delta x_{i+1}+\Delta x_i\right)}\ (1-\theta)\ \Delta t\left(1+ss\ K_{\mathrm{F}}\left(\frac{c_{i+1}^{j+1}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$ROD=\frac{Q_{i-½}^{j+½}}{\Delta x_i}\ \theta\left(1-NWW_{i-½}\right)\ \Delta t\left(1+ss\ K_{\mathrm{F}}\left(\frac{c_{i-1}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$+ \frac{2A_{i-½}^{j+½}\, E_{x\; i-½}^{\;j+½}}{\Delta x_i \left(\Delta x_i + \Delta x_{i-1}\right)}\; \theta \cdot \Delta t \left(1 + ss\; K_F \left(\frac{c_{i-1}^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$RDD = \frac{Q_{i-½}^{j+½}}{\Delta x_i}\; \theta \cdot NWW_{i-½}\; \Delta t \left(1 + ss\; K_F \left(\frac{c_i^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$- \frac{Q_{i+½}^{j+½}}{\Delta x_i}\; \theta \cdot \left(1 - NWW_{i+½}\right)\; \Delta t \left(1 + ss\; K_F \left(\frac{c_i^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$- \frac{2A_{i-½}^{j+½}\, E_{x\; i-½}^{\;j+½}}{\Delta x_i \left(\Delta x_i + \Delta x_{i-1}\right)}\; \theta \cdot \Delta t \left(1 + ss\; K_F \left(\frac{c_i^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$- \frac{2A_{i+½}^{j+½}\, E_{x\; i+½}^{\;j+½}}{\Delta x_i \left(\Delta x_{i+1} + \Delta x_i\right)}\; \theta \cdot \Delta t \left(1 + ss\; K_F \left(\frac{c_i^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+ \left(1 + \frac{DW\, P_{z=0}}{A_i^{j}}\, K_{mp} + ss\; K_F \left(\frac{c_i^{j}}{c_e}\right)^{n_{ss}-1}\right) \left(A_i^{j} - k A_i^{j+½}\, \theta\, \Delta t\right)$$

$$- k_{t,1}\, O_{x\,i}^{\;j+½}\, \theta\, \Delta t - \frac{\ell}{P_{z=0}}\, q_i^{j+½}\, P_{x\,i}^{\;j}\, \theta\, \Delta t$$

$$\frac{2\varepsilon_i^j \, D_{\mathrm{lb}} \, P_{x\,i}^{\;j}}{\Delta z_1} \, \theta \, \Delta t$$

$$RBD = -\frac{Q_{i+½}^{j+½}}{\Delta x_i} \, \theta \cdot NWW_{i+½} \, \Delta t \left(1 + ss \, K_{\mathrm{F}} \left(\frac{c_{i+1}^{j}}{c_e} \right)^{n_m - 1} \right)$$

$$+ \frac{2A_{i+½}^{j+½} \, E_{x\;i+½}^{\;j+½}}{\Delta x_i \left(\Delta x_{i+1} + \Delta x_i \right)} \, \theta \cdot \Delta t \left(1 + ss \, K_{\mathrm{F}} \left(\frac{c_{i+1}^{j}}{c_e} \right)^{n_m - 1} \right)$$

$$RV = k_{t,\,1} \, O_{x\,i}^{\;j+½} \, \frac{c_{a\,i}^{\;j+½}}{K_{\mathrm{H}}} \Delta t + 2 \frac{\varepsilon_i^j \, D_{\mathrm{lb}} \, P_{x\,i}^{\;j}}{\Delta z_1} c_{\mathrm{lb}\;i,\,k=1}^{\;j} \, \Delta t$$

So, the vector *RV* does not appear anymore.

Annex 6 The elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD*, *RBD* and *RV* from Eq. (6.26) for the grid point *fb* = 1 in the case of a negative flow velocity in the situation of alternating positive and negative flow velocities in the water layer

$$LOD = -\frac{Q_{i-½}^{j+½}}{\Delta x_i}(1-\theta)\left(1-NWW_{i-½}\right)\Delta t\left(1 + ss\,K_F\left(\frac{c_{i-1}^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$-\frac{2A_{i-½}^{j+½}\,E_{x\,i-½}^{\;j+½}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}(1-\theta)\,\Delta t\left(1 + ss\,K_F\left(\frac{c_{i-1}^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$LDD = -\frac{Q_{i-½}^{j+½}}{\Delta x_i}(1-\theta)\,NWW_{i-½}\,\Delta t\left(1 + ss\,K_F\left(\frac{c_i^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$+\frac{Q_{i+½}^{j+½}}{\Delta x_i}(1-\theta)\left(1-NWW_{i+½}\right)\Delta t\left(1 + ss\,K_F\left(\frac{c_i^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$+\frac{2A_{i-½}^{j+½}\,E_{x\,i-½}^{\;j+½}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}(1-\theta)\,\Delta t\left(1 + ss\,K_F\left(\frac{c_i^{j+1}}{c_e}\right)^{n_w-1}\right)$$

$$+ \frac{2A_{i+½}^{j+½}\, E_{x\,i+½}^{\;j+½}}{\Delta x_i \left(\Delta x_{i+1} + \Delta x_i\right)} (1-\theta)\, \Delta t \left(1 + ss\; K_F \left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+ \left(1 + \frac{DW \cdot P_{z=0}}{A_i^{j+1}} K_{mp} + ss\; K_F \left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right) \left(A_i^{j+1} + k\, A_i^{j+½} (1-\theta)\, \Delta t\right)$$

$$+ k_{1,1}\, O_{x\,i}^{\;j+½} (1-\theta)\, \Delta t + \frac{\ell}{P_{z=0}}\, q_i^{j+½}\, P_{x\,i}^{\;j} (1-\theta)\, \Delta t$$

$$+ \frac{2\varepsilon_i^j\, D_{lb}\, P_{x\,i}^{\;j}}{\Delta z_1} (1-\theta)\, \Delta t$$

$$LBD = \frac{Q_{i+½}^{j+½}}{\Delta x_i} (1-\theta)\; NWW_{i+½}\; \Delta t \left(1 + ss\; K_F \left(\frac{c_{i+1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$- \frac{2A_{i+½}^{j+½}\, E_{x\,i+½}^{\;j+½}}{\Delta x_i \left(\Delta x_{i+1} + \Delta x_i\right)} (1-\theta)\, \Delta t \left(1 + ss\; K_F \left(\frac{c_{i+1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$ROD = \frac{Q_{i-½}^{j+½}}{\Delta x_i}\, \theta \left(1 - NWW_{i-½}\right) \Delta t \left(1 + ss\; K_F \left(\frac{c_{i-1}^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+\frac{2A_{i-½}^{j+½}\, E_{x\,i-½}^{\,j+½}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}\,\theta\cdot\Delta t\left(1+ss\,K_F\left(\frac{c_{i-1}^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$RDD=\frac{Q_{i-½}^{j+½}}{\Delta x_i}\,\theta\, NWW_{i-½}\,\Delta t\left(1+ss\,K_F\left(\frac{c_i^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$-\frac{Q_{i+½}^{j+½}}{\Delta x_i}\,\theta\cdot\left(1-NWW_{i+½}\right)\,\Delta t\left(1+ss\,K_F\left(\frac{c_i^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$-\frac{2A_{i-½}^{j+½}\, E_{x\,i-½}^{\,j+½}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}\,\theta\cdot\Delta t\left(1+ss\,K_F\left(\frac{c_i^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$-\frac{2A_{i+½}^{j+½}\, E_{x\,i+½}^{\,j+½}}{\Delta x_i\left(\Delta x_{i+1}+\Delta x_i\right)}\,\theta\cdot\Delta t\left(1+ss\,K_F\left(\frac{c_i^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+\left(1+\frac{DW\,P_{z=0}}{A_i^{j}}\,K_{mp}+ss\,K_F\left(\frac{c_i^{j}}{c_e}\right)^{n_{ss}-1}\right)\left(A_i^{j}+kA_i^{j+½}\,\theta\,\Delta t\right)$$

$$+k_{t,1}\,O_{x\,i}^{\,j+½}\,\theta\,\Delta t+\frac{\ell}{P_{z=0}}\,q_i^{j+½}\,P_{x\,i}^{\,j}\,\theta\,\Delta t$$

$$\frac{2\varepsilon_i^j D_{1b} P_{x\,i}^j}{\Delta z_1} \theta \, \Delta t$$

$$RBD = - \frac{Q_{i+½}^{j+½}}{\Delta x_i} \theta \cdot NWW_{i+½} \, \Delta t \left(1 + ss \, K_F \left(\frac{c_{i+1}^j}{c_e} \right)^{n_{ss}-1} \right)$$

$$+ \frac{2A_{i+½}^{j+½} \, E_{x\,i+½}^{\;j+½}}{\Delta x_i \left(\Delta x_{i+1} + \Delta x_i \right)} \theta \cdot \Delta t \left(1 + ss \, K_F \left(\frac{c_{i+1}^j}{c_e} \right)^{n_{ss}-1} \right)$$

$$RV = k_{t,1} \, O_{x\,i}^{j+½} \frac{c_{a\,i}^{j+½}}{K_H} \Delta t + 2 \frac{\varepsilon_i^j D_{1b} P_{x\,i}^j}{\Delta z_1} c_{1b\,i,k=1}^{\;j} \, \Delta t$$

The elements *LOD*, *ROD* and *RV* have disappeared in this case.

Annex 7 The elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD*, *RBD* and *RV* from Eq. (6.26) for the grid point *fb* = 1 in the case of a positive flow velocity in the situation of alternating positive and negative flow velocities in the water layer

$$LOD = \frac{Q_{i-\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i} (1-\theta)\left(1-NWW_{i-\frac{1}{2}}\right) \Delta t \left(1 + ss\, K_F \left(\frac{c_{i-1}^{i+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$\frac{2A_{i-\frac{1}{2}}^{j+\frac{1}{2}}\, E_{L,i-\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i \left(\Delta x_i + \Delta x_{i-1}\right)} (1-\theta)\, \Delta t \left(1 + ss\, K_F \left(\frac{c_{i-1}^{i+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$LDD = \frac{Q_{i-\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i} (1-\theta)\, NWW_{i-\frac{1}{2}}\, \Delta t \left(1 + ss\, K_F \left(\frac{c_{i}^{i+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+ \frac{Q_{i+\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i} (1-\theta)\left(1 - NWW_{i+\frac{1}{2}}\right) \Delta t \left(1 + ss\, K_F \left(\frac{c_{i}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+ \frac{2A_{i-\frac{1}{2}}^{j+\frac{1}{2}}\, E_{L,i-\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i \left(\Delta x_i + \Delta x_{i-1}\right)} (1-\theta)\, \Delta t \left(1 + ss\, K_F \left(\frac{c_{i}^{i+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+ \frac{2A_{i+½}^{j+½}\, E_{x\,i+½}^{\;j+½}}{\Delta x_i \left(\Delta x_{i+1} + \Delta x_i\right)} (1-\theta)\,\Delta t \left(1 + ss\,K_F \left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+ \left(1 + \frac{DW \cdot P_{z=0}}{A_i^{j+1}} K_{mp} + ss\,K_F \left(\frac{c_i^{j+1}}{c_e}\right)^{n_{ss}-1}\right)\left(A_i^{j+1} + k\,A_i^{j+½}(1-\theta)\,\Delta t\right)$$

$$+ k_{t,1}\, O_{x\,i}^{\;j+½} (1-\theta)\,\Delta t + \frac{\ell}{P_{z=0}}\, q_i^{j+½}\, P_{x\,i}^{\;j} (1-\theta)\,\Delta t$$

$$+ \frac{2\varepsilon_i^j\, D_{lb}\, P_{x\,i}^{\;j}}{\Delta z_1} (1-\theta)\,\Delta t$$

$$LBD = \frac{Q_{i+½}^{j+½}}{\Delta x_i} (1-\theta)\, NWW_{i+½}\,\Delta t \left(1 + ss\,K_F \left(\frac{c_{i+1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$- \frac{2A_{i+½}^{j+½}\, E_{x\,i+½}^{\;j+½}}{\Delta x_i \left(\Delta x_{i+1} + \Delta x_i\right)} (1-\theta)\,\Delta t \left(1 + ss\,K_F \left(\frac{c_{i+1}^{j+1}}{c_e}\right)^{n_{ss}-1}\right)$$

$$ROD = \frac{Q_{i-½}^{j+½}}{\Delta x_i}\, \theta \left(1 - NWW_{i-½}\right) \Delta t \left(1 + ss\,K_F \left(\frac{c_{i-1}^{j}}{c_e}\right)^{n_{ss}-1}\right)$$

$$+\frac{2A_{i-\frac{1}{2}}^{j+\frac{1}{2}}\,E_{x\,i-\frac{1}{2}}^{\;j+\frac{1}{2}}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}\,\theta\cdot\Delta t\left(1\,+\,ss\;K_{\mathrm{F}}\left(\frac{c_{i-1}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$RDD=\frac{Q_{i-\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i}\,\theta\cdot NWW_{i-\frac{1}{2}}\;\Delta t\left(1\,+\,ss\;K_{\mathrm{F}}\left(\frac{c_{i}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$-\,\frac{Q_{i+\frac{1}{2}}^{j+\frac{1}{2}}}{\Delta x_i}\,\theta\cdot\left(1-NWW_{i+\frac{1}{2}}\right)\;\Delta t\left(1\,+\,ss\;K_{\mathrm{F}}\left(\frac{c_{i}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$\frac{2A_{i-\frac{1}{2}}^{j+\frac{1}{2}}\,E_{x\,i-\frac{1}{2}}^{\;j+\frac{1}{2}}}{\Delta x_i\left(\Delta x_i+\Delta x_{i-1}\right)}\,\theta\cdot\Delta t\left(1\,+\,ss\;K_{\mathrm{F}}\left(\frac{c_{i}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$-\,\frac{2A_{i+\frac{1}{2}}^{j+\frac{1}{2}}\,E_{x\,i+\frac{1}{2}}^{\;j+\frac{1}{2}}}{\Delta x_i\left(\Delta x_{i+1}+\Delta x_{i}\right)}\,\theta\cdot\Delta t\left(1\,+\,ss\;K_{\mathrm{F}}\left(\frac{c_{i}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)$$

$$+\left(1\,+\,\frac{DW\,P_{z=0}}{A_i^{j}}\,K_{\mathrm{mp}}\,+\,ss\;K_{\mathrm{F}}\left(\frac{c_{i}^{j}}{c_{\mathrm{e}}}\right)^{n_{ss}-1}\right)\left(A_i^{j}\,-\,kA_i^{j-\frac{1}{2}}\;\theta\;\Delta t\right)$$

$$-\,k_{t,1}\;O_{x\,i}^{\;j+\frac{1}{2}}\;\theta\;\Delta t\,-\,\frac{\theta}{P_{z=0}}\,q_i^{j+\frac{1}{2}}\;P_{x\,i}^{\;j}\;\theta\;\Delta t$$

$$\frac{2\varepsilon_i^j D_{1b} P_{x\,i}^j}{\Delta z_1} \theta \, \Delta t$$

$$RBD = - \frac{Q_{i+½}^{j+½}}{\Delta x_i} \theta \cdot NWW_{i+½} \, \Delta t \left(1 + ss \, K_F \left(\frac{c_{i+1}^j}{c_e} \right)^{n_w - 1} \right)$$

$$+ \frac{2 A_{i+½}^{j+½} \, E_{x\,i+½}^{j+½}}{\Delta x_i \left(\Delta x_{i+1} + \Delta x_i \right)} \theta \cdot \Delta t \left(1 + ss \, K_F \left(\frac{c_{i+1}^j}{c_e} \right)^{n_w - 1} \right)$$

$$RV = k_{L,1} \, O_{x\,i}^{j+½} \frac{c_{a\,i}^{j+½}}{K_H} \Delta t + 2 \frac{\varepsilon_i^j D_{1b} P_{x\,i}^j}{\Delta z_1} c_{1b\;i,\,k=1}^{j} \, \Delta t$$

So, the elements *LOD*, *ROD* and *RV* have disappeared in segment *fb* = 1 in the case of a positive flow velocity in the situation of alternating positive and negative flow velocities in the water layer.

Annex 8 The elements *LOD, LDD, LBD, ROD, RDD, RBD* and *RV* from Eq. (6.39) for the grid point $k = 1$ of the sediment subsystem. So, this is the upper boundary condition

In the case of downward seepage ($q > 0$):

LOD disappears

$$LDD = \frac{\ell \, q^{j+½}}{\Delta z_1} (1 - \theta) (1 - NWB_{1½}) \frac{\Delta t}{P_1^j} \, sofd_{1½}$$

$$+ \frac{P_{1½}^j \, \varepsilon_{1½}^j \, (E_{\mathrm{lb}}{}_{1½}^{j+½} + D_{\mathrm{lb}}{}_{1½}^{j})}{(½ \, \Delta z_2 + ½ \, \Delta z_1) \, \Delta z_1} (1 - \theta) \frac{\Delta t}{P_1^j}$$

$$+ \frac{P_{½}^j \, \varepsilon_{½}^j \, D_{\mathrm{lb}}{}_{½}^{j}}{½ \, (\Delta z_1)^2} (1 - \theta) \frac{\Delta t}{P_1^j}$$

$$+ \left(\varepsilon_1^j + \rho_{\mathrm{b}}{}_1^j \, K_{\mathrm{F}} \left(\frac{c_{\mathrm{lb}}{}_1^{j+1}}{c_{\mathrm{e}}} \right)^{n_{fr}-1} \right) \left(1 + k_{\mathrm{b}} \, (1 - \theta) \, \Delta t\right)$$

$$LBD = \frac{\ell \cdot q^{j+½}}{\Delta z_1} (1-\theta) \, NWB_{1½} \frac{\Delta t}{P_1^j} \, sofd_{1½} - \frac{P_{1½}^j \varepsilon_{1½}^j \left(E_{\mathrm{lb}}{}_{1½}^{j+½} + D_{\mathrm{lb}}{}_{1½}^{j}\right)}{(½\Delta z_2 + ½\Delta z_1) \, \Delta z_1} (1-\theta) \frac{\Delta t}{P_1^j}$$

ROD disappears

$$RDD = -\frac{\ell \cdot q^{j+½}}{\Delta z_1}\,\theta\,\left(1 - NWB_{1½}\right)\frac{\Delta t}{P_1^j}\,sofd_{1½} - \frac{P_{1½}^j\,\varepsilon_{1½}^j\left(E_{\mathrm{lb}\,1½}^{\;j+½} + D_{\mathrm{lb}\,1½}^{\;j}\right)}{\left(½\Delta z_2 + ½\Delta z_1\right)\Delta z_1}\,\theta\,\frac{\Delta t}{P_1^j}$$

$$-\frac{P_{½}^j\,\varepsilon_{½}^j\,D_{\mathrm{lb}\,½}^{\;j}\,\theta}{½\left(\Delta z_1\right)^2}\,\frac{\Delta t}{P_1^j}$$

$$+\left(\varepsilon_1^j + \rho_{\mathrm{b}\,1}^{\;j}\,K_{\mathrm{F}}\left(\frac{c_{\mathrm{lb}\,1}^{\;j}}{c_{\mathrm{e}}}\right)^{n_{wb}-1}\right)\left(1 - k_b\,\theta\,\Delta t\right)$$

$$RBD = -\frac{\ell \cdot q^{j+½}}{\Delta z_1}\,\theta\,NWB_{1½}\,\frac{\Delta t}{P_1^j}\,sofd_{1½} + \frac{P_{1½}^j\,\varepsilon_{1½}^j\left(E_{\mathrm{lb}\,1½}^{\;j+½} + D_{\mathrm{lb}\,1½}^{\;j}\right)}{\left(½\Delta z_2 + ½\Delta z_1\right)\Delta z_1}\cdot\theta\cdot\frac{\Delta t}{P_1^j}$$

On the right-hand side the following terms appear:

$$c_i^j\left(\frac{\ell \cdot q^{j+½}}{\Delta z_1}\,\theta\cdot\frac{\Delta t}{P_1^j} + \frac{P_{½}^j\,\varepsilon_{½}^j\,D_{\mathrm{lb}\,½}^{\;j}}{½\left(\Delta z_1\right)^2}\,\theta\,\frac{\Delta t}{P_1^j}\right)$$

$$+\,c_i^{j+1}\left(\frac{\ell \cdot q^{j+½}}{\Delta z_1}\,(1-\theta)\,\frac{\Delta t}{P_1^j} + \frac{P_{½}^j\,\varepsilon_{½}^j\,D_{\mathrm{lb}\,½}^{\;j}}{½\left(\Delta z_1\right)^2}\,(1-\theta)\,\frac{\Delta t}{P_1^j}\right)$$

This implies that a right-hand vector is added to Eq. (6.39).

In the case of upward seepage ($q < 0$):

On the right-hand side the following terms appear:

$$c_i^j\left(\frac{P_{½}^j\,\varepsilon_{½}^j\,D_{\mathrm{lb}\,½}^{\;j}}{½\left(\Delta z_1\right)^2}\,\theta\,\frac{\Delta t}{P_1^j}\right)$$

$$+\ c_i^{j+1}\left(\frac{P^j_{1/2}\varepsilon^j_{1/2}\ D_{\mathrm{lb}}{}^j_{1/2}}{1/2(\Delta z_1)^2}\ (1-\theta)\ \frac{\Delta t}{P_1^j}\right)$$

The term

$$\left(\frac{\ell\cdot q^{j+1/2}}{\Delta z_1}\ \theta\ \frac{\Delta t}{P_1^j}\right)$$

is added to the element *RDD*, and the term

$$-\left(\frac{\ell\cdot q^{j+1/2}}{\Delta z_1}\ (1-\theta)\ \frac{\Delta t}{P_1^j}\right)$$

to the element *LDD*.
So, a right-hand vector is added to Eq. (6.39) (analogously to Eq. (6.26) for the water layer). The first element of this vector is composed of four (in case of $q > 0$) or two terms (in case of $q < 0$), which are described above and the other elements are zero.

Annex 9 The elements *LOD*, *LDD*, *LBD*, *ROD*, *RDD*, and *RBD* from Eq. (6.39) for the grid points *n* + 1 up to *ebbt* - 1 inclusive from the end buffer at the lower end of the sediment

$$LOD = - \frac{\ell \, q^{j+½}}{\Delta z_k} \, (1 - \theta) \, (1 - NWB_{k-½}) \, \frac{\Delta t}{P_k^j} \, sofd_{k-½}$$

$$- \frac{P^j_{k-½} \, \varepsilon^j_{k-½} \, (E_{\mathrm{lb}}{}^{j+½}_{k-½} + D_{\mathrm{lb}}{}^{j}_{k-½}) \, (1 - \theta)\Delta t}{\Delta z_k \, (½ \, \Delta z_k + ½ \, \Delta z_{k-1}) \, P_k^j}$$

$$LDD = \frac{\ell \, q^{j+½}}{\Delta z_k} \, (1 - \theta) \, (1 - NWB_{k+½}) \, \frac{\Delta t}{P_k^j} \, sofd_{k+½}$$

$$+ \frac{P^j_{k+½} \, \varepsilon^j_{k+½} \, (E_{\mathrm{lb}}{}^{j+½}_{k+½} + D_{\mathrm{lb}}{}^{j}_{k+½}) \, (1 - \theta)\Delta t}{\Delta z_k \, (½ \, \Delta z_{k+1} + ½ \, \Delta z_k) \, P_k^j}$$

$$- \frac{\ell \, q^{j+½}}{\Delta z_k} \, (1 - \theta) \, NWB_{k-½} \, \frac{\Delta t}{P_k^j} \, sofd_{k-½}$$

$$+ \frac{P^j_{k-½} \, \varepsilon^j_{k-½} \, (E_{\mathrm{lb}}{}^{j+½}_{k-½} + D_{\mathrm{lb}}{}^{j}_{k-½}) \, (1 - \theta)\Delta t}{\Delta z_k \, (½ \, \Delta z_k + ½ \, \Delta z_{k-1}) \, P_k^j}$$

$$+ \left(\varepsilon_k^j + \rho_{b\,k}^{\;j} \, K_F \left(\frac{c_{lb\,k}^{\;j+1}}{c_e} \right)^{n_{wb}-1} \right) \left(1 + k_b \, P_k^j \, (1 - \theta) \, \frac{\Delta t}{P_k^j} \right)$$

$$LBD = \frac{\ell \, q^{j+½}}{\Delta z_k} \, (1 - \theta) \, NWB_{k+½} \, \frac{\Delta t}{P_k^j} \, sofd_{k+½}$$

$$- \frac{P_{k+½}^j \, \varepsilon_{k+½}^j \, (E_{lb\,k+½}^{\;j+½} + D_{lb\,k+½}^{\;j}) \, (1 - \theta) \Delta t}{\Delta z_k \, (½ \, \Delta z_{k+1} + ½ \, \Delta z_k) \, P_k^j}$$

$$ROD = \frac{\ell \, q^{j+½}}{\Delta z_k} \, \theta \, (1 - NWB_{k-½}) \, \frac{\Delta t}{P_k^j} \, sofd_{k-½}$$

$$+ \frac{P_{k-½}^j \, \varepsilon_{k-½}^j \, (E_{lb\,k-½}^{\;j+½} + D_{lb\,k-½}^{\;j}) \, \theta \, \Delta t}{\Delta z_k \, (½ \, \Delta z_k + ½ \, \Delta z_{k-1}) \, P_k^j}$$

$$RDD = - \frac{\ell \, q^{j+½}}{\Delta z_k} \, \theta \, (1 - NWB_{k+½}) \, \frac{\Delta t}{P_k^j} \, sofd_{k+½}$$

$$- \frac{P_{k+½}^j \, \varepsilon_{k+½}^j \, (E_{lb\,k+½}^{\;j+½} + D_{lb\,k+½}^{\;j}) \, \theta \, \Delta t}{\Delta z_k \, (½ \, \Delta z_{k+1} + ½ \, \Delta z_k) \, P_k^j}$$

$$+ \frac{\ell\, q^{j+½}}{\Delta z_k}\ \theta\ NWB_{k-½}\ \frac{\Delta t}{P_k^j}\ sofd_{k-½}$$

$$- \frac{P_{k-½}^j\ \varepsilon_{k-½}^j\ (E_{1b}{}_{k-½}^{j+½} + D_{1b}{}_{k-½}^{j})\ \theta\ \Delta t}{\Delta z_k\ (½\ \Delta z_k + ½\ \Delta z_{k-1})\ P_k^j}$$

$$+ \left(\varepsilon_k^j + \rho_{b\,k}^{\ j}\ K_F \left(\frac{c_{1b\,k}^{\ j}}{c_e} \right)^{n_{wb}-1} \right) \left(1 - k_b\ P_k^j\ \theta\ \frac{\Delta t}{P_k^j} \right)$$

$$RBD = - \frac{\ell\, q^{j+½}}{\Delta z_k}\ \theta\ NWB_{k+½}\ \frac{\Delta t}{P_k^j}\ sofd_{k+½}$$

$$+ \frac{P_{k+½}^j\ \varepsilon_{k+½}^j\ (E_{1b}{}_{k+½}^{j+½} + D_{1b}{}_{k+½}^{j})\ \theta\ \Delta t}{\Delta z_k\ (½\ \Delta z_{k+1} + ½\ \Delta z_k)\ P_k^j}$$

Annex 10 The lower boundary condition at the grid point *ebbt* in the case of downward water flow in the sediment

$$LOD = - \frac{\ell\, q^{j+½}}{\Delta z_k} (1 - \theta)\, (1 - NWB_{k-½})\, \frac{\Delta t}{P_k^j}\, sofd_{k-½}$$

$$- \frac{P^j_{k-½}\, \varepsilon^j_{k-½}\, (E_{\mathrm{lb}\,k-½}^{\,j+½} + D_{\mathrm{lb}\,k-½}^{\,j})\, (1 - \theta)\Delta t}{\Delta z_k\, (½\, \Delta z_k + ½\, \Delta z_{k-1})\, P_k^j}$$

$$LDD = \frac{\ell\, q^{j+½}}{\Delta z_k} (1 - \theta)\, (1 - NWB_{k+½})\, \frac{\Delta t}{P_k^j}$$

$$- \frac{P^j_{k+½}\, \varepsilon^j_{k+½}\, (E_{\mathrm{lb}\,k+½}^{\,j+½} + D_{\mathrm{lb}\,k+½}^{\,j})\, (1 - \theta)\Delta t}{\Delta z_k\, (½\, \Delta z_{k+1} + ½\, \Delta z_k)\, P_k^j}$$

$$- \frac{\ell\, q^{j+½}}{\Delta z_k} (1 - \theta)\, NWB_{k-½}\, \frac{\Delta t}{P_k^j}\, sofd_{k-½}$$

$$+ \frac{P^j_{k-½}\, \varepsilon^j_{k-½}\, (E_{\mathrm{lb}\,k-½}^{\,j+½} + D_{\mathrm{lb}\,k-½}^{\,j})\, (1 - \theta)\Delta t}{\Delta z_k\, (½\, \Delta z_k + ½\, \Delta z_{k-1})\, P_k^j}$$

$$+ \left(\varepsilon_k^j + \rho_{\mathrm{b}\,k}^{\,j}\, K_{\mathrm{F}} \left(\frac{c_{\mathrm{lb}\,k}^{\,j+1}}{c_{\mathrm{e}}} \right) \right) \left(1 + k_{\mathrm{b}}\, P_k^j\, (1 - \theta)\, \frac{\Delta t}{P_k^j} \right)$$

$$LBD = \frac{\ell \, q^{j+1/2}}{\Delta z_k} (1 - \theta) \, NWB_{k+1/2} \, \frac{\Delta t}{P_k^j}$$

$$- \frac{P^j_{k+1/2} \, \varepsilon^j_{k+1/2} \, (E_{lb}{}^{j+1/2}_{k+1/2} + D_{lb}{}^{j}_{k+1/2}) \, (1 - \theta) \Delta t}{\Delta z_k \, (1/2 \, \Delta z_{k+1} + 1/2 \, \Delta z_k) \, P_k^j}$$

$$ROD = \frac{\ell \, q^{j+1/2}}{\Delta z_k} \, \theta \, (1 - NWB_{k-1/2}) \, \frac{\Delta t}{P_k^j} \, sofd_{k-1/2}$$

$$+ \frac{P^j_{k-1/2} \, \varepsilon^j_{k-1/2} \, (E_{lb}{}^{j+1/2}_{k-1/2} + D_{lb}{}^{j}_{k-1/2}) \, \theta \, \Delta t}{\Delta z_k \, (1/2 \, \Delta z_k + 1/2 \, \Delta z_{k-1}) \, P_k^j}$$

$$RDD = - \frac{\ell \, q^{j+1/2}}{\Delta z_k} \, \theta \, (1 - NWB_{k+1/2}) \, \frac{\Delta t}{P_k^j}$$

$$- \frac{P^j_{k+1/2} \, \varepsilon^j_{k+1/2} \, (E_{lb}{}^{j+1/2}_{k+1/2} + D_{lb}{}^{j}_{k+1/2}) \, \theta \, \Delta t}{\Delta z_k \, (1/2 \, \Delta z_{k+1} + 1/2 \, \Delta z_k) \, P_k^j}$$

$$+ \frac{\ell \, q^{j+1/2}}{\Delta z_k} \, \theta \, NWB_{k-1/2} \, \frac{\Delta t}{P_k^j} \, sofd_{k-1/2}$$

$$- \frac{P^j_{k-1/2} \, \varepsilon^j_{k-1/2} \, (E_{lb}{}^{j+1/2}_{k-1/2} + D_{lb}{}^{j}_{k-1/2}) \, \theta \, \Delta t}{\Delta z_k \, (1/2 \, \Delta z_k + 1/2 \, \Delta z_{k-1}) \, P_k^j}$$

$$+ \left(\varepsilon_k^j + \rho_{b\,k}^{\,j} K_F \left(\frac{c_{lb\,k}^{\,j}}{c_e} \right)^{n_{wb}-1} \right) \left(1 - k_b P_k^j \theta \frac{\Delta t}{P_k^j} \right)$$

$$RBD = \frac{\ell\, q^{j+\frac{1}{2}}}{\Delta z_k} \theta\, NWB_{k+\frac{1}{2}} \frac{\Delta t}{P_k^j}$$

$$+ \frac{P_{k+\frac{1}{2}}^{j}\, \varepsilon_{k+\frac{1}{2}}^{j} \left(E_{lb\,k+\frac{1}{2}}^{\,j+\frac{1}{2}} + D_{lb\,k+\frac{1}{2}}^{\,j} \right) \theta\, \Delta t}{\Delta z_k \left(\frac{1}{2} \Delta z_{k+1} + \frac{1}{2} \Delta z_k \right) P_k^j}$$

So, the elements *LBD* and *RBD* disappear.

Annex 11 The lower boundary condition at the grid point *ebbt* in the case of upward water flow in the sediment

$$LOD = -\frac{\ell\, q^{j+\frac{1}{2}}}{\Delta z_k}(1-\theta)(1-NWB_{k-\frac{1}{2}})\frac{\Delta t}{P_k^j}\, sofd_{k-\frac{1}{2}}$$

$$-\frac{P_{k-\frac{1}{2}}^j\, \varepsilon_{k-\frac{1}{2}}^j\, (E_{\mathrm{lb}\,k-\frac{1}{2}}^{\;j+\frac{1}{2}} + D_{\mathrm{lb}\,k-\frac{1}{2}}^{\;j})(1-\theta)\Delta t}{\Delta z_k\, (\frac{1}{2}\Delta z_k + \frac{1}{2}\Delta z_{k-1})\, P_k^j}$$

$$LDD = \frac{\ell\, q^{j+\frac{1}{2}}}{\Delta z_k}(1-\theta)(1-NWB_{k+\frac{1}{2}})\frac{\Delta t}{P_k^j}$$

$$+\frac{P_{k+\frac{1}{2}}^j\, \varepsilon_{k+\frac{1}{2}}^j\, (E_{\mathrm{lb}\,k+\frac{1}{2}}^{\;j+\frac{1}{2}} + D_{\mathrm{lb}\,k+\frac{1}{2}}^{\;j})(1-\theta)\Delta t}{\Delta z_k\, (\frac{1}{2}\Delta z_{k+1} + \frac{1}{2}\Delta z_k)\, P_k^j}$$

$$-\frac{\ell\, q^{j+\frac{1}{2}}}{\Delta z_k}(1-\theta)\, NWB_{k-\frac{1}{2}}\frac{\Delta t}{P_k^j}\, sofd_{k-\frac{1}{2}}$$

$$+\frac{P_{k-\frac{1}{2}}^j\, \varepsilon_{k-\frac{1}{2}}^j\, (E_{\mathrm{lb}\,k-\frac{1}{2}}^{\;j+\frac{1}{2}} + D_{\mathrm{lb}\,k-\frac{1}{2}}^{\;j})(1-\theta)\Delta t}{\Delta z_k\, (\frac{1}{2}\Delta z_k + \frac{1}{2}\Delta z_{k-1})\, P_k^j}$$

$$+\left(\varepsilon_k^j + \rho_{\mathrm{b}\,k}^{\;j}\, K_{\mathrm{F}}\left(\frac{c_{\mathrm{lb}\,k}^{\;j+1}}{c_{\mathrm{e}}}\right)^{n_{\mathrm{w}}-1}\right)\left(1 + k_{\mathrm{b}}\, P_k^j\, (1-\theta)\frac{\Delta t}{P_k^j}\right)$$

$$LBD = \frac{\ell\, q^{j+½}}{\Delta z_k} (1 - \theta)\, NWB_{k+½} \frac{\Delta t}{P_k^j}$$

$$\frac{P^j_{k+½}\, \varepsilon^j_{k+½} \left(E_{lb\,k+½}^{\;j+½} + D_{lb\,k+½}^{\;j}\right) (1 - \theta) \Delta t}{\Delta z_k \left(½\, \Delta z_{k+1} + ½\, \Delta z_k\right) P_k^j}$$

$$ROD = \frac{\ell\, q^{j+½}}{\Delta z_k}\, \theta\, (1 - NWB_{k-½}) \frac{\Delta t}{P_k^j}\, sofd_{k-½}$$

$$+ \frac{P^j_{k-½}\, \varepsilon^j_{k-½} \left(E_{lb\,k-½}^{\;j+½} + D_{lb\,k-½}^{\;j}\right) \theta\, \Delta t}{\Delta z_k \left(½\, \Delta z_k + ½\, \Delta z_{k-1}\right) P_k^j}$$

$$RDD = \frac{\ell\, q^{j+½}}{\Delta z_k}\, \theta\, (1 - NWB_{k+½}) \frac{\Delta t}{P_k^j}$$

$$\frac{P^j_{k+½}\, \varepsilon^j_{k+½} \left(E_{lb\,k+½}^{\;j+½} + D_{lb\,k+½}^{\;j}\right) \theta\, \Delta t}{\Delta z_k \left(½\, \Delta z_{k+1} + ½\, \Delta z_k\right) P_k^j}$$

$$+ \frac{\ell\, q^{j+½}}{\Delta z_k}\, \theta\, NWB_{k-½} \frac{\Delta t}{P_k^j}\, sofd_{k-½}$$

$$- \frac{P^j_{k-½}\, \varepsilon^j_{k-½} \left(E_{lb\,k-½}^{\;j+½} + D_{lb\,k-½}^{\;j}\right) \theta\, \Delta t}{\Delta z_k \left(½\, \Delta z_k + ½\, \Delta z_{k-1}\right) P_k^j}$$

$$+\left(\varepsilon_k^j - \rho_{b\,k}^{\,j}\, K_F \left(\frac{c_{lb\,k}^{\;j}}{c_e}\right)^{n_{wb}-1}\right)\left(1 - k_b\, P_k^j\, \theta\, \frac{\Delta t}{P_k^j}\right)$$

$$RBD = -\frac{\ell\, q^{j+½}}{\Delta z_k}\, \theta\, NWB_{k+½}\, \frac{\Delta t}{P_k^j}$$

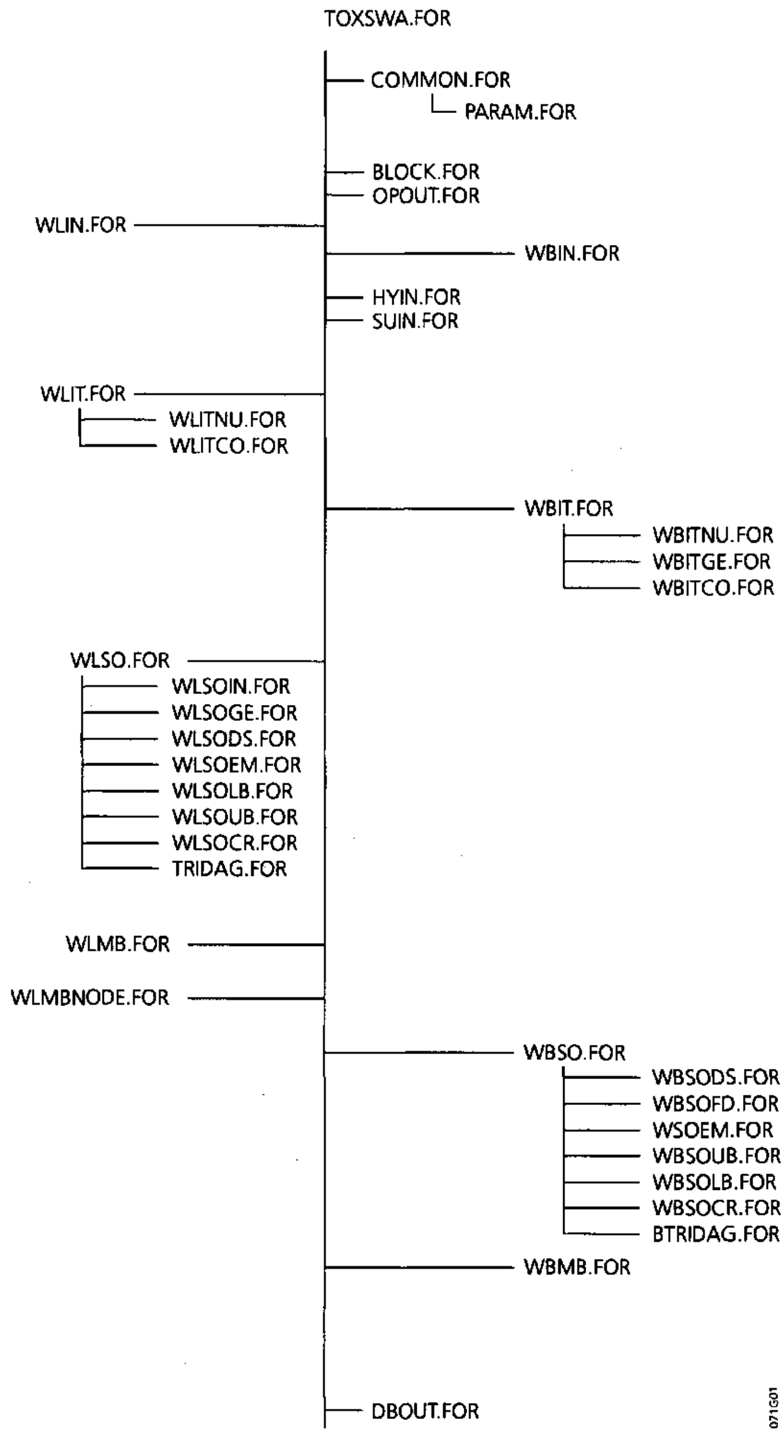
$$+\frac{P_{k+½}^{\,j}\, \varepsilon_{k+½}^{\,j}\, (E_{lb\,k+½}^{\;j+½} + D_{lb\,k+½}^{\;j})\, \theta\, \Delta t}{\Delta z_k\, (½\, \Delta z_{k+1} + ½\, \Delta z_k)\, P_k^j}$$

So, the elements *LBD* and *RBD* disappear. Two additional terms appear on the right-hand side of the numerical equations approximating the conservation equation, namely:

$$-\frac{\ell \cdot q^{j+½}}{\Delta z_{ebbt}}\, \theta\, c_{lb\,lot}^{\;j} \frac{\Delta t}{P_{ebbt}^j} - \frac{\ell \cdot q^{j+½}}{\Delta z_{ebbt}}\, (1-\theta)\, c_{lb\,lot}^{\;j+1} \frac{\Delta t}{P_{ebbt}^j}$$

This implies that, in the case of upward water flow, the last element of the right-hand vector *RV* is composed of the two terms described above.

Annex 12 Structure of the TOXSWA program



Meaning of symbols used:

First pairs:

b	bottom
db	distribution
hy	hydrology
op	options
tri	tridiagonal
su	substance
wb	sediment (water bottom)
wl	water layer

Second pairs:

dag	diagonal
in	input
it	initial calculations
mb	mass balance
out	output
so	solution

Third pairs:

co	concentration
cr	core of matrix
ds	dispersion
em	elements of matrix
fd	flux: dispersion - advection
ge	geometry
in	input
lb	lower boundary
nu	numerical aspects
ub	upper boundary

Annex 13 Description of the TOXSWA model including subroutines

TOXSWA.FOR
Main program. Initiates and controls time step and node handling. Terminates run.

COMMON.FOR
Declares all the common variables for the program TOXSWA.

> PARAM.FOR
> Defines parameters that are used for the declaration of the dimensions of certain arrays for the program. These are included in the common block.

BLOCK.FOR
Initialises identifiers for names of output files.

OPOUT.FOR
Reads all selected options concerning the output, such as desired output files, points of time for output or nodes for which detailed output is desired.

WLIN.FOR
Reads all input data for the water subsystem, concerning numerical aspects and length of ditch (from WLNU.INP), geometry and other characteristics of the ditch (from WLPA.INP) and initial concentrations (from WLST.INP). Data are converted to SI-units (m, s, g, Pa, K, J, mol).

WLIT.FOR
Calculates initial conditions for the water layer. Segment lengths in the ditch plus buffers are determined.

> WLITNU.FOR
> Calculates the numerical weight factors for the ditch and the buffers.

> WLITCO.FOR
> Calculates initial concentrations c, X_{mp} and X_{ss} for every node.

WBIN.FOR
Reads all input data for the sediment subsystem, concerning numerical aspects and thickness of the sediment (from WBNU.INP), properties of the sediment (from WBPA.INP) and initial concentrations (from WBST.INP). The input data are converted to SI-units (m, s, g, Pa, K, J, mol).

SUIN.FOR
Reads all data concerning substance properties, e.g. transformation rates in water and sediment, sorption parameters, solubility from SU.INP.

HYIN.FOR

Reads all hydrological input data for the water layer and sediment from HY.INP. All input data are converted to SI-units (m, s, g, Pa, K, J, mol).

WBIT.FOR

Calculates initial conditions for the sediment. The thickness of the segments in the sediment plus buffer are determined.

WBITNU.FOR

Calculates the numerical weight factors for the sediment and the end buffer.

WBITGE.FOR

Calculates initial conditions concerning perimeter, porosity, bulk density, Freundlich sorption coefficient and diffusion coefficient.

WBITCO.FOR

Calculates initial concentrations c_{1b} and X_b at every node.

WLSO.FOR

Solves the mass conservation equation for the water layer.

WLSOIN.FOR

Reads concentration for the first node of the sediment, for each node of the water layer, and seepage/infiltration per node.

WLSOGE.FOR

Calculates geometrical time-dependent characteristics.

WLSODS.FOR

Calculates dispersion coefficient; equals physical dispersion minus numerical dispersion.

WLSOEM.FOR

Defines elements of the matrix which are used in subroutines WLSOLB.FOR, WLSOUB.FOR and WLSOCR.FOR.

WLSOLB.FOR

Composes the elements of the matrix for the buffer at the lower boundary.

WLSOUB.FOR

Composes the elements of the matrix for the buffer at the upper boundary.

WLSOCR.FOR

Composes the core matrix, consisting of the numerical equations for solving the mass conservation equation.

TRIDAG.FOR

Inverts the left-hand tridiagonal matrix for the water layer.

WLMB.FOR
Checks the mass balance for the water layer.

WLMBNODE.FOR
Calculates the mass balance for selected segments of the water layer (to allow the provision of output at selected nodes).

WBSO.FOR
Solves the mass conservation equation for the sediment.

WBSODS.FOR
Calculates dispersion coefficient; equals physical dispersion minus numerical dispersion.

WBSOFD.FOR
Prohibits calculated dispersion flux from cancelling out advection flux.

WBSOEM.FOR
Defines elements of the matrix which are used in subroutines WBSOLB.FOR, WBSOUB.FOR and WBSOCR.FOR.

WBSOUB.FOR
Composes the elements of the matrix at the upper node of the sediment (upper boundary).

WBSOLB.FOR
Composes the elements of the matrix for the buffer at the lower boundary.

WBSOCR.FOR
Composes the core matrix, consisting of the numerical equations for solving the mass conservation equation.

BTRIDAG.FOR
Inverts the left-hand tridiagonal matrix for the sediment.

WBMB.FOR
Checks the mass balance for the sediment.

DBOUT.FOR
Calculates output concerning the pesticide mass distribution between the different compartments (water phase, macrophytes, suspended solids, liquid phase sediment and solid phase sediment) for the total ditch as well as per running metre.

Annex 14 Outline proving the positivity and stability of the matrix equation $L.c^{n+1} = R.c^n$, with L and R as the two tridiagonal matrices of Eqs. (6.26) or (6.39)

Matrix equation $L.c^{n+1} = R.c^n$ contains two tridiagonal matrices L and R, which read

$$L = \begin{pmatrix} LDD & LBD & O & . & . & . & . \\ LOD & LDD & LBD & O & . & . & . \\ O & LOD & LDD & LBD & O & & \\ . & . & . & . & . & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ . & . & . & O & LOD & LDD & LBD \\ . & . & . & . & O & LOD & LDD \end{pmatrix}$$

, similarly R has been defined and the elements LOD, LDD, LBD, ROD, RDD and RBD are defined in section 7.3 (for a system with constant segment size Δx, flow velocity u and water depth h and a linear sorption isotherm describing sorption to suspended solids). The proof of the following statement (s) will now be sketched.

Condition (c): Δt and Δx are chosen in such a way that

$$LOD \le 0 \qquad ROD \ge 0$$
$$LDD > 0 \quad \text{and} \quad RDD > 0$$
$$LBD \le 0 \qquad RBD \ge 0.$$

Statement (s): If condition (c) holds, then $L^{-1}.R$ exists, $L^{-1}.R$ is positive and $L^{-1}.R$ is stable.

Statement (s) guarantees that matrix equation $L.c^{n+1} = R.c^n$ can be solved by inverting L and subsequent calculation of $L^{-1}.R$. The positivity of $L^{-1}.R$ guarantees positive solutions from positive initial conditions and the stability of $L^{-1}.R$ guarantees that a solution indeed exists. It will be shown that statement (s) holds, because of the dependencies in the non-zero matrix elements of L and R.

We write $A \ge 0$, so A is positive, if a matrix A has no negative elements $a_{ij} \ge 0$. A positive matrix A is called stable if the dominant eigenvalue of A (i.e. the eigenvalue with the largest absolute value) $\rho(A) < 1$.

Step 1: Positivity

Condition (c) states that $R \geq 0$ and that for L, $l_{ii} > 0$ (diagonal elements) and $l_{ij} \leq 0$, $i \neq j$ (off-diagonal elements). Define a matrix $B = I - DL$, where I is the identity matrix and where D is a diagonal matrix with $d_{ii} = 1/l_{ii}$. We now assert that $B \geq 0$ and $\rho(B) < 1$. We illustrate this assertion for the three-dimensional case below. Theorem 3.10 of Varga (1962) states that (i) L^{-1} exists and $L^{-1} \geq 0$, corresponds with (ii) $B \geq 0$ and $\rho(B) < 1$ (for B defined as above and for L with $l_{ii} > 0$). It now follows from Theorem 3.10 that L^{-1} exists and $L^{-1} \geq 0$. As R is also positive, $L^{-1}.R \geq 0$.

Illustration for the three-dimensional case:

Write $a = LDD > 0$, $-b = LBD \leq 0$, $-c = LOD \leq 0$, so that $a,b,c \geq 0$ under condition (c).

$$L = \begin{pmatrix} a & -b & o \\ -c & a & -b \\ o & -c & a \end{pmatrix}$$

From the definition for LOD, LDD and LBD in section 7.3 we find easily: $a > b$, $a > c$ and $a^2 > 2bc$. We find

$$B = \begin{pmatrix} o & \frac{b}{a} & o \\ \frac{c}{a} & o & \frac{b}{a} \\ o & \frac{c}{a} & o \end{pmatrix}$$

for which clearly $B \geq 0$ and for which $\rho(B) = \sqrt{(2bc/a^2)} < 1$.

Step 2: Stability

To prove stability we make use of Theorem 3.13 of Varga (1962). Theorem 3.13 states that if $A = L - R$ with $L^{-1} \geq 0$ and $R \geq 0$ ánd $A^{-1} \geq 0$ then, $\rho(L^{-1}.R) < 1$.

If we define $A = L - R$ we know from step 1 that under condition (c) $L^{-1} \geq 0$ and $R \geq 0$. To demonstrate that $A^{-1} \geq 0$ we can use the same arguments as in step 1, because A has a similar structure as L (i.e. $a_{ii} > 0$ and $a_{ij} \leq 0$, $i \neq j$). So, according to theorem 3.13 of Varga $\rho\ (L^{-1}.\ R) < 1$, so the solution is stable.

So, step 1 and step 2 together outline the proof of statement (s).

Annex 15 Changes in the source code of TOXSWA 1.0 for comparison with the analytical solutions for the water layer and the sediment

```
**      SUMMARY OF THE CHANGES IN THE SOURCE CODE OF THE TOXSWA 1.0
**      PROGRAM, WHICH WERE MADE TO ENABLE THE COMPARISON OF TOXSWA 1.0
**      WITH ANALYTICAL SOLUTIONS FOR THE MASS CONSERVATION EQUATIONS OF
**      THE WATER AND SEDIMENT SUBSYSTEMS
**
**
**
        subroutine wlsoem

** SUBROUTINE
**      wlsoem - the elements of the matrix, which are used in the subroutines
**               wlsolb, wlsoub and wlsocr, are defined here
**


        lddwbdf        = 0.
*                         ((2.*por(1)*kdfwbmh(1)*pez0hw)/delz(1))*
*       &                 (1.-thetawl)*deltwl


        rddwbdf        = 0.
*                         ((2.*por(1)*kdfwbmh(1)*pez0hw)/delz(1))*thetawl*
*       &                 deltwl


        rvwbdf         = 0.
*                          ((2.*por(1)*kdfwbmh(1)*pez0hw)/delz(1)) *
*       &                  cowbjkis1(ixnotot) * deltwl




**
**
**
        subroutine wbsoub

** SUBROUTINE
**      wbsoub - the first row of the two tridiagonal matrices will be
**               composed here
**


        if (qseifjph.ge.zero) then

           bldd(kznotot) = blddadph + blddsdfph +
*       &                    (pemh(1)*pormh(1)*kdfwbmh(1)*(1.-thetawb)
*       &                    *deltwb)
*       &                    / (0.5*(delz(1)**2.)*pe(1)) +
       &                   blddca + blddtf
           blbd(kznotot) = blbdad - blbddsdf

           brdd(kznotot) = -brddadph - brdddsdfph
*       &                    -(pemh(1)*pormh(1)*kdfwbmh(1)*thetawb*deltwb)/
*       &                    (0.5*(delz(1)**2.)*pe(1))
       &                   + brddca - brddtf
           brbd(kznotot) = -brbdad + brbddsdf

           brv(kznotot)  = cowlj(ixnotot) *(
        &                  (leplot*qseifjph*thetawb
        &                  *deltwb)/
        &                  (delz(1)*pe(1))) +
*       &                   (pemh(1)*pormh(1)*kdfwbmh(1)*
*       &                   thetawb*deltwb) / (0.5*(delz(1)**2.)*pe(1)))+
        &                  cowljpl(ixnotot) *(
        &                 (leplot*qseifjph*
        &                  (1.-thetawb)*
        &                  deltwb) / (delz(1)*pe(1)))
```

```
*     &                    (pemh(1)*pormh(1)*
*     &                    kdfwbmh(1)*(1.-thetawb)*deltwb) / (0.5*
*     &                    (delz(1)**2.)*pe(1)))
      else


**
**
**
      subroutine wlmb

** SUBROUTINE
**    wlmb - the mass balance is checked for the water layer
**


C---- Calculation of incoming substance from sediment and of
C     substance penetrating in sediment
C     (this is total P*Jwb*delx at time j and location i)
C     This takes only place in ditch, not in front- and endbuffer

      totrsinwb = 0.0
      totrsoutwb = 0.0
      do 10 ixnotot = nxnofb+1, nxnofb+nxnodit
         if (qseifj.ge.zero) then
            rsinwb = leplot*qseifj*cowlj(ixnotot)*delx(ixnotot)

*     &              - por(1)*kdfwbmh(1)*delx(ixnotot)*pez0hw*
*     &              ((cowbjkis1(ixnotot)-cowlj(ixnotot))
*     &              /(0.5*delz(1)))
         else
C                         qseifj is negative
            rsinwb = leplot*qseifj*cowbjkis1(ixnotot)*delx(ixnotot)

*     &              - por(1)*kdfwbmh(1)*delx(ixnotot)*pez0hw*
*     &              ((cowbjkis1(ixnotot)-cowlj(ixnotot))
*     &              /(0.5*delz(1)))
         end if
         totrsinwb = totrsinwb + amin1(0.0, rsinwb)
         totrsoutwb = totrsoutwb + amax1(0.0, rsinwb)
   10 continue


**
**
**
      subroutine wlmbnode

** SUBROUTINE
**    wlmbnode - the mass balance is made for a selected section of
**               the water layer
**


C---- Calculation of incoming substance from sediment and of substance
C     penetrating in sediment in selected section
C     (this is P*Jwb*delx at time j and location ixnotot)

            if (ixnotot.ge.nxnofb+1 .and. ixnotot.le.nxnofb+nxnodit)
     &                                                             then
               if (qseifj.ge.zero) then
                  rsinwbn = leplot*qseifj*cowlj(ixnotot)*
     &                      delx(ixnotot)

*     &                      -por(1)*kdfwbmh(1)*delx(ixnotot)*pez0hw*
*     &                      ((cowbjkis1(ixnotot)-cowlj(ixnotot))
*     &                      /(0.5*delz(1)))
               else
C                         qseifj is negative
                  rsinwbn = leplot*qseifj*cowbjkis1(ixnotot)*
     &                      delx(ixnotot)
```

```
*     &                        -por(1)*kdfwbmh(1)*delx(ixnotot)*pez0hw*
*     &                        ((cowbjkis1(ixnotot)-cowlj(ixnotot))
*     &                        /(0.5*delz(1)))
                   end if
                   rsoutwbn = amax1(0.0, rsinwbn)
                   rsinwbn = amin1(0.0, rsinwbn)
               else
                   rsoutwbn = zero
                   rsinwbn = zero
               end if


**
**
**
      subroutine wbmb

** SUBROUTINE
**    wbmb - the mass balance is checked for the water bottom
**


C---- Calculation of incoming substance from water layer and of substance
C     lost to water layer
C     (this is P*J1b at time j and k=1/2)

      if (qseifj.ge.zero) then
         rsinwl = leplot*qseifj*cowlj(ixnotot)
*     &             - pemh(1)*pormh(1)*
*     &            kdfwbmh(1)*((cowbj(1)-cowlj(ixnotot))/(.5*delz(1)))
      else
C                            qseifj is negative
         rsinwl = leplot*qseifj*cowbj(1)
*     &             - pemh(1)*pormh(1)*
*     &            kdfwbmh(1)*((cowbj(1)-cowlj(ixnotot))/(.5*delz(1)))
      end if
      rsoutwl = amin1(0.0, rsinwl)
      rsinwl = amax1(0.0, rsinwl)


C---- Calculation of substance lost to water layer
C     (this is P*J1b at time j and k=1/2)

      if (qseifj.ge.zero) then
         rsoutwl = leplot*qseifj*cowlj(ixnotot)
*     &             - pemh(1)*pormh(1)*
*     &            kdfwbmh(1)*((cowbj(1)-cowlj(ixnotot))/(.5*delz(1)))
      else
C                            qseifj is negative
         rsoutwl = leplot*qseifj*cowbj(1)
*     &             - pemh(1)*pormh(1)*
*     &            kdfwbmh(1)*((cowbj(1)-cowlj(ixnotot))/(.5*delz(1)))
      end if
      rsoutwl = amin1(0.0, rsoutwl)
```

Annex 16 Changes in the input files (compared to those of the example simulation for chlorpyrifos) for calculating TOXSWA output for comparison with the analytical solution for the sediment

```
*   Filename :   WLPA.INP
*   Content  :   Input data for TOXSWA concerning geometry and other
*                characteristics of the ditch
*
*------------------------------------------------------------------------
*
*   Section 1:   Geometry of ditch
*------------------------------------------------------------------------
*
*   side slope, horizontal/vertical, s_1
  sisl = 1.E-05              ! unit: -        range: 1.E-05 .... 10.0
*
*----------------End of file---------------------------------------------
```

```
*   Filename :   WLST.INP
*   Content  :   Input data for TOXSWA concerning initial concentrations
*                in water layer
*
*------------------------------------------------------------------------
*
*   Section 1:   Initial concentrations
*------------------------------------------------------------------------
*
*   initial (start) mass concentration of pesticide in water layer, c^*,
*   for the total number of nodes in x direction (nxnotot, so buffers
*   included)
  castwl = 0.                ! unit: g/m^3    range: 0. .... 100.0
           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.
```

```
                0.
                0.
                0.
                0.
                0.
                0.
                0.
                0.
                0.
                0.
                 0.
                 0.
                 0.
                 0.
                 0.
                 0.
                 0.
                 0.
*
*---------------End of file-----------------------------------------

*   Filename :   WBPA.INP
*   Content  :   Input data for TOXSWA concerning characteristics of the
*                sediment
*
*-------------------------------------------------------------------
*
*   Section 1:   Physical properties of sediment
*-------------------------------------------------------------------
*
*   bdwb(1-nznowb), bulk density of dry sediment material, rho_b (as a
*   function of depth, end buffer excluded)
*
*   por(1-nznowb), porosity (volume fraction of void water), epsilon (as
*   a function of depth, end buffer excluded)
*
*   tor(1-nznowb), tortuosity, lamba (as a function of depth, end buffer
*   excluded)
*
 bdwb                    por                    tor
 1000.                   0.65                   0.65
 1000.                   0.65                   0.65
 1000.                   0.65                   0.65
 1000.                   0.65                   0.65
 1000.                   0.65                   0.65
 1000.                   0.65                   0.65
 1000.                   0.65                   0.65
 1000.                   0.65                   0.65
 1000.                   0.65                   0.65
 1000.                   0.65                   0.65
  1000.                   0.65                   0.65
  1000.                   0.65                   0.65
  1000.                   0.65                   0.65
  1000.                   0.65                   0.65
  1000.                   0.65                   0.65
  1000.                   0.65                   0.65
  1000.                   0.65                   0.65
  1000.                   0.65                   0.65
  1000.                   0.65                   0.65
  1000.                   0.65                   0.65
   1000.                   0.65                   0.65
   1000.                   0.65                   0.65
   1000.                   0.65                   0.65
! kg/m^3                 -                      -           unit
! 10. .... 3000.         0. .... 1.             0. .... 1.  range
*
*-------------------------------------------------------------------
*
*   Section 2:   Organic matter in sediment
*-------------------------------------------------------------------
*
*   mass ratio of organic matter of dry sediment material, m_om,wb (as a
*   function of depth, end buffer excluded)
```

```
 raomwb = 0.05                 ! unit: -        range: 0. .... 1.
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
*
*----------------End of file-----------------------------------------

*   Filename :  WBST.INP
*   Content  :  Input data for TOXSWA concerning initial concentrations
*               in sediment
*
*--------------------------------------------------------------------
*
*   Section 1:  Initial concentrations
*--------------------------------------------------------------------
*
*   initial (start) mass concentration pesticide in sediment, c_b^*,
*   for the total number of nodes in z direction (nznotot, so end
*   buffer included)
 castwb = 10.                 ! unit: g/m^3     range: 0. .... 1000.0
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
          0.
*
*----------------End of file-----------------------------------------

*   Filename :  HY.INP
*   Content  :  Input data for TOXSWA concerning hydrological
*               characteristics
*
*--------------------------------------------------------------------
*
*   Section 3:  Seepage with concentration
*--------------------------------------------------------------------
*
```

```
*   qseif:
*   seepage from neighbouring lot to ditch or seepage from ditch to
*   neighbouring lot (negative and positive, resp.) expressed per m^2
*   lot for 30 days
*
*   colot:
*   concentration of pesticide in incoming water seeping from
*   neighbouring lot into the ditch bottom (0. in the case of seepage
*   from ditch into neighbouring lot)
*
  qseif                          colot
   2.0E-03                       0.
   2.0E-03                       0.
   2.0E-03                       0.
   2.0E-03                       0.
   2.0E-03                       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.                            0.
     0.                            0.
     0.                            0.
     0.                            0.
     0.                            0.
     0.                            0.
! m^3/m^2.d                     g/m^3              unit
! -10.E-03 .... +10.E-03        0. .... 1.         range
*
*----------------End of file-------------------------------------------

*   Filename :   SU.INP
*   Content  :   Input data for TOXSWA concerning the substance
*
*----------------------------------------------------------------------
*
*   Section 2:   Sorption to suspended solids
*----------------------------------------------------------------------
*
*   Freundlich exponent for sorption to suspended solids, n_ss
  exfrss = 1.0               ! unit: -          range: 0.1 .... 2.0
*
*----------------------------------------------------------------------
*
*   Section 7:   Sorption to sediment
*----------------------------------------------------------------------
*
*   Freundlich exponent for sorption to sediment material, n_wb
  exfrwb = 1.0               ! unit: -          range: 0.1 .... 2.0
*
*----------------End of file-------------------------------------------
```

Annex 17 Changes in the input files (compared to those of the example simulation for chlorpyrifos) for calculating TOXSWA output for comparison with the analytical solution for the water layer

```
*   Filename :   WLNU.INP
*   Content  :   Input data for TOXSWA concerning numerical solution and
*                time and length of ditch considered
*
*----------------------------------------------------------------------
*
*   Section 3:    Space parameters
*----------------------------------------------------------------------
*
*   total length of ditch considered
  xdit = 400.                ! unit: m          range: 10. .... 10,000.
*
*----------------------------------------------------------------------
*
*   Section 4:    Number of space nodes
*----------------------------------------------------------------------
*
*   number of nodes in ditch
  nxnodit = 58               ! unit: -          range: 10 .... 200
*
*----------------------------------------------------------------------
*
*   Section 5:    Location of space nodes
*                 (x coordinate starts in front buffer)
*----------------------------------------------------------------------
*
*   x coordinate of nodes in ditch
  xcddit =   4.  12.  20.  28. ! unit: m        range: 0.1 .... 10,000.
            36.  44.  52.  60.
            68.  76.  84.  92.
           100. 106. 112. 118.
           124. 130. 136. 142.
           148. 154. 160. 166.
           172. 178. 184. 190.
           196.
           204. 212. 220. 228.
           236. 244. 252. 260.
           268. 276. 284. 292.
           300. 306. 312. 318.
           324. 330. 336. 342.
           348. 354. 360. 366.
           372. 378. 384. 390.
           396.
*
*---------------End of file--------------------------------------------

*   Filename :   WLPA.INP
*   Content  :   Input data for TOXSWA concerning geometry and other
*                characteristics of the ditch
*
*----------------------------------------------------------------------
*
*   Section 1:   Geometry of ditch
*----------------------------------------------------------------------
*
*   side slope, horizontal/vertical, s_1
  sisl = 1.E-05              ! unit: -          range: 1.E-05 .... 10.0
*
*---------------End of file--------------------------------------------

*   Filename :   WLST.INP
*   Content  :   Input data for TOXSWA concerning initial concentrations
*                in the water layer
*
```

```
*------------------------------------------------------------------------
*
*   Section 1:    Initial concentrations
*------------------------------------------------------------------------
*
*   initial (start) mass concentration of pesticide in water layer, c^*,
*   for the total number of nodes in x direction (nxnotot, so buffers
*   included)
 castwl = 0.                    ! unit: g/m^3      range: 0. ... 100.0
          0.
          0.
          0.
          0.
          0.
          0.
          8.264e-03
          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.
              0.
              0.
               0.
               0.
               0.
               0.
               0.
               0.
               0.
*
*-------------End of file-----------------------------------------------


*   Filename  :   WBPA.INP
*   Content   :   Input data for TOXSWA concerning characteristics of the
*                 sediment
*
```

```
*-----------------------------------------------------------------------
*
*   Section 1:   Physical properties of sediment
*-----------------------------------------------------------------------
*
*   bdwb(1-nznowb), bulk density of dry sediment material, rho_b (as a
*   function of depth, end buffer excluded)
*
*   por(1-nznowb), porosity (volume fraction of void water), epsilon (as
*   a function of depth, end buffer excluded)
*
*   tor(1-nznowb), tortuosity, lamba (as a function of depth, end buffer
*   excluded)
*
 bdwb                      por                      tor
1000.                     0.65                     0.65
1000.                     0.65                     0.65
1000.                     0.65                     0.65
1000.                     0.65                     0.65
1000.                     0.65                     0.65
1000.                     0.65                     0.65
1000.                     0.65                     0.65
1000.                     0.65                     0.65
1000.                     0.65                     0.65
1000.                     0.65                     0.65
 1000.                     0.65                     0.65
 1000.                     0.65                     0.65
 1000.                     0.65                     0.65
 1000.                     0.65                     0.65
 1000.                     0.65                     0.65
 1000.                     0.65                     0.65
 1000.                     0.65                     0.65
 1000.                     0.65                     0.65
 1000.                     0.65                     0.65
 1000.                     0.65                     0.65
  1000.                     0.65                     0.65
  1000.                     0.65                     0.65
  1000.                     0.65                     0.65
! kg/m^3                   -                        -          unit
! 10. .... 3000.           0. .... 1.               0. .... 1.  range
*
*-----------------------------------------------------------------------
*
*   Section 2:   Organic matter in sediment
*-----------------------------------------------------------------------
*
*   mass ratio of organic matter of dry sediment material, m_om,wb (as a
*   function of depth, end buffer excluded)
 raomwb = 0.05             ! unit: -          range: 0. .... 1.
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
          0.05
           0.05
           0.05
           0.05
           0.05
           0.05
           0.05
           0.05
           0.05
           0.05
           0.05
            0.05
            0.05
            0.05
*
*---------------End of file---------------------------------------------
```

```
*   Filename :    HY.INP
*   Content  :    Input data for TOXSWA concerning hydrological
*                 characteristics
*
*-----------------------------------------------------------------------
*
*   Section 3:    Seepage with concentration
*-----------------------------------------------------------------------
*
*   qseif:
*   seepage from neighbouring lot to ditch or seepage from ditch to
*   neighbouring lot (negative and positive, resp.) expressed per m^2
*   lot for 30 days
*
*   colot:
*   concentration of pesticide in incoming water seeping from
*   neighbouring lot into the ditch bottom (0. in the case of seepage
*   from ditch into neighbouring lot)
*
 qseif                        colot
  2.0E-03                     0.
  2.0E-03                     0.
  2.0E-03                     0.
  2.0E-03                     0.
  2.0E-03                     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.                          0.
    0.                          0.
    0.                          0.
    0.                          0.
    0.                          0.
    0.                          0.
! m^3/m^2.d                  g/m^3          unit
! -10.E-03 .... +10.E-03     0. .... 1.     range
*
*---------------End of file---------------------------------------------

*   Filename :    SU.INP
*   Content  :    Input data for TOXSWA concerning the substance
*
*-----------------------------------------------------------------------
*
*   Section 2:    Sorption to suspended solids
*-----------------------------------------------------------------------
*
*   Freundlich exponent for sorption to suspended solids, n_ss
 exfrss = 1.0             ! unit: -          range: 0.1 .... 2.0
*
*-----------------------------------------------------------------------
*
*   Section 7:    Sorption to sediment
*-----------------------------------------------------------------------
*
*   Freundlich exponent for sorption to sediment material, n_wb
 exfrwb = 1.0             ! unit: -          range: 0.1 .... 2.0
*
*---------------End of file---------------------------------------------
```

Annex 18 Input files for the example simulation for chlorpyrifos

```
*   Filename :   OPOUT.INP
*   Content  :   Input data for TOXSWA concerning options for obtaining
*                model output
*
*------------------------------------------------------------------------
*
*   Section 1:   Output files selected
*------------------------------------------------------------------------
*
*   output file input.out (echo of all input) desired ? (0 = no, 1 = yes
  op_input = 1                  ! unit: -          range: 0 .... 1
*   output file icwlhy.out (initial calculations for water layer and
*   hydrology and exposure concentrations) desired ? (0 = no, 1 = yes)
  op_icwlhy = 1                 ! unit: -          range: 0 .... 1
*   output file icwb.out (initial calculations for sediment) desired ?
*   (0 = no, 1 = yes)
  op_icwb = 1                   ! unit: -          range: 0 .... 1
*   output file wlmb.out ( mass balance for water layer) desired ?
*   0 = no, 1 = yes)
  op_wlmb = 1                   ! unit: -          range: 0 .... 1
*   output file wlmbnodenr.out ( mass balance for a segment of water
*   layer desired ? (0 = no, 1 = yes)
  op_wlmbnodenr = 1             ! unit: -          range: 0 .... 1
*   output file wbsconodenr.out (concentrations in a sediment subsystem
*   desired ? (0 = no, 1 = yes)
  op_wbsconodenr = 1            ! unit: -          range: 0 .... 1
*   output file wbmbnodenr.out (mass balance for a sediment subsystem)
*   desired ? (0 = no, 1 = yes)
  op_wbmbnodenr = 1             ! unit: -          range: 0 .... 1
*   output file wbmball.out (mass balance of all sediment subsystems)
*   desired ? (0 = no, 1 = yes)
  op_wbmball = 1                ! unit: -          range: 0 .... 1
*   output file dbnodenr.out (distribution of substance in wl+wb at
*   nodenr wl) desired ? (0 = no, 1 = yes)
  op_dbnodenr = 1               ! unit: -          range: 0 .... 1
*   output file dbdit.out (distribution of substance in entire ditch)
*   desired ? (0 = no, 1 = yes)
  op_dbdit = 1                  ! unit: -          range: 0 .... 1
*
*------------------------------------------------------------------------
*
*   Section 2:   Points in time for output
*------------------------------------------------------------------------
*
*   number of points in time at which output is desired (max. of 9)
  nditout = 5                     ! unit: -          range: 1 .... 9
*   points in time at which output is desired
  ptditout = 0.  0.5  1.          ! unit: d          range: 0. .... 100.
             2.  4.               ! (only ptditout(1) may equal 0.)
*
*------------------------------------------------------------------------
*
*   Section 3:   Segments of wl+sediment subsystems selected for output
*------------------------------------------------------------------------
*
*   number of segments wl, coupled to sediment subsystems for which
*   output is desired (max. of 9)
  nwbsy = 3                     ! unit: -          range: 1 .... 9
*   node number in water layer at/or under which output is desired
  iwbsy= 1  13  29              ! unit: -          range: 1 .... 200
*   number of upper segments forming the top layer for which the
*   accumulated pesticide mass will be calculated
  ktop = 11                     ! unit: -          range: 1 .... 30
*
*----------------End of file---------------------------------------------
```

```
*   Filename :    WLNU.INP
*   Content  :    Input data for TOXSWA concerning numerical solution and
*                 time and length of ditch considered
*
*-----------------------------------------------------------------------
*
*   Section 1:    Numerical weight factors
*-----------------------------------------------------------------------
*
*   numerical weight factor for space
  betawl = 0.5              ! unit: -           range: 0.0 .... 1.0
*   numerical weight factor for time
  thetawl = 1.0             ! unit: -           range: 0.0 .... 1.0
*
*-----------------------------------------------------------------------
*
*   Section 2:    Time parameters
*-----------------------------------------------------------------------
*
*   selected time step for water layer
  deltwl = 100.             ! unit: s           range: 1. .... 86,400.
*   total time considered (holds also for sediment)
  ttot = 4.0                ! unit: d           range: 0.1 .... 100.
*
*-----------------------------------------------------------------------
*
*   Section 3:     Space parameters
*-----------------------------------------------------------------------
*
*   total length of ditch considered
  xdit = 200.               ! unit: m           range: 10. .... 10,000.
*   length of front buffer (0. if none)
  xfb = 0.                  ! unit: m           range: 0. .... 1000.
*   length of end buffer (0. if none)
  xeb = 0.                  ! unit: m           range: 0. .... 1000.
*
*-----------------------------------------------------------------------
*
*   Section 4:     Number of space nodes
*-----------------------------------------------------------------------
*
*   number of nodes in ditch
  nxnodit = 29              ! unit: -           range: 10 .... 200
*   number of nodes in front buffer (0 if none)
  nxnofb = 0                ! unit: -           range: 0 .... 25
*   number of nodes in end buffer (0 if none)
  nxnoeb = 0                ! unit: -           range: 0 .... 25
*
*-----------------------------------------------------------------------
*
*   Section 5:     Location of space nodes
*                  (x coordinate starts in front buffer)
*-----------------------------------------------------------------------
*
*   x coordinate of nodes in front buffer (0. if none)
  xcdfb = 0.                ! unit: m           range: 0. .... 1000.
*   x coordinate of nodes in ditch
  xcddit =  4.  12.  20.  28. ! unit: m         range: 0.1 .... 10,000.
           36.  44.  52.  60.
           68.  76.  84.  92.
          100. 106. 112. 118.
          124. 130. 136. 142.
          148. 154. 160. 166.
          172. 178. 184. 190.
          196.
*   x coordinate of nodes in end buffer (0. if none)
  xcdeb = 0.                ! unit: m           range: 0. .... 1000.
*
*----------------End of file--------------------------------------------
```

```
*   Filename :   WLPA.INP
*   Content  :   Input data for TOXSWA concerning geometry and other
*                characteristics of the ditch
*
*--------------------------------------------------------------------------
*
*   Section 1:   Geometry of ditch
*--------------------------------------------------------------------------
*
*   bottom width of ditch, b
  wibot = 1.65              ! unit: m          range: 0.1 .... 10.0
*   side slope, horizontal/vertical, s_1
  sis1 = 2.0                ! unit: -          range: 1.E-05 .... 10.0
*   water depth defining perimeter for exchange water layer - sediment,
*   h_w
  wdhfl = 0.10              ! unit = m         range =0.01 .... 2.0
*
*--------------------------------------------------------------------------
*
*   Section 2:   Geometry of the neighbouring plot of land
*--------------------------------------------------------------------------
*
*   length of draining plot, perpendicular to ditch and located at one
*   or both sides
  leplot = 100.              ! unit: m          range: 1. .... 1000.
*
*--------------------------------------------------------------------------
*
*   Section 3:    Concentration and organic matter in suspended solids
*--------------------------------------------------------------------------
*
*   concentration of suspended solids, ss
  coss = 50.                ! unit: g/m3       range: 1. .... 100,000.
*   mass ratio of organic matter, m_om,ss
  raomss = 0.10             ! unit: -          range: 0. .... 1.
*
*--------------------------------------------------------------------------
*
*   Section 4:    Amount of macrophytes
*--------------------------------------------------------------------------
*
*   dry weight of macrophyte biomass per m^2 bottom, DW
  dwmp = 250.               ! unit: g/m^2      range: 0. .... 1000.
*
*----------------End of file-----------------------------------------------
```

```
*    Filename :   WLST.INP
*    Content  :   Input data for TOXSWA concerning initial concentrations
*                 in the water layer
*
*-----------------------------------------------------------------------
*
*    Section 1:   Initial concentrations
*-----------------------------------------------------------------------
*
*    initial (start) mass concentration of pesticide in water layer, c^*,
*    for the total number of nodes in x direction (nxnotot, so buffers
*    included)
  castwl = 0.                    ! unit: g/m^3     range: 0. ... 100.0
           0.
           0.
           8.264e-03
           8.264e-03
           8.264e-03
           8.264e-03
           8.264e-03
           8.264e-03
           8.264e-03
             8.264e-03
             8.264e-03
             8.264e-03
             8.264e-03
             8.264e-03
             8.264e-03
             8.264e-03
             8.264e-03
             8.264e-03
             8.264e-03
               8.264e-03
               8.264e-03
               8.264e-03
               8.264e-03
               8.264e-03
               8.264e-03
               8.264e-03
               8.264e-03
               8.264e-03
*
*-----------------------------------------------------------------------
*
*    Section 2:   Constant background concentration in air
*-----------------------------------------------------------------------
*
*    constant background concentration of pesticide in air
  coair = 0.                     ! unit: g/m^3     range: 0.
*
*-----------------End of file-------------------------------------------
```

```
*   Filename :   WBNU.INP
*   Content  :   Input data for TOXSWA concerning numerical solution and
*                thickness of the sediment considered
*
*----------------------------------------------------------------------
*
*   Section 1:   Numerical weight factors
*----------------------------------------------------------------------
*
*   numerical weight factor for space
  betawb = 0.5              ! unit: -          range: 0.0 .... 1.0
*   numerical weight factor for time
  thetawb = 1.0             ! unit: -          range: 0.0 .... 1.0
*
*----------------------------------------------------------------------
*
*   Section 2:   Time parameters
*----------------------------------------------------------------------
*
*   selected time step for sediment
  deltwb = 100.             ! unit: s          range: 1. .... 86,400.
*
*----------------------------------------------------------------------
*
*   Section 3:    Space parameters
*----------------------------------------------------------------------
*
*   length of sediment considered (end buffer excluded)
  zwb = 0.10                ! unit: m          range: 0.01 .... 0.5
*   length of end buffer sediment (0. if none)
  zebb = 0.                 ! unit: m          range: 0.0 .... 0.1
*
*----------------------------------------------------------------------
*
*   Section 4:    Number of space nodes
*----------------------------------------------------------------------
*
*   number of nodes in sediment (end buffer excluded)
  nznowb = 23               ! unit: -          range: 10 .... 200
*   number of nodes in end buffer (0 if none)
  nznoebb = 0               ! unit: -          range: 0 .... 25
*
*----------------------------------------------------------------------
*
*   Section 5:    Location of space nodes
*----------------------------------------------------------------------
*
*   z coordinate of nodes in sediment
  zcdwb = 0.0005            ! unit: m          range: 0.0001 .... 0.6
          0.0015
          0.0025
          0.0035
          0.0045
          0.0060
          0.0080
          0.0100
          0.0120
          0.0145
           0.0175
           0.0205
           0.0240
           0.0280
           0.0325
           0.0375
           0.0425
           0.0475
           0.0550
           0.0650
            0.0750
            0.0850
            0.0950
*   z coordinate of nodes in end buffer (0. if none)
  zcdebb =  0.0             ! unit: m          range: 0.0 .... 0.6
*
*-------------End of file----------------------------------------------
```

```
*    Filename :   WBPA.INP
*    Content  :   Input data for TOXSWA concerning characteristics of
*                 the sediment
*
*------------------------------------------------------------------------
*
*    Section 1:   Physical properties of sediment
*------------------------------------------------------------------------
*
*    bdwb(1-nznowb), bulk density of dry sediment material, rho_b (as a
*    function of depth, end buffer excluded)
*
*    por(1-nznowb), porosity (volume fraction of void water), epsilon (as
*    a function of depth, end buffer excluded)
*
*    tor(1-nznowb), tortuosity, lamba (as a function of depth, end buffer
*    excluded)
*
  bdwb                         por                   tor
  400.                         0.80                  0.80
  400.                         0.80                  0.80
  400.                         0.80                  0.80
  400.                         0.80                  0.80
  400.                         0.80                  0.80
  400.                         0.80                  0.80
  400.                         0.80                  0.80
  600.                         0.75                  0.75
  800.                         0.70                  0.70
 1000.                         0.65                  0.65
   1200.                         0.60                  0.60
   1300.                         0.55                  0.55
   1400.                         0.50                  0.50
   1400.                         0.50                  0.50
   1500.                         0.50                  0.50
   1500.                         0.45                  0.42
   1500.                         0.40                  0.34
   1500.                         0.40                  0.34
   1500.                         0.40                  0.34
   1500.                         0.40                  0.34
    1500.                         0.40                  0.34
    1500.                         0.40                  0.34
    1500.                         0.40                  0.34
! kg/m^3                       -                     -            unit
! 10. .... 3000.               0. .... 1.            0. .... 1.   range
*
*------------------------------------------------------------------------
*
*    Section 2:   Organic matter in sediment
*------------------------------------------------------------------------
*
*    mass ratio of organic matter of dry sediment material, m_om,wb (as a
*    function of depth, end buffer excluded)
  raomwb = 0.08                ! unit: -           range: 0. .... 1.
           0.08
           0.08
           0.08
           0.08
           0.08
           0.08
           0.07
           0.06
           0.05
            0.04
            0.03
            0.03
            0.03
            0.02
            0.02
            0.01
            0.01
            0.01
            0.01
             0.01
             0.005
             0.005
*
```

```
*------------------------------------------------------------------------
*
*   Section 3:    Dispersion in sediment
*------------------------------------------------------------------------
*
*   dispersion length
 ldis = 0.015               ! unit: m            range: 0. ..... 1.
*
*----------------End of file---------------------------------------------
```

```
*     Filename :    WBST.INP
*     Content  :    Input data for TOXSWA concerning initial concentrations
*                   in sediment
*
*------------------------------------------------------------------------------
*
*     Section 1:    Initial concentrations
*------------------------------------------------------------------------------
*
*     initial (start) mass concentration of pesticide in sediment, c_b^*,
*     for the total number of nodes in z direction (nznotot, so end
*     buffer included)
   castwb = 0.                   ! unit: g/m^3      range: 0. .... 1000.0
            0.
            0.
            0.
            0.
            0.
            0.
            0.
            0.
            0.
             0.
             0.
             0.
             0.
             0.
             0.
             0.
             0.
             0.
             0.
              0.
              0.
              0.
*
*----------------End of file-------------------------------------------------
```

```
*   Filename :   HY.INP
*   Content  :   Input data for TOXSWA concerning hydrological
*                characteristics
*
*----------------------------------------------------------------------
*
*   Section 1:   Flow velocity and water depth of water layer
*----------------------------------------------------------------------
*
*   constant flow velocity in ditch
 u = 100.                ! unit: m/d        range: -100,000.
                         !                         .... +100,000.
*   constant water depth in ditch
 wdh = 0.50              ! unit: m          range: 0.1 .... 2.0
*
*----------------------------------------------------------------------
*
*   Section 2:   Dispersion coefficient
*----------------------------------------------------------------------
*
*   dispersion coefficient of pesticide in water, E_x
 kds = 600.              ! unit: m^2/d      range: 10. .... 100,000.
*
*----------------------------------------------------------------------
*
*   Section 3:   Seepage with concentration
*----------------------------------------------------------------------
*
*   qseif:
*   seepage from neighbouring lot to ditch or seepage from ditch to
*   neighbouring lot (negative and positive, resp.) expressed per m^2
*   lot for 30 days
*
*   colot:
*   concentration of pesticide in incoming water seeping from
*   neighbouring lot into the ditch bottom (0. in the case of seepage
*   from ditch into neighbouring lot)
*
 qseif                   colot
  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.
    0.                      0.
    0.                      0.
    0.                      0.
    0.                      0.
    0.                      0.
    0.                      0.
    0.                      0.
    0.                      0.
    0.                      0.
    0.                      0.
! m^3/m^2.d              g/m^3         unit
! -10.E-03 .... +10.E-03  0. .... 1.   range
*
*----------------End of file-------------------------------------------
```

```
*   Filename :   SU.INP
*   Content  :   Input data for TOXSWA concerning the substance
*
*-----------------------------------------------------------------------
*
*   Section 1:   Transformation in water layer
*-----------------------------------------------------------------------
*
*   rate coefficient for transformation in water layer, k (= ln2/DT_50)
 krtfwl = 0.0092           ! unit: 1/d        range: 0. .... 50.
*
*-----------------------------------------------------------------------
*
*   Section 2:   Sorption to suspended solids
*-----------------------------------------------------------------------
*
*   sorption coefficient based at organic matter content, K_om,ss,
*   (distribution coefficient)
 kdomssdit = 16.4          ! unit: m^3/kg      range: 0. .... 100.
*   concentration of pesticide at which the K_om of the suspended solids
*   has been observed, c_e,ss
 coobkomss = 1.0E-06       ! unit: kg/m^3      range: 1.0E-08 .... 0.1
*   Freundlich exponent for sorption to suspended solids, n_ss
 exfrss = 0.984            ! unit: -           range: 0.1 .... 2.0
*
*-----------------------------------------------------------------------
*
*   Section 3:   Sorption to macrophytes
*-----------------------------------------------------------------------
*
*   slope of sorption isotherm based at dry weight macrophytes, K_mp,
*   (distribution coefficient)
 kdmpdit = 2.0             ! unit: m^3/kg     range: 0. .... 100.
*
*
*-----------------------------------------------------------------------
*
*   Section 4:   Volatilisation
*-----------------------------------------------------------------------
*
*   transport coefficient of pesticide in liquid phase, k_l
 klq = 1.70                ! unit: m/d        range: 0.1 .... 10,000.
*   transport coefficient of pesticide in gas phase, k_g
 kga = 163.1               ! unit: m/d        range: 0.1 .... 10,000.
*   saturated gas pressure of pesticide, P
 psat = 0.0025             ! unit: Pa         range: 0.00001 .... 0.1
*   molecular mass of pesticide, M
 mamol = 350.6             ! unit: g/mol      range: 10. .... 10,000.
*   temperature during observation of saturated vapour pressure and
*   solubility with which Henry coefficient is calculated, T
 tekhe = 298.              ! unit: K          range: 200. .... 350.
*   solubility of pesticide in water, c_sol
 cosol = 2.0               ! unit: g/m^3      range: 0.001 .... 100.
*
*-----------------------------------------------------------------------
*
*   Section 5:   Exchange water layer - sediment
*-----------------------------------------------------------------------
*
*   diffusion coefficient of pesticide in water, D_w
 kdfw = 40.                ! unit: mm^2/d     range: 1. .... 200.
*
*-----------------------------------------------------------------------
*
*   Section 6:   Transformation in sediment
*-----------------------------------------------------------------------
*
*   rate coefficient for transformation in sediment, k_b (= ln2/DT_50)
 krtfwb = 0.0040           ! unit: 1/d        range: 0. .... 50.
*
```

```
*-----------------------------------------------------------------------
*
*  Section 7:  Sorption to sediment
*-----------------------------------------------------------------------
*
*   sorption coefficient based at organic matter content of sediment
*   material K_om,wb, (distribution coefficient)
  kdomwb1 = 16.4           ! unit: m^3/kg      range: 0. .... 100.
*   concentration of pesticide at which the K_om of the sediment
*   material has been observed, c_e,wb
  coobkomwb = 1.0E-06      ! unit: kg/m^3      range: 1.0E-08 .... 0.1
*   Freundlich exponent for sorption to sediment material, n_wb
  exfrwb = 0.984           ! unit: -           range: 0.1 .... 2.0
*
*-----------------End of file-------------------------------------------
```

Annex 19 Source code of the TOXSWA program, version 1.0, including a guide to the vocabulary used

List of letters and combinations of letters which are used more than once in the source code of TOXSWA 1.0 to form the names of the variables and parameters

a	all
ad	advection
ai(r)	air
all	all sediment subsystems
av	average
b	sediment [water bottom]
bd	bulk density
bot	bottom
c	concentration (in water or liquid phase)
ca	total concentration, c^* or c_b^*
cd	coordinate
cl	calculation
co	concentration
cu	accumulated (over time)
db	distribution
dd	distribution in ditch
del(t)	delta Δ, difference
df	diffusion
dh	depth
dit	ditch
ds	dispersion
dw	dry weight
eb	end buffer (of water layer)
ebb	end buffer of water bottom
en	end-side node
er	error
ex	exponent
f	front
fb	front buffer (of water layer)
fd	flux dispersion exceeds advection (in sediment)
fl	flux
fn	front-side node
fr	Freundlich
ga	gass
he	Henry
hw	water level defining the exchanging perimeter $P_{z=0}$
hy	hydrology
i	at node i
ic	input concentration
imh	at node i-½
in	input or incoming
iph	at node i+½
im1	at node i-1
ip1	at node i+1
j	at time j
jmh	at time j-½
jph	at time j+½
jp1	at time j+1
k	coefficient
kd	distribution coefficient
kis1	k = 1
kr	rate coefficient
l	water layer
lbd	band above the diagonal of the left-hand tridiagonal matrix
ldd	diagonal of the left-hand tridiagonal matrix

le	length
lin	lineic
lm	lineic mass
lo	loop
lod	band under the diagonal of the left-hand tridiagonal matrix
lot	neighbouring field lot
lq	liquid
m	mass
mb	mass balance
mn	minimum
mp	macrophytes
mx	maximum
n	number or numerical
no	node(s)
node	node(s)
nu	numerical
nww	numerical weight factor water layer
nwb	numerical weight factor sediment
nx	number in x direction
nz	number in z direction
ob	observed
old	of past time step
om	organic matter
op	option(s)
out	output
p	parameter
pe	perimeter
per	percentage or percolated
por	porosity
prc	percentage
pt	point of time
qseif	seepage/infiltration flux
qu	quantity
qvo	discharge volume
ra	ratio
rbd	band above the diagonal of the right-hand tridiagonal matrix
rdd	diagonal of the right-hand tridiagonal matrix
rh	right hand
rod	band under the diagonal of the right-hand tridiagonal matrix
rs	rate of substance
rv	right-hand vector
s	start
sisl	side slope
so	sorbed or solution
ss	suspended solids
st	start, beginning
su	substance
sum	sum of
sy	systems
t	time
tf	transformation
top	top layer in sediment or top of water level in ditch
tor	tortuosity
tot	total
ts	time steps
tt	iteration
u	flow velocity in ditch
ui	unit number for input file
uo	unit number for output file
us	upward seepage
var	variable
vol	volatilisation
w	water, wetted
war	wetted area
wb	sediment [water bottom]

wdh	water depth
wi	width
wl	water layer
x	x direction
xp	exposure
z	z direction

```
      program TOXSWA

** PROGRAM:
**    toxswa.for - fate of pesticides in small surface waters
**    author     - P.I. Adriaanse
**
** SYNOPSIS:
**    run TOXSWA
**
** DESCRIPTION:
**    Main program of the TOXSWA simulation model; it calculates
**    (i) pesticide concentration profiles in water layer (water phase,
**    sorbed to suspended solids and sorbed to macrophytes) and in
**    sediment (pore water and sorbed to solid bottom material),
**    (ii) mass balances for the water and sediment layer and
**    (iii) distribution of pesticide mass between the different
**    compartments by calling different subroutines.
**
**    References:
**    Adriaanse, P.I., 1996. Fate of pesticides in field ditches: the
**      TOXSWA simulation model. Report 90, DLO Winand Staring Centre,
**      Wageningen.
**    Beltman, W.H.J., P.I. Adriaanse and M. van Elswijk, 1996. User's
**      manual TOXSWA 1.0. Technical Document 33. DLO Winand Staring
**      Centre, Wageningen.
**
**    Restrictions for use:
**    Use is restricted according to the 'General Terms and Conditions
**    as to the Making Available of Computer Software', dated 1 November
**    1990 of the DLO Winand Staring Centre, Wageningen.
**
** HISTORY:
**    10 April 1996  P.I. Adriaanse
**                   version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

c                              common variables
      include 'common.for'
c                              local variables
      real     cowbjp1_2d(mxnxnotot, mxnznotot)
      logical  allconczero


c                              options for model output
      call opout
c                              general and initial input water layer
      call wlin
c                              general and initial input sediment
      call wbin
c                              hydrological input data water layer and
c                              sediment
      call hyin
c                              characteristics substance
      call suin
c                              initial calculations water layer
      call wlit
c                              initial calculations sediment
      call wbit


c                              timestep loop
      do 90 its = 1, ntswl
c                              solution water layer
         call wlso
c                              mass balance water layer
         call wlmb
c                              output mass balance selected node wl
         call wlmbnode
c                              sediment subsystems loop (under each water
c                              layer segment ixnotot there is a sediment
c                              subsystem)
         do 50 ixnotot = nxnofb+1, nxnofb+nxnodit

c                              fill array cowbj(1-nznotot) with right
c                              values under this node of water layer
            do 30 kznotot = 1, nznotot
               if (its.eq.1) then
                  cowbj(kznotot) = costwb(kznotot)
               else
                  cowbj(kznotot) = cowbjp1_2d(ixnotot, kznotot)
               end if
   30       continue
c                              solution sediment
            call wbso
c                              mass balance sediment
            call wbmb
c                              output partition mass wl+wb
            call dbout
c                              sediment subsystems loop
c                              fill two-dimensional array cowbjp1_2d
            do 40 kznotot = 1, nznotot
               cowbjp1_2d(ixnotot, kznotot) = cowbjp1(kznotot)
   40       continue
c                              wb subsystems loop
   50    continue

c ---- Program stops when all concentrations are zero (below 1E-20)

         allconczero = .true.
         do 60 ixnotot = 1, nxnotot
            if (cowljp1(ixnotot).gt.1.0E-20) then
               allconczero = .false.
            end if
   60    continue

         do 80 ixnotot = nxnofb+1, nxnofb+nxnodit
            do 70 kznotot = 1, nznotot
               if (cowbjp1_2d(ixnotot, kznotot).gt.1.0E-20) then
                  allconczero = .false.
               end if
   70       continue
   80    continue

         if (allconczero) then
            print*, ' All concentrations in water layer and in'
            print*, ' sediment are zero (less than 1E-20).'
            print*, ' Programs stops.'
            stop
         end if


c                              time steps loop
   90 continue


      print*, '     end of simulation'

      stop

      end


**
** COMMON BLOCK:
**    common.for - contains declarations of all common variables for
**                 the TOXSWA program
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    This common block contains all the common varables of TOXSWA; it
**    is included in all the subroutines of the TOXSWA program.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

      implicit none
```

```fortran
c                             array dimensions assigned by parameter
c                             statements
      include 'param.for'

c                             common variables
      character  dmy*10, hms*10

      logical  wantoutp

      integer  nts, ntswl, ntswb,
     &         uiwn, uiwp, uiws, uibn, uibp, uibs, uoin, uowl, uowb,
     &         uoer, uocl,
     &         uomw, uomn, uihy, uisu, uioo, uob1, uob2, uocb, uomb,
     &         uoma, uodd, nxnodit, nxnofb, nxnoeb, nxnotot,
     &         nznowb, nznoebb, nznotot, nwbsy, iwbsy(mxnxnotot),
     &         nditout, itsout(9), ktop, noldit,
     &         its, ixnotot, kznotot,
     &         op_input, op_icwlhy, op_icwb, op_wlmb, op_wlmbnodenr,
     &         op_wbsconodenr, op_wbmbnodenr, op_wbmball, op_dbnodenr,
     &         op_dbdit

      real     betawl, thetawl,
     &         deltwl, ttot, ptditout(9),
     &         xdit, xfb, xeb,
     &         xcdfb(mxnxnofb), xcddit(mxnxnodit), xcdeb(mxnxnoeb),
     &         xcd(mxnxnotot),
     &         wibot, sisl, wdhfl,
     &         leplot, krtfwl,
     &         coss, raomss, kdomssdit, coobkomss, exfrss,
     &         kdomss(mxnxnotot), kdfrss(mxnxnotot),
     &         dwmp, kdmpdit, kdmp(mxnxnotot),
     &         klq, kga, psat, mamol, unga, tekhe, cosol, khe, ktl,
     &         kdfw, kds,
     &         castwl(mxnxnotot), coair,
     &         costwl(mxnxnotot), sompst(mxnxnotot), sossst(mxnxnotot),
     &         sossjp1(mxnxnotot), sompjp1(mxnxnotot),
     &         cawljp1(mxnxnotot), cowljp1(mxnxnotot), cowlj(mxnxnotot),
     &         ufb, ueb
      real    lod(mxnxnotot), ldd(mxnxnotot), lbd(mxnxnotot),
     &        rod(mxnxnotot), rdd(mxnxnotot), rbd(mxnxnotot),
     &        rv(mxnxnotot), oldcowljp1(mxnxnotot),
     &        kdswcljimh(mxnxnotot), kdswcljiph(mxnxnotot),
     &        lodad, lodds, lddadmh, lddadph, ldddsmh, ldddsph, lddca,
     &        lddtf, lddai, lddwbad, lddwbdf, lbdad, lbdds,
     &        rodad, rodds, rddadmh, rddadph, rdddsmh, rdddsph, rddca,
     &        rddtf, rddai, rddwbad, rddwbdf, rbdad, rbdds,
     &        rvai, rvwbdf, rvwbad

      real     delx(mxnxnotot), u,
     &         wdhst(mxnxnotot),
     &         kdswnu, kdswcl,
     &         nwwmh(mxnxnotot), nwwph(mxnxnotot),
     &         pez0hw,
     &         colot(100), colotj, colotjp1,
     &         kdfwbmh(mxnznotot), kdfwbph(mxnznotot),
     &         cowbjp1kisl(mxnxnotot),
     &         cowbjkisl(mxnxnotot),
     &         qseif(100), qseifj, qseifjph,
     &         tcoxp(4), sumct(9), coxp(9,4)

      real     delz(mxnznotot),
     &         qvojimh, qvojiph, qvojp1imh, qvojp1iph,
     &         wdhji, wdhjim1, wdhjip1, wdhjp1i, wdhjp1im1, wdhjp1ip1,
     &         warj(mxnxnotot), warjim1, warjip1, warjph(mxnxnotot),
     &         warjp1(mxnxnotot), warjp1im1, warjp1ip1,
     &         warjiph(mxnxnotot), warjp1iph,
     &         warjimh(mxnxnotot), warjp1imh,
     &         warjphimh(mxnxnotot), warjphiph(mxnxnotot) ,
     &         qvojphimh(mxnxnotot), qvojphiph(mxnxnotot),
     &         witopj(mxnxnotot), witopjph(mxnxnotot),
     &        uditin, uditout, wdhditin, wdhditout, qvoditin, qvoditout

      real     betawb, thetawb,
     &         deltwb,
     &         zwb, zebb,
     &         zcdwb(mxnznowb), zcdebb(mxnznoebb), zcd(mxnznotot),
     &         krtfwb,
     &         bdwb(mxnznotot), bdwbmh(mxnznotot), bdwbph(mxnznotot),
     &         por(mxnznotot), pormh(mxnznotot),
     &         porph(mxnznotot), tor(mxnznotot),
     &         coobkomwb, exfrwb, raomwb(mxnznotot),
     &         kdomwb(mxnznotot), kdfrwb(mxnznotot),
     &         kdfrwbmh(mxnznotot), kdfrwbph(mxnznotot),
     &         ldis,
     &         kdswbjkmh(mxnznotot), kdswbjkph(mxnznotot),
     &         kdsbnujkmh(mxnznotot), kdsbnujkph(mxnznotot),
     &         kdsbcljkmh(mxnznotot), kdsbcljkph(mxnznotot)
      real     pe(mxnznotot), pemh(mxnznotot), peph(mxnznotot),
     &         castwb(mxnznotot), costwb(mxnznotot), sowbst(mxnznotot),
     &         nwbmh(mxnznotot), nwbph(mxnznotot),
     &         cowbj(mxnznotot),
     &         cowbjp1(mxnznotot), oldcowbjp1(mxnznotot),
     &         cowbjkmh(mxnznotot), cowbjkph(mxnznotot),
     &         sofdfactormh(mxnznotot), sofdfactorph(mxnznotot),
     &         sowbjp1(mxnznotot), cawbjp1(mxnznotot)
      real     blod(mxnznotot), bldd(mxnznotot), blbd(mxnznotot),
     &         brod(mxnznotot), brdd(mxnznotot), brbd(mxnznotot) ,
     &         brv(mxnznotot), brh(mxnznotot),
     &         blodad, bloddsdf, blddadph, blddsdfph,
     &         blddadmh, blddsdfmh, blddca, blddtf, blbdad, blbddsdf,
     &         brodad, broddsdf, brddadph, brdddsdfph, brddadmh,
     &         brdddsdfmh, brddca, brddtf, brbdad, brbddsdf

      real     zero
c                             common blocks
      common/char/ dmy, hms

      common/logi/ wantoutp

      common/intwlwb/ nts, ntswl, ntswb,
     &           uiwn, uiwp, uiws, uibn, uibp, uibs, uoin, uowl, uowb,
     &           uoer, uocl,
     &           uomw, uomn, uihy, uisu, uioo, uob1, uob2, uocb, uomb,
     &           uoma, uodd, nxnodit, nxnofb, nxnoeb, nxnotot,
     &           nznowb, nznoebb, nznotot, nwbsy, iwbsy,
     &           nditout, itsout, ktop, noldit,
     &           its, ixnotot, kznotot,
     &           op_input, op_icwlhy, op_icwb, op_wlmb, op_wlmbnodenr,
     &           op_wbsconodenr, op_wbmbnodenr, op_wbmball,
     &           op_dbnodenr, op_dbdit

      common/realwl/ betawl, thetawl,
     &            deltwl, ptditout,
     &            xdit, xfb, xeb,
     &            xcdfb, xcddit, xcdeb,
     &            xcd,
     &            krtfwl,
     &            coss, raomss, kdomssdit, coobkomss, exfrss,
     &            kdomss, kdfrss,
     &            dwmp, kdmpdit, kdmp,
     &            klq, kga, psat, mamol, unga, tekhe, cosol, khe, ktl,
     &            kdfw, kds,
     &            castwl, coair,
     &            costwl, sompst, sossst,
     &            delx, u,
     &            wdhst,
     &            kdswnu, kdswcl,
     &            nwwmh, nwwph,
     &            sossjp1, sompjp1, cawljp1, cowljp1, cowlj,
     &            ufb, ueb
      common/realwl2/ lod, ldd, lbd, rod, rdd, rbd, rv,
     &           oldcowljp1,
     &           kdswcljimh, kdswcljiph,
     &           lodad, lodds, lddadmh, lddadph, ldddsmh, ldddsph,
     &           lddca, lddtf, lddai, lddwbad, lddwbdf, lbdad, lbdds,
     &           rodad, rodds, rddadmh, rddadph, rdddsmh, rdddsph,
     &           rddca, rddtf, rddai, rddwbad, rddwbdf, rbdad, rbdds,
     &           rvai, rvwbdf, rvwbad

      common/realwlwb/ ttot,
     &              wibot, sisl, wdhfl,
     &              leplot,
     &              pez0hw,
     &              colot, colotj, colotjp1,
     &              kdfwbmh, kdfwbph,
     &              cowbjp1kisl,
     &              cowbjkisl,
     &              qseif, qseifj, qseifjph,
```

```
     &                tcoxp, sumct, coxp

      common/realwlwb2/  qvojimh, qvojiph, qvojplimh, qvojpliph,
     &                wdhji, wdhjim1, wdhjip1, wdhjpli,
     &                wdhjplim1, wdhjplip1,
     &                warj, warjim1, warjip1,
     &                warjp1, warjplim1, warjplip1, warjph,
     &                warjiph, warjpliph, warjimh, warjplimh,
     &                warjphimh, warjphiph,
     &                qvojphimh, qvojphiph, witopj, witopjph,
     &                zero, delz,
     &                uditin, uditout, wdhditin, wdhditout,
     &                qvoditin, qvoditout

      common/realwb/ betawb, thetawb,
     &              deltwb,
     &              zwb, zebb,
     &              zcdwb, zcdebb, zcd,
     &              krtfwb,
     &              bdwb, bdwbmh, bdwbph, por, pormh, porph, tor,
     &              kdomwb, kdfrwb, kdfrwbmh, kdfrwbph,
     &              coobkomwb, exfrwb, raomwb,
     &              ldis,
     &              kdswbjkmh, kdswbjkph, kdsbnujkmh, kdsbnujkph,
     &              kdsbcljkmh, kdsbcljkph
      common/realwbl/ pe, pemh, peph,
     &              castwb, costwb, sowbst,
     &              nwbmh, nwbph,
     &              cowbj, cowbjp1, oldcowbjp1, cowbjkmh, cowbjkph,
     &              sofdfactormh, sofdfactorph,
     &              sowbjp1, cawbjp1
      common/realwb2/ blod, bldd, blbd, brod, brdd, brbd, brv, brh,
     &          blodad, bloddsdf, blddadph, bldddsdfph,
     &          blddadmh, bldddsdfmh, blddca, blddtf, blbdad, blbddsdf,
     &          brodad, broddsdf, brddadph, brdddsdfph, brddadmh,
     &          brdddsdfmh, brddca, brddtf, brbdad, brbddsdf


**
** FILE:
**    param.for - include-file, it defines parameters that are used for
**                the declaration of the dimensions of certain arrays
**                for the TOXSWA program
**    author    - P.I. Adriaanse
**
** DESCRIPTION:
**    This include-file is included in the common block of the TOXSWA
**    program
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

      integer  mxnxnofb, mxnxnodit, mxnxnoeb, mxnxnotot,
     &         mxnznowb, mxnznoebb, mxnznotot
      real     tiny

c                        maximum number nodes in front buffer wl
      parameter (mxnxnofb = 20)
c                        maximum number nodes in ditch
      parameter (mxnxnodit = 100)
c                        maximum number nodes in end buffer wl
      parameter (mxnxnoeb = 20)
c                        maximum number nodes in ditch and 2 buffers
      parameter (mxnxnotot = 140)

c                        maximum number nodes in sediment
c                                              (end buffer excluded)
      parameter (mxnznowb = 100)
c                        maximum number nodes in end buffer wb
      parameter (mxnznoebb = 20)
c                        maximum number nodes in sediment
c                                              (end buffer included)
      parameter (mxnznotot = 120)
c                        tiny will be used to prevent dividing by
c                        zero or raising a (tiny) negative number
c                        to a power
      parameter (tiny = 1.0E-25)


      block data

**
** DATA BLOCK:
**    block.for - initialises identifyers of the TOXSWA program
**    author    - P.I. Adriaanse
**
** DESCRIPTION:
**    This data block initialises identifyers of TOXSWA
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

c                        common variables
      include 'common.for'
c                        local variables

c                        unit numbers input files
      data uioo, uiwn, uiwp, uiws, uibn, uibp, uibs, uihy, uisu
     &    /  81,   83,   85,   87,   89,   91,   93,   95,   97/
c                        unit numbers output files
      data uoer, uoin, uowl, uowb, uocl, uomw, uoma, uodd
     &    /  21,   22,   23,   24,   25,   26,   27,   28/

      data zero  / 0./

      end



      subroutine opout

** SUBROUTINE
**    opout.for - input data for the TOXSWA program about options for
**                obtaining model output are read
**    author    - P.I. Adriaanse
**
** DESCRIPTION:
**    In this subroutine it is read which output the user wants of the
**    TOXSWA model.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

c                        common variables
      include 'common.for'
c                        local variables
      integer     i,k, helpnditout, helpnwbsy


c---- statement for Vax Fortran 77 compiler for date and time
c
c      call date (dmy)
c      call time (hms)


c                          open file for error messages
      open (unit = uoer, file = 'message.out', status = 'unknown')
      write (uoer, 10) dmy, hms
   10 format (' TOXSWA simulation:', 2A10, /,
```

```
     &        ' Messages (error/warning/ttutil):'/)

C---- input data from file opout.inp

      call rdinit (uioo, uoer, 'opout.inp')


C                               selected output files
      call rdsinr ('op_input', 0, 1, op_input)
      call rdsinr ('op_icwlhy', 0, 1, op_icwlhy)
      call rdsinr ('op_icwb', 0, 1, op_icwb)
      call rdsinr ('op_wlmb', 0, 1, op_wlmb)
      call rdsinr ('op_wlmbnodenr', 0, 1, op_wlmbnodenr)
      call rdsinr ('op_wbsconodenr', 0, 1, op_wbsconodenr)
      call rdsinr ('op_wbmbnodenr', 0, 1, op_wbmbnodenr)
      call rdsinr ('op_wbmball', 0, 1, op_wbmball)
      call rdsinr ('op_dbnodenr', 0, 1, op_dbnodenr)
      call rdsinr ('op_dbdit', 0, 1, op_dbdit)

      if (op_input.eq.1) then
         open (unit = uoin, file = 'input.out', status = 'unknown')
         write (uoin, 20) dmy, hms
   20 format ('TOXSWA simulation: ', 2A10, /, ' Output of input:', //,
     &        ' All input from subroutine opout:')
      end if

      if (op_input.eq.1) then
         write (uoin, 30)
   30 format (' Section 1: Selected output files:      (no dimensions)')
         write (uoin, *) op_input
         write (uoin, *) op_icwlhy
         write (uoin, *) op_icwb
         write (uoin, *) op_wlmb
         write (uoin, *) op_wlmbnodenr
         write (uoin, *) op_wbsconodenr
         write (uoin, *) op_wbmbnodenr
         write (uoin, *) op_wbmball
         write (uoin, *) op_dbnodenr
         write (uoin, *) op_dbdit
      end if


C                               points of time for output
      call rdsinr ('nditout', 1, 9, nditout)
      call rdarer ('ptditout', 0., 100., ptditout, 9, helpnditout)
      if (helpnditout.ne.nditout) then
         write (uoer, 40)
   40 format (//, ' Error in input opout.inp,', /,
     &        ' actual length array ptditout is not equal', /,
     &        ' to declared length, please correct data')
      end if

      if (op_input.eq.1) then
         write (uoin, 41)
   41 format (' Section 2: Points of time for output:        (d)')
         write (uoin, *) nditout
         write (uoin, *) (ptditout(i), i = 1, nditout)
      end if


C                               selected sediment subsystems for output
      call rdsinr ('nwbsy', 1, 9, nwbsy)
      call rdainr ('iwbsy', 1, 200, iwbsy, 9, helpnwbsy)
      if (helpnwbsy.ne.nwbsy) then
         write (uoer, 50)
   50 format (//, ' Error in input opout.inp,', /,
     &        ' actual length array iwbsy is not equal',
     &        /, ' to declared length, please correct data')
      end if
      call rdsinr ('ktop', 1, 30, ktop)

      if (op_input.eq.1) then
         write (uoin, 51)
   51 format (' Section 3: Selected sediment subsystems for output:',
     &        '    (no dimensions)')
         write (uoin, *) nwbsy
         write (uoin, *) (iwbsy(k), k = 1, nwbsy)
         write (uoin, *) ktop
      end if

      close (uioo)

      if (op_input.eq.1) then
         close (unit = uoin)
      end if


      return


      end



      subroutine wlin

** SUBROUTINE:
**    wlin.for - all input data of the water layer for the TOXSWA
**               program are read
**    author   - P.I. Adriaanse
**
** DESCRIPTION:
**    Input data for water layer concerning numerical aspects, the ditch
**    geometry and other ditch characteristics and the initial
**    concentrations are read; the data are converted to SI-units
**    (m, s, g, Pa, K, J, mol).
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                               common variables
      include 'common.for'
C                               local variables
      integer    i, helpnxnofb, helpnxnodit, helpnxnoeb,
     &           helpnxnotot


      if (op_input.eq.1) then
         open (unit = uoin, file = 'input.out', status = 'unknown')
         write (uoin, 11) dmy, hms
   11 format (' TOXSWA simulation:', 2A10, /' Output of input:'/)
         write (uoin, 12)
   12 format (/, ' All input data from wlin:')
      end if

      if (op_icwlhy.eq.1) then
         open (unit = uowl, file = 'icwlhy.out', status = 'unknown')
         write (uowl, 13) dmy, hms
   13 format (' TOXSWA simulation:', 2A10, /,
     &    ' Output of initial calculations water layer and hydrology:',
     &        /)
      end if


C---- input data (numerical aspects) from file wlnu.inp

      call rdinit (uiwn, uoer, 'wlnu.inp')


C                               numerical weight factors
      call rdsrer ('betawl', 0.0, 1.0, betawl)
      call rdsrer ('thetawl', 0.0, 1.0, thetawl)
      if (op_input.eq.1) then
         write (uoin, 20)
   20 format (/, ' Input from wlnu.inp:', /,
     &        ' Section 1: Numerical weight factors',
     &        '    (no dimensions)')
         write (uoin, *) betawl, thetawl
      end if


C                               time parameters
      call rdsrer ('deltwl', 1., 86400., deltwl)
```

```
      call rdsrer ('ttot', 0.1, 100., ttot)
      if (op_input.eq.1) then
         write (uoin,30)
   30 format (' Section 2: Time parameters    (s, resp d)')
         write (uoin, *) deltwl, ttot
      end if
      ttot = ttot*86400.
      ntswl = int (ttot/deltwl) + 1

C                            calculation of timesteps, itsout(i) for
C                            output with the aid of points of time
C                            ptditout(i) of subroutine opout
      if (ptditout(1).le.zero) then
         itsout(1) = 1
      else
         itsout(1) = nint (ptditout(1)*86400./deltwl)
      end if

      do 31 i = 2, nditout
         itsout(i) = nint (ptditout(i)*86400./deltwl)
   31 continue

      if (op_icwlhy.eq.1) then
         write (uowl, 32)
   32 format (/, ' Results of initial calculations wlin:')
         write (uowl, 33) nditout
   33 format (' There are ', I6, ' timesteps for output, these are: ')
         write (uowl, *) (itsout(i), i = 1, nditout)
      end if


C                            space parameters
      call rdsrer ('xdit', 10., 10000., xdit)
      call rdsrer ('xfb', 0., 1000., xfb)
      call rdsrer ('xeb', 0., 1000., xeb)
      if (op_input.eq.1) then
         write (uoin, 40)
   40 format (' Section 3: Space parameters      (m)')
         write (uoin, *) xdit, xfb, xeb
      end if


C                            number space nodes
      call rdsinr ('nxnodit', 10, 100, nxnodit)
      call rdsinr ('nxnofb', 0, 20, nxnofb)
      call rdsinr ('nxnoeb', 0, 20, nxnoeb)
      if (op_input.eq.1) then
         write (uoin, 50)
   50 format (' Section 4: Number space nodes   (no dimensions)')
         write (uoin, *) nxnodit, nxnofb, nxnoeb
      end if


C                            location space nodes
      if (nxnofb.eq.0) then
C                            dummy statement
         nxnofb = 0
      else
         call rdarer ('xcdfb', 0., 1000., xcdfb, mxnxnofb, helpnxnofb)
      end if

      if (helpnxnofb.ne.nxnofb) then
         write (uoer, 60)
   60 format (//, ' Error in input wlnu.inp,', /,
     &        ' actual length array xcdfb is not equal', /,
     &        ' to declared length, please correct data')
      end if

      call rdarer ('xcddit', 0.1, 10000., xcddit, mxnxnodit,helpnxnodit)
      if (helpnxnodit.ne.nxnodit) then
         write (uoer, 61)
   61 format (//, ' Error in input wlnu.inp,', /,
     &        ' actual length array xcddit is not equal', /,
     &        ' to declared length, please correct data')
      end if

      if (nxnoeb.eq.0) then
C                           dummy statement
         nxnoeb = 0
      else
         call rdarer ('xcdeb', 0., 1000., xcdeb, mxnxnoeb, helpnxnoeb)
      end if

      if (helpnxnoeb.ne.nxnoeb) then
         write (uoer, 62)
   62 format (//, ' Error in input wlnu.inp,' /,
     &        ' actual length array xcdeb is not equal', /,
     &        ' to declared length, please correct data')
      end if

      if (op_input.eq.1) then
         write (uoin, 63)
   63 format (' Section 5: Location space nodes     (m)')
         write (uoin, *) (xcdfb(i), i = 1, nxnofb)
         write (uoin, *) (xcddit(i), i = 1, nxnodit)
         write (uoin, *) (xcdeb(i), i = 1, nxnoeb)
       end if


      close (uiwn)


C---- input data (geometry and characteristics) from file wlpa.inp

      call rdinit (uiwp, uoer, 'wlpa.inp')


C                              geometry ditch
      call rdsrer ('wibot', 0.1, 10., wibot)
      call rdsrer ('sisl', 1.E-05, 10., sisl)
      call rdsrer ('wdhfl', 0.01, 2.0, wdhfl)
      if (op_input.eq.1) then
         write (uoin, 80)
   80 format (/, ' Input from wlpa.inp:', /,
     &        ' Section 1: geometry ditch',
     &        '   (m, dimensionless, resp m)')
         write (uoin, *) wibot, sisl, wdhfl
      end if
      pez0hw = wibot + 2.*wdhfl*sqrt((sisl**2)+1.)


C                              geometry neighbouring plot of land
      call rdsrer ('leplot', 1., 1000., leplot)
      if (op_input.eq.1) then
         write (uoin, 81)
   81 format (' Section 2: Geometry neighbouring plot      (m)')
         write (uoin, *) leplot
       end if


C                              concentration and org. matter susp. solids
      call rdsrer ('coss', 1., 100000., coss)
      call rdsrer ('raomss', 0., 1., raomss)
      if (op_input.eq.1) then
         write (uoin, 90)
   90 format (' Section 3: Concentration and org. matter susp.solids',
     &        '   (g/m^3, dimensionless')
         write (uoin, *) coss, raomss
      end if


C                              amount macrophytes
      call rdsrer ('dwmp', 0., 1000., dwmp)
      if (op_input.eq.1) then
         write (uoin, 100)
  100 format (' Section 4: Amount macrophytes     (g/m^2')
         write (uoin, *) dwmp
      end if


      close (uiwp)


C---- input data (initial concentrations wl) from file wlst.inp

      call rdinit (uiws, uoer, 'wlst.inp')


C                              initial concentrations
      call rdarer ('castwl', 0., 100., castwl, mxnxnotot, helpnxnotot)
```

```
      nxnotot = nxnodit+nxnofb+nxnoeb
      if (helpnxnotot.ne.nxnotot) then
         write (uoer, 110)
  110 format (//, ' Error in input wlst.inp,' /,
     &        ' actual length array castwl is not equal', /,
     &        ' to declared length, please correct data')
      end if

      if (op_input.eq.1) then
         write (uoin, 111)
  111 format (/, ' Input from wlst.inp:', /,
     &        ' Section 1: Initial concentrations',
     &        '    (g/m^3)')
         write (uoin, *) (castwl(i), i = 1, nxnotot)
      end if


C                            constant background concentration in air
      call rdsrer ('coair', 0., 0., coair)
      if (op_input.eq.1) then
         write (uoin, 120)
  120 format (' section 2: Background concentration     (g/m^3)')
         write (uoin, *) coair
      end if


      close (uiws)


      return


      end



      subroutine wbin

** SUBROUTINE:
**    wbin.for - all input data of the sediment layer for the TOXSWA
**               program are read
**    author   - P.I. Adriaanse
**
** DESCRIPTION:
**    Input data for the sediment layer concerning numerical aspects,
**    thickness and other characteristics of the sediment layer and the
**    initial concentrations are read; the input data are converted to
**    SI-units (m, s, g, Pa, K, J, mol).
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                            common variables
      include 'common.for'
C                            local variables
      integer      k, helpnznowb, helpnznoebb,
     &             helplnznowb, help2nznowb, help3nznowb, help4nznowb,
     &             helpnznotot

C---- the following statements have already been executed in the options
C     for output subroutine, opout.for
C---- statement for Vax Fortran 77 compiler for date and time
C
C     call date (dmy)
C     call time (hms)
C                             open file for error messages
C     open (unit = uoer, file = 'message.out', status = 'unknown')
C     write (uoer, 10) dmy, hms
C  10 format (' TOXSWA simulation:', 2A10, /' Error messages:'/)


C---- input data (numerical aspects) from file wbnu.inp

      call rdinit (uibn, uoer, 'wbnu.inp')

      if (op_input.eq.1) then
         write (uoin, 20)
   20 format (/, /, ' All input data from wbin:')
      end if

C                            numerical weight factors
      call rdsrer ('betawb', 0., 1., betawb)
      call rdsrer ('thetawb', 0., 1., thetawb)

      if (op_input.eq.1) then
         write (uoin, 30)
   30 format (/, ' Input from wbnu.inp:', /,
     &        ' Section 1: Numerical weight factors',
     &        '    (no dimensions)')
         write (uoin, *) betawb, thetawb
      end if


C                            time parameters
      call rdsrer ('deltwb', 1., 86400., deltwb)
      if (deltwb.ne.deltwl) then
         write (uoer, 40)
   40 format (//, ' Error in wbnu.inp:', /,
     &        ' deltwb should be equal to deltwl in TOXSWA 1.0 !', /,
     &        ' Program stops')
      end if

      if (deltwb.ne.deltwl) then
           print*, ' Error in wbnu.inp:'
           print*, ' Deltwb should be equal to deltwl in TOXSWA 1.0'
           print*, ' Programs stops'
           stop
      end if

      if (op_input.eq.1) then
         write (uoin, 41)
   41 format (' Section 2: Time parameters        (s)')
         write (uoin, *) deltwb
      end if


C                            space parameters
      call rdsrer ('zwb', 0.01, 0.5, zwb)
      call rdsrer ('zebb', 0., 0.5, zebb)

      if (op_input.eq.1) then
         write (uoin, 50)
   50 format (' Section 3: Space parameters        (m)')
         write (uoin, *) zwb, zebb
      end if


C                            number space nodes
      call rdsinr ('nznowb', 10, 100, nznowb)
      call rdsinr ('nznoebb', 0, 20, nznoebb)

      if (op_input.eq.1) then
         write (uoin, 60)
   60 format (' Section 4: Number space nodes   (no dimensions)')
         write (uoin, *) nznowb, nznoebb
      end if


C                            location space nodes
      call rdarer ('zcdwb', 0.0001, 0.6, zcdwb, mxnznowb, helpnznowb)
      if (helpnznowb.ne.nznowb) then
         write (uoer, 70)
   70 format (//, ' Error in input wbnu.inp,', /,
     &        ' actual length array zcdwb is not equal', /,
     &        ' to declared length, please correct data')
      end if

      if (nznoebb.eq.0) then
C                            dummy statement
         nznoebb = 0
      else
         call rdarer ('zcdebb', 0., 0.6, zcdebb,mxnznoebb,helpnznoebb)
      end if
```

```
      if (helpnznoebb.ne.nznoebb) then
         write (uoer, 71)
   71 format (//, ' Error in input wbnu.inp,' /,
     &        ' actual length array zcdebb is not equal', /,
     &        ' to declared length, please correct data')
      end if


      if (op_input.eq.1) then
         write (uoin, 72)
   72 format (' Section 5: Location space nodes        (m)')
         write (uoin, *) (zcdwb(k), k = 1, nznowb)
         write (uoin, *) (zcdebb(k), k = 1, nznoebb)
      end if


      close (uibn)


C---- input data (sediment characteristics) from file wbpa.inp

      call rdinit (uibp, uoer, 'wbpa.inp')


C                             physical properties sediment
      call rdarer ('bdwb', 10., 3000., bdwb, mxnznowb, help1nznowb)
      call rdarer ('por', 0., 1., por, mxnznowb, help2nznowb)
      call rdarer ('tor', 0., 1., tor, mxnznowb, help3nznowb)
      if (help1nznowb.ne.nznowb .or. help2nznowb.ne.nznowb .or.
     &                                help3nznowb.ne.nznowb) then
         write (uoer, 90)
   90 format (//, ' Error in input wbpa.inp,', /,
     &        ' actual length array bdwb, por or tor is not equal',
     &        /, ' to declared length, please correct data')
      end if


      if (op_input.eq.1) then
         write (uoin, 91)
   91 format (' Section 1: Physical properties sediment',
     &        '    (kg/m^3, (2x) dimensionless)')
         write (uoin, *) (bdwb(k), k = 1, nznowb)
         write (uoin, *) (por(k), k = 1, nznowb)
         write (uoin, *) (tor(k), k = 1, nznowb)
      end if
C                             compose vectors bdwb(k), por(k) and tor(k)
C                             in end buffer
      if (nznoebb.eq.0) then
C                             'dummy' statement
         nznotot = nznowb
      else
         do 92 k = nznowb+1, nznowb+nznoebb
            bdwb(k) = bdwb(nznowb)
            por(k)  = por(nznowb)
            tor(k)  = tor(nznowb)
   92    continue
      end if
      do 93 k = 1, nznowb+nznoebb
         bdwb(k) = bdwb(k)*1000.
   93 continue


C                             organic matter sediment
      call rdarer ('raomwb', 0., 1., raomwb, mxnznowb, help4nznowb)
      if (help4nznowb.ne.nznowb) then
         write (uoer, 100)
  100 format (//, ' Error in input wbpa.inp,' /,
     &        ' actual length array raomwb is not equal', /,
     &        ' to declared length, please correct data')
      end if


      if (op_input.eq.1) then
         write (uoin, 101)
  101 format (' Section 2: Organic matter sediment',
     &        '    (dimensionless)')
         write (uoin, *) (raomwb(k), k = 1, nznowb)
      end if
C                             compose vector raomwb(k) in end buffer
      if (nznoebb.eq.0) then
C                             'dummy' statement
         nznotot = nznowb
      else
         do 102 k = nznowb+1, nznowb+nznoebb
            raomwb(k) = raomwb(nznowb)
  102    continue
      end if


C                             dispersion in sediment
      call rdsrer ('ldis', 0., 1., ldis)

      if (op_input.eq.1) then
         write (uoin, 110)
  110 format (' Section 3: Dispersion length          (m)')
         write (uoin, *) ldis
      end if


      close (uibp)


C---- input data (initial concentrations wb) from file wbst.inp

      call rdinit (uibs, uoer, 'wbst.inp')


C                             initial concentrations
      nznotot = nznowb+nznoebb
      call rdarer ('castwb', 0., 1000., castwb, mxnznotot, helpnznotot)
      if (helpnznotot.ne.nznotot) then
         write (uoer, 120)
  120 format (//, ' Error in input wbst.inp,', /,
     &        ' actual length array castwb is not equal', /,
     &        ' to declared length, please correct data')
      end if


      if (op_input.eq.1) then
         write (uoin, 121)
  121 format (/, ' Input from wbst.inp:', /,
     &        ' Section 1: Initial concentrations',
     &        '    (g/m^3')
         write (uoin, *) (castwb(k), k = 1, nznotot)
      end if


      close (uibs)


      return


      end



      subroutine hyin

** SUBROUTINE:
**    hyin.for - all hydrological input data for the TOXSWA program are
**               read, for the water layer as well as for the sediment
**               layer
**    author   - P.I. Adriaanse
**
** DESCRIPTION:
**    Input data for the water layer concern the constant water flow and
**    water depth; to facilitate possible later linkage to a hydro-
**    dynamic model these constant values are transscribed in variables;
**    input data for the sediment layer concern upward or downward
**    seepage from the neighbouring lot into the ditch or vice versa
**    and the pesticide concentration in the water seeping upwardly into
**    the ditch bottom; all input data are converted to SI-units
**    (m, s, g, Pa, K, J, mol).
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
```

```
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                              common variables
      include 'common.for'
C                              local variables
      integer       i, helpqseif, helpcolot
      real          qvo, wdh


C---- input data from file hy.inp

      call rdinit (uihy, uoer, 'hy.inp')

      if (op_input.eq.1) then
   10 format (//, ' All input from subroutine hyin:')
      end if


C                               flow velocity and water depth water layer
      call rdsrer ('u', -100000., 100000., u)
      call rdsrer ('wdh', 0.1, 2.0, wdh)
      if (op_input.eq.1) then
         write (uoin, 20)
   20 format (/, ' Input from hy.inp:', /,
     &         ' Section 1: Flow velocity and water depth',
     &         '    (m/d, resp m)')
         write (uoin, *) u, wdh
      end if
      u = u/86400.


C                              dispersion coefficient
      call rdsrer ('kds', 10., 100000., kds)
      if (op_input.eq.1) then
         write (uoin, 30)
   30 format (' Section 2: Dispersion coefficient      (m^2/d)')
         write (uoin, *) kds
      end if
      kds = kds/86400.


C                              seepage with concentration
      call rdarer ('qseif', -10.E-03, 10.E-03, qseif, 30, helpqseif)
      call rdarer ('colot', 0., 1., colot, 30, helpcolot)
      if (helpqseif.ne.30 .or. helpcolot.ne.30) then
         write (uoer, 40)
   40 format (//, ' Error in input hy.inp,', /
     &         ' actual length array qseif or colot is not equal', /,
     &         ' to the declared length of 30, please correct data')
      end if

      if (op_input.eq.1) then
         write (uoin, 41)
   41 format (/, ' Section 3: Seepage with concentration',
     &         '    (m^3/m^2.d, resp g/m^3)')
         write (uoin, *) (qseif(i), i = 1, 30)
         write (uoin, *) (colot(i), i = 1, 30)
      end if
      do 42 i = 1, 30
         qseif(i) = qseif(i)/86400.
   42 continue


      close (uihy)


C                              fill array with initial water depth
C                              wdhst(1-nxnotot) and variable water depths
      do 50 i = 1, nxnotot
         wdhst(i) = wdh
   50 continue

      wdhji = wdh
      wdhjim1 = wdh
      wdhjip1 = wdh
      wdhjp1i = wdh
      wdhjp1im1 = wdh
      wdhjp1ip1 = wdh

C                              fill variable discharges
      qvo = (wibot*wdh + (wdh**2)*sisl)*u
      qvojimh = qvo
      qvojiph = qvo
      qvojp1imh = qvo
      qvojp1iph = qvo

C                              output
      if (op_icwlhy.eq.1) then
         write (uowl, 60) qvo
   60 format (/, ' Results from subroutine hyin:', /,
     &         ' discharge equals', 2X, E10.4, ' m3.s-1')
      end if


      return


      end



      subroutine suin

** SUBROUTINE:
**    suin.for - all input data concerning the substance for the TOXSWA
**               program are read
**    author   - P.I. Adriaanse
**
** DESCRIPTION:
**    Transformation rates of the pesticide in water and sediment are
**    read as well as data concerning sorption to sediment, suspended
**    solids and macrophytes; all input data are converted to SI-units
**    (m, s, g, Pa, K, J, mol).
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                              common variables
      include 'common.for'
C                              local variables
      integer       i, k
      real          helpvar, kdomwb1


C---- input data from file su.inp

      call rdinit (uisu, uoer, 'su.inp')

      if (op_input.eq.1) then
         write (uoin, 10)
   10 format (//, ' All input from subroutine suin:')
      end if


C                              transformation of pesticide
      call rdsrer ('krtfwl', 0., 50., krtfwl)
      if (op_input.eq.1) then
         write (uoin, 20)
   20 format (' Section 1: Transformation in water layer',
     &         '    (1/d)')
         write (uoin, *) krtfwl
      end if
      krtfwl = krtfwl/86400.


C                              sorption suspended solids
      call rdsrer ('kdomssdit', 0., 100., kdomssdit)
      call rdsrer ('coobkomss', 1.0E-8, 0.1, coobkomss)
      call rdsrer ('exfrss', 0.1, 2.0, exfrss)
      if (op_input.eq.1) then
         write (uoin, 30)
```

```
   30 format (' section 2: Sorption susp. solids',
     &        '    (m^3/kg, kg/m^3, resp dimensionless)')
         write (uoin, *) kdomssdit, coobkomss, exfrss
      end if
      kdomssdit    = kdomssdit/1000.
      coobkomss = coobkomss*1000.

c --- compose vector kdomss(i), distribution coefficient for sorption to
c     suspended solids, for buffers (no sorption, kdomss=0) and in ditch
c                             in front buffer
      if (nxnofb.eq.0) then
         kdomss(1) = 1.
      else
         do 31 i = 1, nxnofb
            kdomss(i) = zero
   31    continue
      end if

c                             in ditch
      do 32 i = nxnofb+1, nxnofb+nxnodit
         kdomss(i) = kdomssdit
   32 continue

c                             in end buffer
      if (nxnoeb.eq.0) then
c                             'dummy' statement
         nxnotot = nxnofb+nxnodit
      else
         do 33 i = nxnofb+nxnodit+1, nxnofb+nxnodit+nxnoeb
            kdomss(i) = zero
   33    continue
      end if

      do 34 i = 1, nxnofb+nxnodit+nxnoeb
         kdfrss(i) = raomss*kdomss(i)
   34 continue


c                             sorption to macrophytes
      call rdsrer ('kdmpdit', 0., 100., kdmpdit)
      if (op_input.eq.1) then
         write (uoin, 40)
   40 format (' Section 3: Sorption to macrophytes',
     &        '    (m^3/kg)')
         write (uoin, *) kdmpdit
      end if
      kdmpdit = kdmpdit/1000.

c --- compose vector kdmp(i), distribution coefficient for sorption to
c     macrophytes, for buffers (no sorption, kdmp=0) and in ditch
c                             in front buffer
      if (nxnofb.eq.0) then
         kdmp(1) = 1.
      else
         do 41 i = 1, nxnofb
            kdmp(i) = zero
   41    continue
      end if

c                             in ditch
      do 42 i = nxnofb+1, nxnofb+nxnodit
         kdmp(i) = kdmpdit
   42 continue

c                             in end buffer
      if (nxnoeb.eq.0) then
c                             'dummy' statement
         nxnotot = nxnofb+nxnodit
      else
         do 43 i = nxnofb+nxnodit+1, nxnofb+nxnodit+nxnoeb
            kdmp(i) = zero
   43    continue
      end if

c                             volatilisation
      call rdsrer ('klq', 0.1, 10000., klq)
      call rdsrer ('kga', 0.1, 10000., kga)
      call rdsrer ('psat', 0.00001, 0.1, psat)
      call rdsrer ('mamol', 10., 10000., mamol)

      call rdsrer ('tekhe', 200., 350., tekhe)
      call rdsrer ('cosol', 0.001, 100., cosol)

      if (op_input.eq.1) then
         write (uoin, 50)
   50 format (' Section 4: Volatilisation',
     &        '    (m/d, m/d, Pa, g/mol, K, g/m^3)')
         write (uoin, *) klq, kga
         write (uoin, *) psat, mamol, tekhe, cosol
      end if
      klq = klq/86400.
      kga = kga/86400.
      unga = 8.3144                        ! unit J/mol.K
      khe = (psat*mamol)/(unga*tekhe*cosol)
      helpvar = (1./klq)+(1./(khe*kga))
      ktl = 1./helpvar

c                             exchange water layer and sediment
      call rdsrer ('kdfw', 1., 200., kdfw)
      if (op_input.eq.1) then
         write (uoin, 60)
   60 format (' Section 5: Exchange water layer - sed.',
     &        '    (mm^2/d)')
         write (uoin, *) kdfw
      end if
      kdfw = kdfw*(1.0E-06)/86400.


c                             transformation in sediment
      call rdsrer ('krtfwb', 0., 50., krtfwb)
      if (op_input.eq.1) then
         write (uoin, 70)
   70 format (' Section 6: Transformation in sediment ',
     &        '    (1/d)')
         write (uoin, *) krtfwb
      end if
      krtfwb = krtfwb/86400.


c                             dry sediment material
      call rdsrer ('kdomwb1', 0., 100., kdomwb1)
      call rdsrer ('coobkomwb', 1.0E-08, 0.1, coobkomwb)
      call rdsrer ('exfrwb', 0.1, 2.0, exfrwb)
      if (op_input.eq.1) then
         write (uoin, 80)
   80 format (' Section 7: Sorption sediment',
     &        '    (m^3/kg, kg/m^3, resp. dimensionless)')
         write (uoin, *) kdomwb1, coobkomwb, exfrwb
      end if
      kdomwb1    = kdomwb1*0.001
      coobkomwb = coobkomwb*1000.

c                             compose vector kdomwb(k), distribution
c                             coefficient for sorption to sediment
c                             material, for end buffer (no sorption,
c                             kdomwb(k)=0) and in sediment itself

c                             in sediment
      do 81 k = 1, nznowb
         kdomwb(k) = kdomwb1
   81 continue

c                             in end buffer
      if (nznoebb.eq.0) then
c                             'dummy' statement
         nznotot = nznowb
      else
         do 82 k = nznowb+1, nznowb+nznoebb
            kdomwb(k) = zero
   82    continue
      end if

      do 83 k = 1, nznowb+nznoebb
         kdfrwb(k) = raomwb(k)*kdomwb(k)
   83 continue


      close (uisu)
```

```
C --- Additional output

      if (op_icwlhy.eq.1) then
         write (uowl, 90)
   90 format (/, ' Results of some initial calculations from',
     &        ' subroutine suin:')
         write (uowl, 91)
   91 format (' kdfrss(1-nxnotot)', '   (m^3/kg)')
         write (uowl, *) (kdfrss(i), i = 1, nxnotot)
      end if

      if (op_icwb.eq.1) then
         open (unit = uowb, file = 'icwb.out', status = 'unknown')
         write (uowb, 92) dmy, hms
   92 format (' TOXSWA simulation: ', 2A10, /,
     &        ' Results from initial calculations for sediment:', //)
         write (uowb, 93)
   93 format (' Results of some initial calculations from',
     &        ' subroutine suin:')
         write (uowb, 94)
   94 format (' kdfrwb(1-nznotot):', '   (m^3/kg)')
         write (uowb, *) (kdfrwb(k), k = 1, nznotot)
      end if


      return


      end



      subroutine wlit

** SUBROUTINE:
**    wlit.for - initial calculations for the water layer of the TOXSWA
**               program are executed
**    author   - P.I. Adriaanse
**
** DESCRIPTION:
**    The segment lengths in the ditch plus buffers are determined;
**    two subroutines are called:
**   # wlitnu, in which:
**    - the numerical weight factors are calculated for the ditch and
**      for the needed buffers;
**   # wlitco, in which:
**    - the initial concentrations c, Xmp and Xss are calculated at
**      every node;
**    - the exposure concentration at 0 d (total initial concentration
**      c*) is determined.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                            common variables
      include 'common.for'
C                            local variables

      integer    i

C---- first calculation of segment length around every node

C                            in front buffer if present
      if (nxnofb.eq.0) then
         delx(1) = zero
      else
         do 10 i = 1, nxnofb
            if (i.eq.1) then
               delx(i) = (xcdfb(1)+xcdfb(2))/2.
            else
               if (i.eq.nxnofb) then
                  delx(i) = (xcdfb(i)+xcddit(1))/2. -
     &                                          (xcdfb(i-1)+xcdfb(i))/2.
               else
                  delx(i) = (xcdfb(i)+xcdfb(i+1))/2. -
     &                                          (xcdfb( i-1)+xcdfb(i))/2.
               end if
            end if
   10    continue
      end if
C                            in ditch
      do 20 i = nxnofb+1, nxnofb+nxnodit
         if (i.eq.(nxnofb+1) .and. nxnofb.ne.0) then
            delx(i) = (xcddit(2)+xcddit(1))/2. -
     &                      (xcddit(1)+xcdfb(nxnofb))/2.
         else
            if (i.eq.(nxnofb+1) .and. nxnofb.eq.0) then
               delx(i) = (xcddit(2)+xcddit(1))/2.
            else
               if (i.eq.(nxnofb+nxnodit) .and. nxnoeb.eq.0) then
                  delx(i) = xfb+xdit -
     &                   (xcddit(i-nxnofb)+xcddit(i-nxnofb-1))/2.
               else
                  if (i.eq.(nxnofb+nxnodit)) then
                     delx(i) = (xcdeb(1)+xcddit(i-nxnofb))/2.-
     &                         (xcddit(i-nxnofb)+xcddit(i-nxnofb-1))/2.
                  else
                     delx(i) = (xcddit(i-nxnofb+1)+xcddit(i-nxnofb))/2.-
     &                         (xcddit(i-nxnofb)+xcddit(i-nxnofb-1))/2.
                  end if
               end if
            end if
         end if
   20 continue

C                            in end buffer
      if (nxnoeb.eq.0) then
C                            'dummy' statement
         nxnotot= nxnofb+nxnodit
      else
         do 30 i = nxnofb+nxnodit+1, nxnofb+nxnodit+nxnoeb
            if (i.eq.(nxnofb+nxnodit+1)) then
               delx(i) = (xcdeb(2)+xcdeb(1))/2. -
     &                       (xcdeb(1)+xcddit(nxnodit))/2.
            else
               if (i.eq.(nxnofb+nxnodit+nxnoeb)) then
                  delx(i) = xfb+xdit+xeb - (xcdeb(i-nxnofb-nxnodit)
     &                            +xcdeb(i-nxnofb-nxnodit-1))/2.
               else
                   delx(i) = (xcdeb(i-nxnofb-nxnodit+1)+
     &                       xcdeb(i-nxnofb-nxnodit))/2. -
     &                       (xcdeb(i-nxnofb-nxnodit)+
     &                       xcdeb(i-nxnofb-nxnodit-1))/2.
               end if
            end if
   30    continue
         end if

C---- Compose vector xcd(nxnotot) with x coordinates in ditch included
C     the buffers

C                            in front buffer
      if (nxnofb.eq.0) then
         xcd(1) = 1.
      else
         do 40 i = 1, nxnofb
            xcd(i) = xcdfb(i)
   40    continue
      end if

C                            in ditch
      do 50 i = nxnofb+1, nxnofb+nxnodit
         xcd(i) = xcddit(i-nxnofb)
   50 continue

C                            in end buffer
      if (nxnoeb.eq.0) then
C                            'dummy' statement
         nxnotot = nxnofb+nxnodit
      else
         do 60 i = nxnofb+nxnodit+1, nxnofb+nxnodit+nxnoeb
            xcd(i) = xcdeb(i-nxnofb-nxnodit)
   60    continue
```

```
      end if

      if (op_icwlhy.eq.1) then
         write (uowl, 70)
   70 format ( /,
     &        ' Results of initial calculations from subroutine wlit:')
         write (uowl, 71)
   71 format (' delx(1-nxnotot):', '   (m)')
         write (uowl, *) (delx(i), i = 1, nxnotot)
         write (uowl, 72)
   72 format (' xcd(1-nxnotot):', '   (m)')
         write (uowl, *) (xcd(i), i = 1, nxnotot)
      end if

C                                 numerical weight factors
      call wlitnu
C                                 initial concentrations
      call wlitco (nxnotot, costwl, sompst, sossst)

      return

      end



      subroutine wlitnu

** SUBROUTINE:
**    wlitnu.for - initial calculations concerning numerical aspects
**                 for the water layer of the TOXSWA program are
**                 executed
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The numerical weight factors are calculated for the water layer
**    and the needed buffers.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                              common variables
      include 'common.for'
C                              local variables
      integer    i


C---- numerical weight factors at the segment boundaries

      do 230 i = 2, nxnotot-1
         if (betawl.ge.zero .and. betawl.le.0.5) then
            nwwmh(i) = (betawl*delx(i-1))/(xcd(i)-xcd(i-1))
            nwwph(i) = (betawl*delx(i))/(xcd(i+1)-xcd(i))
         else
            if (betawl.gt.0.5 .and. betawl.le.1.0) then
               nwwmh(i) = (0.5*delx(i-1) + (betawl-0.5)*delx(i)) /
     &                    (xcd(i)-xcd(i-1))
               nwwph(i) = (0.5*delx(i) + (betawl-0.5)*delx(i+1)) /
     &                    (xcd(i+1)-xcd(i))
            end if
         end if
  230 continue

      nwwph(1) = nwwmh(2)
      nwwmh(nxnotot) = nwwph(nxnotot-1)


      if (op_icwlhy.eq.1) then
         write (uowl, 235)
  235 format (/, ' Results of calculations from subroutine wlitnu:')
         write (uowl, 285)
  285 format (' num. weight factors for all nodes:', /,
     &     '  nodenumber    nwwmh          nwwph        (dimensionless)')
         do 295 i = 1, nxnotot
            write (uowl, *) i, nwwmh(i), nwwph(i)
  295    continue
      end if

      return

      end



      subroutine wlitco (ntot, co, somp, soss)

** SUBROUTINE:
**    wlitco.for - initial concentrations for the water layer of the
**                 TOXSWA program are calculated
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The initial concentrations in the water phase, c, sorbed at the
**    macrophytes, Xmp, and sorbed at the suspended solids, Xss, are
**    calculated at every node. The exposure concentration at 0 d (total
**    initial concentration, c*) is also determined.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                              common variables
      include 'common.for'
C                              local variables
      integer    i, ii, ntot
      real       co(ntot), somp(ntot), soss(ntot), warst,
     &           col(mxnxnotot), co2(mxnxnotot)

C                              node loop
      do 30 i = 1, ntot

         warst = wibot*wdhst(i)+(wdhst(i)**2)*sisl

         if (castwl(i).le.0.) then
            co(i)   = zero
            somp(i) = zero
            soss(i) = zero
         else
            col(i) = castwl(i)
            co2(i) = castwl(i)/(1.+ (dwmp*pez0hw*kdmp(i))/warst +
     &               coss*raomss*kdomss(i)*(col(i)/coobkomss)**
     &                                                 (exfrss-1.))
C                              iteration loop
   10       if (abs(co2(i)-col(i)).lt.abs(0.001*co2(i))) go to 20
               col(i) = co2(i)
               co2(i) = castwl(i)/(1.+ (dwmp*pez0hw*kdmp(i))/warst +
     &                  coss*raomss*kdomss(i)*
     &                  (col(i)/coobkomss)**(exfrss-1.))
               go to 10
   20       continue
            co(i)   = co2(i)
            somp(i) = kdmp(i)*co(i)
            soss(i) = raomss*kdomss(i)*coobkomss*
     &                ((co(i)/coobkomss)**exfrss)
         end if
   30 continue

C---- Exposure concentration at 0 d
      do 35 i = 1, ntot
         do 33 ii = 1, nwbsy
            if (i.eq.iwbsy(ii)) then
               coxp(ii,1) = castwl(i)
            end if
   33    continue
   35 continue

      if (op_icwlhy.eq.1) then
```

```
      write (uowl, 40)
   40 format (/, ' Results of calculations from subroutine wlitco:', /,
     &         ' length exchanging perimeter, pez0hw:', '    (m)')
        write (uowl, *) pez0hw
        write (uowl, 41)
   41 format (' initial wetted area ditch, warst:', '    (m2)')
        write (uowl, *) warst
        write (uowl, 42)
   42 format (' initial water depths for all nodes, wdhst(1-nxnotot):',
     &        '     (m)')
        write (uowl, *) (wdhst(i), i = 1, ntot)
        write (uowl, 43)
   43 format (' initial concentrations for all nodes', /,
     & ' nodenr, total, dissolved, at macrophytes and at susp solids',/,
     &    '      ixnotot     castwl(i)       co(i)          somp(i)  and',
     &    '    soss(i)'
     & /, '                   (g.(m)-3        g.(m)-3        g.(g)-1      ',
     &    '    g.(g)-1)')
        do 44 i = 1, ntot
           write (uowl, *) i, castwl(i), co(i), somp(i), soss(i)
   44   continue
      end if

      return


      end



      subroutine wbit

** SUBROUTINE:
**    wbit.for - initial calculations for the sediment are executed
**    author   - P.I. Adriaanse
**
** DESCRIPTION:
**    The segment lengths in the sediment plus buffer are determined.
**    Three subroutines are called:
**    # wbitnu, in which:
**     - the numerical weight factors are calculated for the sediment
**       and the buffer;
**    # wbitge, in which:
**     -the perimeter at a distance d of the water-sediment interface
**      is calculated (Pk-1/2, Pk en Pk+1/2);
**     -the porosity (epsilon) is calculated for k-1/2 and k+1/2;
**     -the bulk density is calculated for k-1/2 and k+1/2;
**     -the Freundlich sorption coefficient, KF, is calculated at k-1/2
**      and k+1/2;
**     -the diffusion coefficients for the pesticide in the liquid phase
**      of the sediment, Dlb,k-1/2 and Dlb,k+1/2, are calculated;
**    # wbitco, in which:
**     - the initial concentrations in the liquid phase, clb, and sorbed
**       to solid sediment material, Xb, are calculated at every node.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                              common variables
      include 'common.for'
C                              local variables

      integer     k


C---- first calculation of segment length around every node

C                              in sediment
      do 10 k = 1, nznowb
         if (k.eq.1) then
            delz(k) = (zcdwb(1)+zcdwb(2))/2.
         else
            if (k.eq.nznowb) then
               delz(k) = zwb - (zcdwb(k-1)+zcdwb(k))/2.
            else
               delz(k) = (zcdwb(k)+zcdwb(k+1))/2. -
     &                           (zcdwb(k-1)+zcdwb(k))/2.
            end if
         end if
   10 continue

C                              in end buffer
      if (nznoebb.eq.0) then
C                              'dummy' statement
         nznotot = nznowb+nznoebb
      else
         do 20 k = nznowb+1, nznowb+nznoebb
            if (k.eq.nznowb+1) then
               delz(k) = (zcdebb(2)+zcdebb(1))/2. -
     &                  (zcdwb(nznowb)+zcdebb(1))/2.
            else
               if (k.eq.nznowb+nznoebb) then
                  delz(k) = zwb+zebb -
     &                  (zcdebb(k-nznowb)+zcdebb(k-nznowb-1))/2.
               else
                  delz(k) = (zcdebb(k-nznowb+1)+zcdebb(k-nznowb))/2. -
     &                  (zcdebb(k-nznowb)+zcdebb(k-nznowb-1))/2.
               end if
            end if
   20    continue
      end if


C---- Compose vector zcd(nznotot) with z coordinates in sediment
C     included the buffer

C                              in sediment
      do 30 k = 1, nznowb
         zcd(k) = zcdwb(k)
   30 continue

C                              in end buffer
      if (nznoebb.eq.0) then
C                              'dummy' statement
         nznotot = nznowb+nznoebb
      else
         do 40 k = nznowb+1, nznowb+nznoebb
            zcd(k) = zcdebb(k-nznowb)
   40    continue
      end if


      if (op_icwb.eq.1) then
         write (uowb, 50)
   50 format (/, ' Results of initial calculations',
     &         ' from subroutine wbit:')
         write (uowb, 51)
   51 format (' delz(1-nznotot) from wbit:', '    (m)')
         write (uowb, *) (delz(k), k = 1, nznotot)
         write (uowb, 52)
   52 format (' zcd(1-nznotot) from wbit:', '    (m)')
         write (uowb, *) (zcd(k), k = 1, nznotot)
      end if


C                              - numerical weight factors
      call wbitnu
C                              - perimeters
C                              - porosities
C                              - diffusion coefficients
      call wbitge
C                              - initial concentrations
      call wbitco (nznotot, costwb, sowbst)

      return


      end



      subroutine wbitnu

** SUBROUTINE:
```

```
**    wbitnu.for - initial calculations concerning numerical aspects for
**                 the sediment of the TOXSWA program are executed
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The numerical weight factors are calculated for the sediment
**    and the end buffer.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

c                             common variables
      include 'common.for'
c                             local variables
      integer    k


C---- numerical weight factors at the segment boundaries

      do 10 k = 2, nznotot-1
         if (betawb.ge.zero .and. betawb.le.0.5) then
            nwbmh(k) = (betawb*delz(k-1))/(zcd(k)-zcd(k-1))
            nwbph(k) = (betawb*delz(k))/(zcd(k+1)-zcd(k))
         else
            if (betawb.gt.0.5 .and. betawb.le.1.0) then
               nwbmh(k) = (0.5*delz(k-1) + (betawb-0.5)*delz(k)) /
     &                    (zcd(k)-zcd(k-1))
               nwbph(k) = (0.5*delz(k) + (betawb-0.5)*delz(k+1)) /
     &                    (zcd(k+1)-zcd(k))
            end if
         end if
   10 continue

      nwbph(1) = nwbmh(2)
      nwbmh(nznotot) = nwbph(nznotot-1)


      if (op_icwb.eq.1) then
         write (uowb, 20)
   20 format (/,
     &        ' Results of initial calculations from subroutine wbitnu:')
         write (uowb, 21)
   21 format (' num. weight factors for all nodes:', /,
     &        ' nodenumber    nwbmh          nwbph      (dimensionless)')
         do 22 k = 1, nznotot
            write (uowb, *) k, nwbmh(k), nwbph(k)
   22    continue
      end if

      return


      end



      subroutine wbitge

** SUBROUTINE:
**    wbitge.for - initial calculations concerning sediment and sorption
**                 characteristics for the TOXSWA program are executed
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The following initial calculations are executed here:
**    -the perimeter at a distance d of the water-sediment interface
**     is calculated (Pk-1/2, Pk en Pk+1/2);
**    -the porosity (epsilon) is calculated for k-1/2 and k+1/2;
**    -the bulk density is calculated for k-1/2 and k+1/2;
**    -the Freundlich sorption coefficient, KF, is calculated for k-1/2
**     and k+1/2;
**    -the diffusion coefficients for the pesticide in the liquid phase
**     of the sediment, Dlb,k-1/2 and Dlb,k+1/2, are calculated.
```

```
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

c                             common variables
      include 'common.for'
c                             local variables
      integer    k
      real       betasisl, sumdelz(mxnznotot), tormh, torph,
     &           raomwbmh, raomwbph

C---- Calculate perimeter at depth d from the water-sediment interface,
c     Pk-1/2, Pk and Pk+1/2


c     P at z=0, pez0hw, has already been calculated in wlin.for

      betasisl = atan(1./sisl)

      do 10 k = 1, nznotot
         pe(k) = wibot + 2.*zcd(k)*tan(0.5*betasisl) +
     &                   2.*(wdhfl+zcd(k))*sqrt((sisl**2)+1.)
   10 continue

      do 15 k = 1, nznotot
         sumdelz(k) = zcd(k) + 0.5*delz(k)
   15 continue

      do 20 k = 1, nznotot
         if (k.eq.1) then
            pemh(k) = pez0hw
         else
            pemh(k) = wibot + 2.*sumdelz(k-1)*tan(0.5*betasisl) +
     &                2.*(wdhfl+sumdelz(k-1))*sqrt((sisl**2)+1.)
         end if
   20 continue

      do 25 k = 1, nznotot
            peph(k) = wibot + 2.*sumdelz(k)*tan(0.5*betasisl) +
     &                2.*(wdhfl+sumdelz(k))*sqrt((sisl**2)+1.)
   25 continue

C---- Calculate porosity, epsilon(k+1/2) and epsilon(k-1/2), in
c     the sediment included the buffer

c                             in sediment included the buffer
      do 30 k = 1, nznotot
         if (k.eq.1) then
            pormh(k) = por(1)
            porph(k) = (delz(k+1)/(delz(k)+delz(k+1)))*por(k) +
     &                 (delz(k)/(delz(k)+delz(k+1)))*por(k+1)
         else
            if (k.eq.nznotot) then
               pormh(k) = (delz(k)/(delz(k-1)+delz(k)))*por(k-1) +
     &                    (delz(k-1)/(delz(k-1)+delz(k)))*por(k)
               porph(k) = por(nznotot)
            else
               pormh(k) = (delz(k)/(delz(k-1)+delz(k)))*por(k-1) +
     &                    (delz(k-1)/(delz(k-1)+delz(k)))*por(k)
               porph(k) = (delz(k+1)/(delz(k)+delz(k+1)))*por(k) +
     &                    (delz(k)/(delz(k)+delz(k+1)))*por(k+1)
            end if
         end if
   30 continue

C---- Calculate bulk density, bdwb(k+1/2) and bdwb(k-1/2), in
c     the sediment included the buffer

c                             in sediment included the buffer
      do 40 k = 1, nznotot
         if (k.eq.1) then
            bdwbmh(k) = bdwb(1)
            bdwbph(k) = (delz(k+1)/(delz(k)+delz(k+1)))*bdwb(k) +
     &                  (delz(k)/(delz(k)+delz(k+1)))*bdwb(k+1)
```

```fortran
         else
            if (k.eq.nznotot) then
               bdwbmh(k) = (delz(k)/(delz(k-1)+delz(k)))*bdwb(k-1) +
     &                     (delz(k-1)/(delz(k-1)+delz(k)))*bdwb(k)
               bdwbph(k) = bdwb(nznotot)
            else
               bdwbmh(k) = (delz(k)/(delz(k-1)+delz(k)))*bdwb(k-1) +
     &                     (delz(k-1)/(delz(k-1)+delz(k)))*bdwb(k)
               bdwbph(k) = (delz(k+1)/(delz(k)+delz(k+1)))*bdwb(k) +
     &                     (delz(k)/(delz(k)+delz(k+1)))*bdwb(k+1)
            end if
         end if
   40 continue

C---- Calculate Freundlich sorption coefficient, kdfrwb(k+1/2) and
C     kdfrwb(k-1/2), in the sediment included the buffer

C                             in sediment included the buffer
      do 50 k = 1, nznotot
         if (k.eq.1) then
            raomwbmh = raomwb(1)
            raomwbph = (delz(k+1)/(delz(k)+delz(k+1)))*raomwb(k) +
     &                 (delz(k)/(delz(k)+delz(k+1)))*raomwb(k+1)
         else
            if (k.eq.nznotot) then
               raomwbmh = (delz(k)/(delz(k-1)+delz(k)))*raomwb(k-1) +
     &                    (delz(k-1)/(delz(k-1)+delz(k)))*raomwb(k)
               raomwbph = raomwb(nznotot)
            else
               raomwbmh = (delz(k)/(delz(k-1)+delz(k)))*raomwb(k-1) +
     &                    (delz(k-1)/(delz(k-1)+delz(k)))*raomwb(k)
               raomwbph = (delz(k+1)/(delz(k)+delz(k+1)))*raomwb(k) +
     &                    (delz(k)/(delz(k)+delz(k+1)))*raomwb(k+1)
            end if
         end if
         kdfrwbmh(k) = raomwbmh * kdomwb(k)
         kdfrwbph(k) = raomwbph * kdomwb(k)
   50 continue

C                             Avoid that two values exist for kdfrwb at
C                             interface between sediment and endbuffer
         kdfrwbph(nznowb)   = (kdfrwb(nznowb)+kdfrwb(nznowb+1))/2.
         kdfrwbmh(nznowb+1) = (kdfrwb(nznowb)+kdfrwb(nznowb+1))/2.


C---- Calculate diffusion coefficient Dlb,k+1/2 and Dlb,k-1/2 in
C     the sediment included the buffer

C                             in sediment included the buffer
      do 70 k = 1, nznotot
         if (k.eq.1) then
            tormh = tor(1)
            torph = (delz(k+1)/(delz(k)+delz(k+1)))*tor(k) +
     &              (delz(k)/(delz(k)+delz(k+1)))*tor(k+1)
         else
            if (k.eq.nznotot) then
               tormh = (delz(k)/(delz(k-1)+delz(k)))*tor(k-1) +
     &                 (delz(k-1)/(delz(k-1)+delz(k)))*tor(k)
               torph = tor(nznotot)
            else
               tormh = (delz(k)/(delz(k-1)+delz(k)))*tor(k-1) +
     &                 (delz(k-1)/(delz(k-1)+delz(k)))*tor(k)
               torph = (delz(k+1)/(delz(k)+delz(k+1)))*tor(k) +
     &                 (delz(k)/(delz(k)+delz(k+1)))*tor(k+1)
            end if
         end if
         kdfwbmh(k) = tormh*kdfw
         kdfwbph(k) = torph*kdfw
   70 continue

      if (op_icwb.eq.1) then
         write (uowb, 80)
   80 format (/, ' Results of initial calculations from subroutine',
     &        ' wbitge:')
         write (uowb, 81)
   81 format (' length perimeter at nodes and interfaces for',
     &        ' all nodes:', /, ' nodenumber      pemh            pe',
     &        '          peph         (m)')
         do 82 k = 1, nznotot
            write (uowb, *) k, pemh(k), pe(k), peph(k)
   82    continue

         write (uowb, 83)
   83 format(' porosity at interfaces for all nodes:', /,
     &       ' nodenumber    pormh           porph',
     &       '         (dimensionless)')
         do 84 k = 1, nznotot
            write (uowb, *) k, pormh(k), porph(k)
   84    continue

         write (uowb, 85)
   85 format(' bulkdensity at interfaces for all nodes:', /,
     &       ' nodenumber   bdwbmh          bdwbph       (g.(m)-3)')
         do 86 k = 1, nznotot
            write (uowb, *) k, bdwbmh(k), bdwbph(k)
   86    continue

         write (uowb, 87)
   87 format(' Freundlich sorption coefficient at interfaces for',
     &       ' all nodes:', /,
     &       ' nodenumber     kdfrwbmh        kdfrwbph     (m3.(g)-1)')
         do 88 k = 1, nznotot
            write (uowb, *) k, kdfrwbmh(k), kdfrwbph(k)
   88    continue

         write (uowb, 89)
   89 format(' diffusion coefficients at interfaces for all nodes:',
     &       /, ' nodenumber    kdfwbmh          kdfwbph       (m3.(g)-1)')
         do 90 k = 1, nznotot
            write (uowb, *) k, kdfwbmh(k), kdfwbph(k)
   90    continue
      end if


      return


      end



      subroutine wbitco (ztot, co, sowb)

** SUBROUTINE:
**    wbitco.for - initial concentrations for the sediment layer for
**                 the TOXSWA program are executed
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The initial concentrations in the liquid phase, clb, and sorbed to
**    solid sediment material, Xb, are calculated at every node.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                            common variables
      include 'common.for'
C                            local variables
      integer    k, ztot
      real       co(ztot), sowb(ztot),
     &           col(mxnznotot), co2(mxnznotot)

C                            node loop
      do 30 k = 1, ztot

         if (castwb(k).le.0.) then
            co(k)   = zero
            sowb(k) = zero
         else
            col(k) = castwb(k)
            co2(k) = castwb(k)/(por(k) + bdwb(k)*kdfrwb(k)*
     &               (col(k)/coobkomwb)**(exfrwb-1.))
C                            iteration loop
   10       if (abs(co2(k)-col(k)).lt.abs(0.001*co2(k))) go to 20
```

```
            col(k) = co2(k)
            co2(k) = castwb(k)/(por(k) + bdwb(k)*kdfrwb(k)*
     &                  (col(k)/coobkomwb)**(exfrwb-1.))
            go to 10
20       continue
         co(k)    = co2(k)
         sowb(k) = kdfrwb(k)*coobkomwb*((co(k)/coobkomwb)**exfrwb)
      end if
30 continue


   if (op_icwb.eq.1) then
      write (uowb, 40)
40 format (/, ' Results of initial calculations from subroutine',
     &       ' wbitco:')
      write (uowb, 41)
41 format (' initial concentrations for all nodes', /,
     & '      nodenr,  total,          dissolved and  sorbed at solid',
     &        ' bottom mat.', /,
     & '     kznotot      castwb(k)       co(k)           sowb(k) ', /,
     & '                  (g.(m)-3        g.(m)-3         g.(g)-1)')
      do 42 k = 1, ztot
         write (uowb, *) k, castwb(k), co(k), sowb(k)
42    continue
   end if

   return


   end


   subroutine wlso

** SUBROUTINE:
**    wlso.for - solves the mass conservation equation for the water
**               layer of the TOXSWA program
**    author   - P.I. Adriaanse
**
** DESCRIPTION:
**    The mass conservation equation for the sediment subsystem is
**    solved by solving the matrix equation, containing two tridiagonal
**    matrices describing the pesticide concentration in the liquid
**    phase of the sediment (Eq. (6.39) in SC-DLO report 90). The
**    equation can be solved in an implicit or in an explicit way and
**    due to the sorption description according to the Freundlich
**    equation an iteration takes place to calculate the final
**    pesticide concentration.
**    Exposure concentrations at 0, 3, 21 and 28 day are also
**    calculated in this subroutine.
**    The following subroutines are called:
**   # wlsoin, in which:
**    - concentration and seepage rates are read for timestep j;
**   # wlsoge, in which:
**    - some geometrical characteristics for the water layer are
**      calculated;
**   # wlsods, in which:
**    - the (physical) dispersion coefficient is corrected for the
**      numerical dispersion;
**   # wlsoem, in which:
**    - the elements of the tridiagonal matrices are defined, which are
**      used in the subroutines wlsoub, wlsolb and wlsocr;
**   # wlsolb, in which:
**    - the lod, ldd etc terms of the matrices for the lower boundary of
**      the water layer are composed;
**   # wlsoub, in which:
**    - the lod, ldd etc terms of the matrices for the upper boundary of
**      the water layer are composed;
**   # wlsocr, in which:
**    - the lod, ldd etc terms of the core parts of the matrices for the
**      water layer are composed;
**   # tridag, in which:
**    - the tridiagonal matrix for the water layer is inverted.
**
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

c                             common variables
      include 'common.for'
c                             local variables
      integer    i, ii, ittlo, poscounter
      real       rh(mxnxnotot)
      logical    loghelp, convergency_OK


      do 10 ixnotot = 1, nxnotot
         if (its.eq.1) then
            cowlj(ixnotot) = costwl(ixnotot)
         else
            cowlj(ixnotot) = cowljp1(ixnotot)
         end if
   10 continue


      do 20 ixnotot = 1, nxnotot
c                             read concentration c_lb of first
c                             node sediment for each node water layer;
c                             calculate seepage/infiltration plus
c                             concentration for j and j+1
         call wlsoin
c                             calculate wetted areas, water depths, width
c                             water surface and discharges at nodes and
c                             segment boundaries at time j, j+1/2 and j+1
         call wlsoge
   20 continue


C---- Calculate numerical and calculation dispersion

      do 30 ixnotot = 1, nxnotot
         call wlsods
   30 continue

      if (its.eq.1) then
         poscounter = 1
      end if
c                              poscounter counts the number of times that
c                              the positivity conditions are not
c                              fullfilled

      ittlo = 1

C---  Loop while no final value of cowljp1. (So start of iteration
c     procedure due to sorption according to Freundlich equation to
c     find c at time j+1)
   40 continue


C---- Make estimation for c at time j+1 for calculation of lod, ldd and
c     lbd

      do 50 ixnotot = 1, nxnotot
         if (ittlo.eq.1) then
            oldcowljp1(ixnotot) = cowlj(ixnotot)
         else
            oldcowljp1(ixnotot) = cowljp1(ixnotot)
         end if

C---- Compose left-hand and right-hand matrices

c                             prevent undefined exponentiation in lod,
c                             ldd, lbd and rod, rdd and rbd terms
         if (cowlj(ixnotot).le.zero) then
            cowlj(ixnotot) = tiny
         end if
         if (oldcowljp1(ixnotot).le.zero) then
            oldcowljp1(ixnotot) = tiny
         end if

   50 continue
```

```fortran
      do 60 ixnotot = 1, nxnotot
C                             definition separate terms of elements of
C                             matrix
         call wlsoem

         if (ixnotot.ge.nxnofb+nxnodit) then
C                             compose elements matrix in end buffer wl
            call wlsolb
         else
            if (ixnotot.le.nxnofb+1) then
C                             compose elements matrix in front buffer wl
               call wlsoub
            else
C                             compose core matrix (wl excluded buffers)
               call wlsocr
            end if
         end if

C---- Output positivity conditions for first timestep

         if (op_icwlhy.eq.1) then
            if (its.eq.1 .and. ixnotot.eq.1 .and. ittlo.eq.1) then
               write (uowl, 51)
   51 format (//, ' Results from subroutine wlso')
               write (uowl, 52)
   52 format (/, ' positivity conditions are: lbd =< 0    rbd >= 0',
     &        /, '                            ldd > 0     rdd > 0',
     &        /, '                            lod =< 0    rod >= 0')
            end if
            if (its.eq.1 .and. ittlo.eq.1) then
               write (uowl, 53)
   53 format (/, ' lbd    rbd    timestep   nodenumber', /,
     &           ' ldd    rdd', /,
     &           ' lod    rod', /,
     &           '        rv')
               write (uowl, *) lbd(ixnotot), rbd(ixnotot), its, ixnotot
               write (uowl, *) ldd(ixnotot), rdd(ixnotot)
               write (uowl, *) lod(ixnotot), rod(ixnotot)
               write (uowl, *) rv(ixnotot)
            end if
         end if

C---- Test positivity conditions

         loghelp = (lbd(ixnotot).gt.zero .or. ldd(ixnotot).lt.zero .or.
     &              lod(ixnotot).gt.zero .or. rbd(ixnotot).lt.zero .or.
     &              rdd(ixnotot).lt.zero .or. rod(ixnotot).lt.zero)
         if (loghelp) then

            poscounter = poscounter+1
            if (poscounter.eq.50) then
               print*, ' Error: '
               print*, ' Positivity conditions wl not fulfilled;'
               print*, ' see message.out'
               print*, ' Program stops'
               stop
            end if

            write (uoer, 54)
   54 format (//, ' Message from subroutine wlso')
            write (uoer, 55)
   55 format (/, ' positivity conditions are: lbd =< 0    rbd >= 0',
     &        /, '                            ldd > 0     rdd > 0',
     &        /, '                            lod =< 0    rod >= 0')
            write (uoer, 56)
   56 format (/, ' lbd    rbd    timestep   nodenumber', /,
     &           ' ldd    rdd', /,
     &           ' lod    rod', /,
     &           '        rv')
            write (uoer, *) lbd(ixnotot), rbd(ixnotot), its, ixnotot
            write (uoer, *) ldd(ixnotot), rdd(ixnotot)
            write (uoer, *) lod(ixnotot), rod(ixnotot)
            write (uoer, *) rv(ixnotot)
            write (uoer, 57)
   57 format (/, ' So positivity conditions have not been met.',
     &        /, ' Are solutions given and are these OK ?')
         end if

   60 continue

C---- Split solution of matrix equations in solution with explicit
C     calculation scheme and with implicit calculation scheme

C                             explicit calculation scheme
      if (abs(thetawl-1.).lt.1.0E-06) then
C                             thetawl = 1;
C                             no inversion of left-hand matrix needed
C                             to solve cj+1,i;
C                             terms lod and lbd are zero in this case
         do 70 ixnotot = 1, nxnotot

            if (ixnotot.eq.1) then
               cowljp1(ixnotot) = (1/ldd(ixnotot))*(
     &                            rdd(ixnotot)*cowlj(ixnotot)+
     &                            rbd(ixnotot)*cowlj(ixnotot+1)+
     &                            rv(ixnotot))
            else
               if (ixnotot.eq.nxnotot) then
                  cowljp1(ixnotot) = (1/ldd(ixnotot))*(
     &                               rod(ixnotot)*cowlj(ixnotot-1)+
     &                               rdd(ixnotot)*cowlj(ixnotot)+
     &                               rv(ixnotot))
               else
                  cowljp1(ixnotot) = (1/ldd(ixnotot))*(
     &                               rod(ixnotot)*cowlj(ixnotot-1)+
     &                               rdd(ixnotot)*cowlj(ixnotot)+
     &                               rbd(ixnotot)*cowlj(ixnotot+1)+
     &                               rv(ixnotot))
               end if
            end if

   70    continue

C                             implicit calculation scheme
      else
         if (thetawl.ge.zero .and. abs(thetawl-1.).gt.1.0E-06) then
C                             thetawl is smaller than 1 and equal or
C                             bigger than 0;
C                             inversion of left-hand tridiagonal matrix
C                             needed
            do 80 ixnotot = 1, nxnotot

C                             compose right-hand of matrix equation
               if (ixnotot.eq.1) then
                  rh(ixnotot) = rdd(1)*cowlj(1)+rbd(1)*cowlj(2)+rv(1)
               else
                  if (ixnotot.eq.nxnotot) then
                     rh(ixnotot) = rod(nxnotot)*cowlj(ixnotot-1)+
     &                             rdd(nxnotot)*cowlj(ixnotot)+
     &                             rv(nxnotot)
                  else
                     rh(ixnotot) = rod(ixnotot)*cowlj(ixnotot-1)+
     &                             rdd(ixnotot)*cowlj(ixnotot)+
     &                             rbd(ixnotot)*cowlj(ixnotot+1)+
     &                             rv(ixnotot)
                  end if
               end if

   80       continue

            call tridag (lod, ldd, lbd, rh, cowljp1, nxnotot)

         end if
      end if

C---- First iteration due to sorption according to Freundlich equation
C     has been done now, so first value of vector cowljp1 (nxnotot) has
C     been found; test whether more iterations are needed

      ixnotot = 1
      convergency_OK = .true.
   90 if (convergency_OK .and. (ixnotot.le.nxnotot)) then
         convergency_OK = (abs(cowljp1(ixnotot)-oldcowljp1(ixnotot)).lt.
     &                    abs(0.00001*cowljp1(ixnotot)))
         ixnotot = ixnotot+1
      go to 90
      end if
```

```
      if (.not. convergency_OK) then
C                             give error message
         if (ittlo.eq.50) then
            write (uoer, 91)
   91 format (//, ' Message from subroutine wlso:')
            write (uoer, 92) ittlo, ixnotot, its
   92 format (/, ' More than', /,
     &        I6, ' iteration loops will be needed to calculate cj+1',
     &       /, ' for gridpoint:', I6, ' at timestep:', I8,
     &       /, ' in the water layer')
                  write (uoer, 93) oldcowljpl(ixnotot), cowljpl(ixnotot)
   93 format (/, ' Iteration values oldcowljpl and cowljpl are:', E10.4,
     &        2X, E10.4, /, ' Never ending loop !')
            print*,    ' Error: '
            print*,    ' Never ending loop in wlso; '
            print*,    ' see message.out '
            print*,    ' Program stops '

            stop

         end if
      end if

      if (.not. convergency_OK) then
         ittlo = ittlo +1
         go to 40
      end if
C---- End of loop while no final value of cowljpl


C---- Now final value of c at time j+1, cowljpl found

C---- Calculate Xss, Xmp and c* at time j+1 at each node

      do 100 ixnotot = 1, nxnotot
         if (cowljpl(ixnotot).lt.zero) then
            write (uoer, 96)
   96 format (//, ' Message from subroutine wlso:')
            write (uoer, 97) its, ixnotot, cowljpl(ixnotot)
   97 format (/, ' Negative concentration in ditch !', /,
     &          ' at timestep:', I6, ' and node:', I3, /,
     &        ' Concentration is:', E10.4)
            cowljpl(ixnotot) = tiny
         end if

C---- When the concentration at an individual node becomes smaller than
C     1E-25 the concentration will be set at 1E-25. (At concentrations
C     in the order of 1E-34 no convergency will be reached in the
C     iteration loop at statement label 90 in this subroutine wlso.for.
C     Also other numerical problems due to computer range limitations
C     will be prevented in this way.)
         if (cowljpl(ixnotot).lt.1.0E-25) then
            cowljpl(ixnotot) = 1.0E-25
         end if


         sossjpl(ixnotot) = raomss*kdomss(ixnotot)*coobkomss*
     &                         ((cowljpl(ixnotot)/coobkomss)**exfrss)
         sompjpl(ixnotot) = kdmp(ixnotot)*cowljpl(ixnotot)
         cawljpl(ixnotot) = cowljpl(ixnotot) +
     &                      dwmp*pez0hw*sompjpl(ixnotot)/warjpl(ixnotot)
     &                         + coss*sossjpl(ixnotot)
  100 continue


C---- Calculation of exposure concentrations

C                             duration of exposure (days)
      tcoxp(1) = 0.00
      tcoxp(2) = 3.00
      tcoxp(3) = 21.00
      tcoxp(4) = 28.00

      do 103 ixnotot = 1, nxnotot
         do 102 i = 1, nwbsy

C                             initialisation of sum of product of
C                             pesticide concentration in water phase
C                             * time step, separately for each node
C                             selected for output
            if (its.eq.1 .and. ixnotot.eq.iwbsy(i)) then
               sumct(i) = zero
            end if

C                             calculation of sumct
            if (ixnotot.eq.iwbsy(i)) then
               sumct(i) = deltwl*cowljpl(ixnotot) + sumct(i)
            end if

C                             calculation of exposure concentrations
            if (ixnotot.eq.iwbsy(i)) then
               do 101 ii = 2, 4
                  if (nint(tcoxp(ii)).le.nint(ttot-0.5)) then
                     if ((tcoxp(ii)*86400.).ge.(its*deltwl) .and.
     &                   (tcoxp(ii)*86400.).lt.((its+1)*deltwl)) then
                        coxp(i,ii) = sumct(i)/(its*deltwl)
                     end if
                  else
                     coxp(i,ii) = -100.
                  end if
  101          continue
            end if

  102    continue
  103 continue

C                             output of exposure concentrations in output
C                             file icwlhy.out
      if (op_icwlhy.eq.1) then

         if (its.eq.1) then
            write (uowl, 104)
  104 format (//, ' Output from wlso.for' , /,
     &        ' Exposure concentrations at 0, 3, 21 and 28 days', /,
     &     ' node output   dist.in ditch   time    exp.concentration', /,
     &     '                    (m)         (d)        (g/m3)')
         end if

         do 107 ixnotot = 1, nxnotot
            do 106 i = 1, nwbsy
               if (ixnotot.eq.iwbsy(i)) then

                  if (its.eq.1) then
                     write (uowl, 105) ixnotot, xcd(ixnotot),
     &                                 tcoxp(1), coxp(i,1)
                  end if

                  if ((tcoxp(2)*86400.).ge.(its*deltwl) .and.
     &                (tcoxp(2)*86400.).lt.((its+1)*deltwl)) then
                     write (uowl, 105) ixnotot, xcd(ixnotot),
     &                                 tcoxp(2), coxp(i,2)
                  end if

                  if ((tcoxp(3)*86400.).ge.(its*deltwl) .and.
     &                (tcoxp(3)*86400.).lt.((its+1)*deltwl)) then
                     write (uowl, 105) ixnotot, xcd(ixnotot),
     &                                 tcoxp(3), coxp(i,3)
                  end if

                  if ((tcoxp(4)*86400.).ge.(its*deltwl) .and.
     &                (tcoxp(4)*86400.).lt.((its+1)*deltwl)) then
                     write (uowl, 105) ixnotot, xcd(ixnotot),
     &                                 tcoxp(4), coxp(i,4)
  105 format (I8, 6X, F8.2, 7X, F6.2, 2X, E12.5)
                  end if

               end if
  106       continue

            if (its.eq.ntswl .and. ixnotot.eq.nxnotot) then
               close (unit = uowl)
            end if

  107    continue

      end if

C---- End of calculation of exposure concentrations
```

```
C---- Output of calculated concentrations, lineic mass and percentages
C     of total mass present

      if (its.eq.1) then
         open (unit = uocl, file = 'wlsoco.out', status = 'unknown')
         write (uocl, 109)
  109 format (/, ' Output from subroutine wlso:')
      end if

      do 130 i = 1, 9
         if (its.eq.itsout(i)) then
            write (uocl, 110) its, itsout(i)*deltwl/86400.
  110 format (/,
     &         ' concentrations, lineic mass (mass per running metre',
     &         ' ditch) and', /, ' percentages for all nodes at',
     &         ' timestep:', I6, ' (', F8.4, ' day after application)',
     &        / , ' concentrations', /, ' nodenr      total        ',
     &        '    dissolved      at macrophytes  at susp solids',
     &        /, '                  c*               c              ',
     &        ' Xmp                Xss', /, '             g.m-3      ',
     &        '         g.m-3           g.g-1           g.g-1', /,
     &        ' lineic mass (g.m''-1)', /, ' percentage of total',
     &        ' present in water layer (%)')
            do 120 ixnotot= 1, nxnotot
              write (uocl, 111) ixnotot, cawljp1(ixnotot),
     &                cowljp1(ixnotot), sompjp1(ixnotot),
     &                sossjp1(ixnotot), warjp1(ixnotot)*cawljp1(ixnotot),
     &                warjp1(ixnotot)*cowljp1(ixnotot),
     &                dwmp*pez0hw*sompjp1(ixnotot),
     &                warjp1(ixnotot)*coss*sossjp1(ixnotot),
     &                cawljp1(ixnotot)/cawljp1(ixnotot)*100.,
     &                cowljp1(ixnotot)/cawljp1(ixnotot)*100.,
     &                (dwmp*pez0hw/warjp1(ixnotot))*
     &                sompjp1(ixnotot)/cawljp1(ixnotot)*100.,
     &                coss*sossjp1(ixnotot)/cawljp1(ixnotot)*100.
  111 format (I5, 4X, E12.4, 4X, E12.4, 4X, E12.4, 4X, E12.4,
     &        /, 11X, E12.4, 4X, E12.4, 4X, E12.4, 4X, E12.4,
     &        /, 13X, F6.2, 10X, F6.2, 10X, F6.2, 10X, F6.2)
  120       continue
         end if
  130 continue


      if (its.eq.ntswl) then
         close (unit = uocl)
      end if

      return


      end



      subroutine wlsoin

** SUBROUTINE:
**    wlsoin.for - concentrations and seepage rates are read for this
**                 timestep j of the TOXSWA program
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    Concentration in the liquid phase of the sediment, c_lb, in the
**    uppermost node of the sediment is read for each node of the water
**    layer as well as the seepage/infiltration rate plus
**    concentrations; also the concentration, c, for the water layer
**    at time j for nodes i-1, i, i+1 is read.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                              common variables
      include 'common.for'
C                              local variables
      integer     i
      real        qseifjp1

      if (its.eq.1) then
         cowbjkisl(ixnotot) = costwb(1)
      else
         cowbjkisl(ixnotot) = cowbjp1kisl(ixnotot)
      end if


C--- Calculation seepage rate plus concentration between neighbouring
C    lot and ditch per timestep

      do 10 i = 1, 30
         if ((its*deltwl).gt.((i-1)*86400.) .and.
     &                      (its*deltwl).le.(i*86400.)) then
C                             now daynr i determined for timestep its
            if (((its+1)*deltwl).ge.(i*86400.)) then
C                             now timestep its+1 is at the next day
               qseifj = qseif(i)
               qseifjp1 = qseif(i+1)
               colotj = colot(i)
               colotjp1 = colot(i+1)
            else
               if (((its+1)*deltwl).lt.(i*86400.)) then
C                                now timestep its+1 is at the same day
                  qseifj = qseif(i)
                  qseifjp1 = qseif(i)
                  colotj = colot(i)
                  colotjp1 = colot(i)
               end if
            end if
         end if
      qseifjph = thetawl*qseifj+(1.-thetawl)*qseifjp1
   10 continue


      return


      end



      subroutine wlsoge

** SUBROUTINE:
**    wlsoge.for - some geometrical (time-dependant) characteristics
**                 are calculated here for the water layer of the
**                 TOXSWA program
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    Wetted cross section areas, A, water depths, h, discharges and
**    widths water surfaces, Ox, are calculated here for time j, j+1/2,
**    j+1 at nodes and segment boundaries.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                              common variables
      include 'common.for'
C                              local variables
      real        wdhjphi, wdhjimh, wdhjp1imh, wdhjiph, wdhjp1iph


C---- calculate Aj,i-1, Aj,i+1, Aj+1,i-1, Aj+1,i+1
C                Aj,i, Aj+1,i, Aj+1/2,i, Aj+1/2,i-1/2, Aj+1/2,i+1/2
C                Aj,i+1/2, Aj+1,i+1/2, Aj,i-1/2, Aj+1,i-1/2
C                ufb, ueb
C                Qj+1/2,i-1/2, Qj+1/2,i+1/2
C                Oxj+1/2,i

      warj(ixnotot) = wibot*wdhji + (wdhji**2)*sisl
```

```
      warjpl(ixnotot) = wibot*wdhjpli + (wdhjpli**2)*sisl
      warjph(ixnotot) = thetawl*warj(ixnotot) +
     &                  (1.-thetawl)*warjpl(ixnotot)

      if (ixnotot.gt.1) then
         wdhjimh = (delx(ixnotot)/(delx(ixnotot-1)+
     &             delx(ixnotot)))*wdhjiml + (delx(ixnotot-1)/
     &             (delx(ixnotot-1)+delx(ixnotot)))*wdhji
         wdhjplimh = (delx(ixnotot)/(delx(ixnotot-1)+
     &               delx(ixnotot)))*wdhjpliml +
     &               (delx(ixnotot-1)/(delx(ixnotot-1)+
     &               delx(ixnotot)))*wdhjpli
      else
         if (ixnotot.eq.1) then
            wdhjimh = wdhji
            wdhjplimh = wdhjpli
         end if
      end if

      if (ixnotot.lt.nxnotot) then
         wdhjiph = (delx(ixnotot+1)/(delx(ixnotot)+
     &             delx(ixnotot+1)))*wdhji + (delx(ixnotot)/
     &             (delx(ixnotot)+delx(ixnotot+1)))*wdhjipl
         wdhjpliph = (delx(ixnotot+1)/(delx(ixnotot)+
     &               delx(ixnotot+1)))*wdhjpli +
     &               (delx(ixnotot)/(delx(ixnotot)+
     &               delx(ixnotot+1)))*wdhjplipl
      else
         if (ixnotot.eq.nxnotot) then
            wdhjiph = wdhji
            wdhjpliph = wdhjpli
         end if
      end if

      warjimh(ixnotot)    = wibot*wdhjimh + (wdhjimh**2)*sisl
      warjplimh = wibot*wdhjplimh + (wdhjplimh**2)*sisl
      warjiph(ixnotot)    = wibot*wdhjiph + (wdhjiph**2)*sisl
      warjpliph = wibot*wdhjpliph + (wdhjpliph**2)*sisl

      warjphimh(ixnotot) = thetawl*warjimh(ixnotot) +
     &                                 (1.-thetawl)*warjplimh
      warjphiph(ixnotot) = thetawl*warjiph(ixnotot) +
     &                                 (1.-thetawl)*warjpliph


C                             flow in front buffer, ufb towards first
C                             node will indicate flow direction at time j
C                             for subroutine wlsoub
      if (ixnotot.eq.1) then
         ufb = qvojimh/warjimh(ixnotot)
      end if

C                             flow in end buffer, ueb, towards first
C                             node will indicate flow direction at time j
C                             for subroutine wlsolb
      if (ixnotot.eq.nxnofb+nxnodit) then
         ueb = qvojiph/warjiph(ixnotot)
      end if

C                             u, wdh and qvo-dit indicate flow, water
C                             depth and discharge at first node in ditch
C                             at time j+1 and will be used in subroutine
C                             wlout
      if (ixnotot.eq.nxnofb+1) then
         uditin = qvojplimh/warjplimh
         wdhditin = wdhjpli
         qvoditin = qvojplimh
      end if

C                             u, wdh and qvo-dit indicate flow, water
C                             depth and discharge at last node in ditch
C                             at time j+1 and will be used in subroutine
C                             wlout
      if (ixnotot.eq.nxnofb+nxnodit) then
         uditout = qvojpliph/warjpliph
         wdhditout = wdhjpli
         qvoditout = qvojpliph
      end if
```

```
      qvojphimh(ixnotot) = thetawl*qvojimh + (1.-thetawl)*qvojplimh
      qvojphiph(ixnotot) = thetawl*qvojiph + (1.-thetawl)*qvojpliph

      witopj(ixnotot) = wibot + 2.*wdhji*sisl
      wdhjphi = thetawl*wdhji + (1.-thetawl)*wdhjpli
      witopjph(ixnotot) = wibot + 2.*wdhjphi*sisl

      if (op_icwlhy.eq.1) then
         if (its.eq.1 .and. ixnotot.eq.1) then
            write (uowl, 10)
 10   format (/, ' Results of calculations from subroutine wlsoge',
     &        ' (for first timestep):')
            write (uowl, 11)
 11   format (' first water depth at interfaces for all nodes:', /,
     &        '      ixnotot      wdhjimh        wdhjiph        (m)')
            write (uowl, 12)
 12   format (' then wetted area at interfaces at time j+1/2',
     &        ' for all nodes:', /,
     &        '                   warjphimh      warjphiph      (m2)')
         end if
         if (its.eq.1) then
            write (uowl, 13)
 13   format (/,
     &        '      ixnotot      wdhjimh        wdhjiph        (m)', /,
     &        '                   warjphimh      warjphiph      (m2)')
            write (uowl, *) ixnotot, wdhjimh, wdhjiph
            write (uowl, 14) warjphimh(ixnotot), warjphiph(ixnotot)
 14   format (15X, F9.7, 7X, F9.7)
         end if
      end if

      return


      end



      subroutine wlsods

** SUBROUTINE:
**    wlsods.for  - the physical dispersion coefficient in water layer
**                  of the TOXSWA program is corrected for the numerical
**                  dispersion
**    author      - P.I. Adriaanse
**
** DESCRIPTION:
**    The physical dispersion coefficient for the water layer is
**    corrected for the numerical dispersion. The calculation is done at
**    segment boundaries and at time j+1/2; the calculation dispersion
**    equals the physical dispersion minus the numerical dispersion.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                             common variables
      include 'common.for'
C                             local variables
      real        cowljimh(mxnxnotot), cowljiph(mxnxnotot),
     &            kdswnujimh, kdswnujiph

C---- Numerical dispersion has been calculated with c at time j instead
C     of time j+1/2

      if (ixnotot.gt.1) then
         cowljimh(ixnotot) = (1.-nwwmh(ixnotot))*cowlj(ixnotot-1) +
     &                       nwwmh(ixnotot)*cowlj(ixnotot)
      end if
      if (ixnotot.lt.nxnotot) then
         cowljiph(ixnotot) = (1.-nwwph(ixnotot))*cowlj(ixnotot) +
     &                       nwwph(ixnotot)*cowlj(ixnotot+1)
      end if
```

```fortran
c                          prevent undefined exponentiation in
c                          calculation of Enum
      if (cowljimh(ixnotot).le.zero) then
         cowljimh(ixnotot) = tiny
      end if
      if (cowljiph(ixnotot).le.zero) then
         cowljiph(ixnotot) = tiny
      end if

c                          No dispersion terms Ei-1/2 exist for
c                          ixnotot=1 and no dispersion terms Ei+1/2
c                          exist for ixnotot=nxnotot (they do not
c                          appear in the two tridiagonal matrices)
      if (ixnotot.gt.1) then
         kdswnujimh = (qvojimh/(2.*warjimh(ixnotot))) *
     &                ((1.-2.*nwwmh(ixnotot))
     &                *(xcd(ixnotot)-xcd(ixnotot-1))+(1.-2.*thetawl)*
     &                (qvojimh/warjimh(ixnotot))*deltwl *
     &                (1.+coss*kdfrss(ixnotot)*exfrss*
     &                ((cowljimh(ixnotot)/coobkomss)**(exfrss-1.))) /
     &                (1.+dwmp*pez0hw*kdmp(ixnotot)/warjimh(ixnotot) +
     &                coss*kdfrss(ixnotot)*exfrss*
     &                ((cowljimh(ixnotot)/coobkomss)**(exfrss-1.))))
      end if
      if (ixnotot.lt.nxnotot) then
         kdswnujiph = (qvojiph/(2.*warjiph(ixnotot))) *
     &                ((1.-2.*nwwph(ixnotot))
     &                *(xcd(ixnotot+1)-xcd(ixnotot))+(1.-2.*thetawl)*
     &                (qvojiph/warjiph(ixnotot))*deltwl *
     &                (1.+coss*kdfrss(ixnotot)*exfrss*
     &                ((cowljiph(ixnotot)/coobkomss)**(exfrss-1.))) /
     &                (1.+dwmp*pez0hw*kdmp(ixnotot)/warjiph(ixnotot) +
     &                coss*kdfrss(ixnotot)*exfrss*
     &                ((cowljiph(ixnotot)/coobkomss)**(exfrss-1.))))
      end if

      if (ixnotot.gt.1) then
         kdswcljimh(ixnotot) = kds - kdswnujimh
      end if
      if (ixnotot.lt.nxnotot) then
         kdswcljiph(ixnotot) = kds - kdswnujiph
      end if

      if (op_icwlhy.eq.1) then
         if (its.eq.1 .and. ixnotot.eq.1) then
            write (uowl, 5)
    5 format (/, ' Some calculation results from subroutine wlsods:', /,
     &        ' physical, numerical and calculation dispersion for',
     &        /, ' first timestep and first node',
     &        /, ' kds  kdswnujiph  kdswcljiph  (m2.(day)-1)')
            write (uowl, *) 86400.*kds, 86400.*kdswnujiph,
     &                      86400.*kdswcljiph(1)
         end if
      end if


      return


      end


      subroutine wlsoem

** SUBROUTINE:
**    wlsoem.for - the elements of the tridiagonal matrices for the
**                 water layer of the TOXSWA program are defined
**                 here
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The elements composing the two tridiagonal matrices for the water
**    layer are defined here; these elements are used in the subroutines
**    wlsolb, wlsoub and wlsocr.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

c                          common variables
      include 'common.for'
c                          local variables


c---- Elements of the core and buffer matrices will be composed here


c                          the terms lod.., lddadmh, ldddsmh, rod..,
c                          rddadmh and rdddsmh will not be defined for
c                          ixnotot=1, as well as lddadph, ldddsph,
c                          lbd.., rddadph, rdddsph and rbd.. for
c                          ixnotot=nxnotot, because they do not
c                          appear in the two tridiagonal matrices

      if (ixnotot.gt.1) then
         lodad        = (qvojphimh(ixnotot)/delx(ixnotot))*(1.-thetawl)*
     &                  (1.-nwwmh(ixnotot))*deltwl * (1.+coss*
     &                  kdfrss(ixnotot-1)*
     &                 ((oldcowljp1(ixnotot-1)/coobkomss)**(exfrss-1.)))
         lodds        = (2.*warjphimh(ixnotot)*kdswcljimh(ixnotot))/
     &                  (delx(ixnotot)*(delx(ixnotot)+delx(ixnotot-1)))
     &                  * (1.-thetawl)* deltwl *
     &                  (1.+coss*kdfrss(ixnotot-1)*
     &                  ((oldcowljp1(ixnotot-1)/coobkomss)**
     &                  (exfrss-1.)))

         lddadmh      = (qvojphimh(ixnotot)/delx(ixnotot))*(1.-thetawl)*
     &                  nwwmh(ixnotot)*deltwl * (1.+coss*
     &                  kdfrss(ixnotot)*
     &                  ((oldcowljp1(ixnotot)/coobkomss)**(exfrss-1.)))
      end if
      if (ixnotot.lt.nxnotot) then
         lddadph      = (qvojphiph(ixnotot)/delx(ixnotot))*(1.-thetawl)*
     &                  (1.-nwwph(ixnotot))*deltwl * (1.+coss*
     &                  kdfrss(ixnotot)*
     &                  ((oldcowljp1(ixnotot)/coobkomss)**(exfrss-1.)))
      end if
      if (ixnotot.gt.1) then
         ldddsmh      = (2.*warjphimh(ixnotot)*kdswcljimh(ixnotot))/
     &                  (delx(ixnotot)*(delx(ixnotot)+delx(ixnotot-1)))
     &                  * (1.-thetawl)*deltwl *
     &                  (1.+coss*kdfrss(ixnotot)*
     &                  ((oldcowljp1(ixnotot)/coobkomss)**(exfrss-1.)))
      end if
      if (ixnotot.lt.nxnotot) then
         ldddsph      = (2.*warjphiph(ixnotot)*kdswcljiph(ixnotot))/
     &                  (delx(ixnotot)*(delx(ixnotot+1)+delx(ixnotot)))
     &                  * (1.-thetawl)*deltwl *
     &                  (1.+coss*kdfrss(ixnotot)*
     &                  ((oldcowljp1(ixnotot)/coobkomss)**(exfrss-1.)))
      end if
      lddca        = (1.+((dwmp*pez0hw)/warjp1(ixnotot))*kdmp(ixnotot)
     &               +coss*kdfrss(ixnotot)*
     &               ((oldcowljp1(ixnotot)/coobkomss)**(exfrss-1.))) *
     &               (warjp1(ixnotot))
      lddtf        = (1.+((dwmp*pez0hw)/warjp1(ixnotot))*kdmp(ixnotot)+
     &               coss*kdfrss(ixnotot)*
     &               ((oldcowljp1(ixnotot)/coobkomss)**(exfrss-1.))) *
     &               (krtfwl*warjph(ixnotot)*(1.-thetawl)*
     &               deltwl)
      lddai        =  ktl*witopjph(ixnotot)*(1.-thetawl)*deltwl
      lddwbad      = (leplot/pez0hw)*qseifjph*pez0hw*
     &               (1.-thetawl)*deltwl
      lddwbdf      = ((2.*por(1)*kdfwbmh(1)*pez0hw)/delz(1))*
     &               (1.-thetawl)*deltwl


      if (ixnotot.lt.nxnotot) then
         lbdad        = (qvojphiph(ixnotot)/delx(ixnotot))*(1.-thetawl)*
     &                 nwwph(ixnotot)*deltwl*(1.+coss*kdfrss(ixnotot+1)*
     &                 ((oldcowljp1(ixnotot+1)/coobkomss)**(exfrss-1.)))
         lbdds        = ((2.*warjphiph(ixnotot)*kdswcljiph(ixnotot))/
     &                  (delx(ixnotot)*(delx(ixnotot+1)+delx(ixnotot))))
     &                  * (1.-thetawl)*deltwl *
```

```fortran
     &                    (1.+coss*kdfrss(ixnotot+1)*
     &                    ((oldcowljp1(ixnotot+1)/coobkomss)**
     &                    (exfrss-1.)))
      end if


      if (ixnotot.gt.1) then
         rodad          = (qvojphimh(ixnotot)/delx(ixnotot))*thetawl*
     &                    (1.-nwwmh(ixnotot))*deltwl * (1.+coss*
     &                    kdfrss(ixnotot-1)*
     &                    ((cowlj(ixnotot-1)/coobkomss)**(exfrss-1.)))
         rodds          = ((2.*warjphimh(ixnotot)*kdswcljimh(ixnotot))/
     &                    (delx(ixnotot)*(delx(ixnotot)+delx(ixnotot-1))))
     &                    * thetawl*deltwl *
     &                    (1.+coss*kdfrss(ixnotot-1)*
     &                    ((cowlj(ixnotot-1)/coobkomss)**(exfrss-1.)))

         rddadmh        = (qvojphimh(ixnotot)/delx(ixnotot))*thetawl*
     &                    nwwmh(ixnotot)*deltwl *
     &                    (1.+coss*kdfrss(ixnotot)*
     &                    ((cowlj(ixnotot)/coobkomss)**(exfrss-1.)))
      end if
      if (ixnotot.lt.nxnotot) then
         rddadph        = (qvojphiph(ixnotot)/delx(ixnotot))*thetawl*(1.-
     &                    nwwph(ixnotot))*deltwl * (1.+coss*
     &                    kdfrss(ixnotot)*
     &                    ((cowlj(ixnotot)/coobkomss)**(exfrss-1.)))
      end if
      if (ixnotot.gt.1) then
         rdddsmh        = ((2.*warjphimh(ixnotot)*kdswcljimh(ixnotot))/
     &                    (delx(ixnotot)*(delx(ixnotot)+delx(ixnotot-1))))
     &                    * thetawl*deltwl *
     &                    (1.+coss*kdfrss(ixnotot)*
     &                    ((cowlj(ixnotot)/coobkomss)**(exfrss-1.)))
      end if
      if (ixnotot.lt. nxnotot) then
         rdddsph        = ((2.*warjphiph(ixnotot)*kdswcljiph(ixnotot))/
     &                    (delx(ixnotot)*(delx(ixnotot+1)+delx(ixnotot))))
     &                    * thetawl*deltwl *
     &                    (1.+coss*kdfrss(ixnotot)*
     &                    ((cowlj(ixnotot)/coobkomss)**(exfrss-1.)))
      end if
      rddca          = (1.+((dwmp*pez0hw)/warj(ixnotot))*kdmp(ixnotot)
     &                 +coss*kdfrss(ixnotot)*
     &                 ((cowlj(ixnotot)/coobkomss)**(exfrss-1.))) *
     &                 (warj(ixnotot))
      rddtf          = (1.+((dwmp*pez0hw)/warj(ixnotot))*kdmp(ixnotot)
     &                 +coss*kdfrss(ixnotot)*
     &                 ((cowlj(ixnotot)/coobkomss)**(exfrss-1.))) *
     &                 (krtfwl*warjph(ixnotot)*thetawl*deltwl)
      rddai          = ktl*witopjph(ixnotot)*thetawl*deltwl
      rddwbad        = (leplot/pez0hw)*qseifjph*pez0hw*thetawl*
     &                 deltwl
      rddwbdf        = ((2.*por(1)*kdfwbmh(1)*pez0hw)/delz(1))*thetawl*
     &                 deltwl


      if (ixnotot.lt.nxnotot) then
         rbdad          = (qvojphiph(ixnotot)/delx(ixnotot))*thetawl*
     &                    nwwph(ixnotot)*deltwl *
     &                    (1.+coss*kdfrss(ixnotot+1)*
     &                    ((cowlj(ixnotot+1)/coobkomss)**(exfrss-1.)))
         rbdds          = ((2.*warjphiph(ixnotot)*kdswcljiph(ixnotot))/
     &                    (delx(ixnotot)*(delx(ixnotot+1)+delx(ixnotot))))
     &                    * thetawl*deltwl *
     &                    (1.+coss*kdfrss(ixnotot+1)*
     &                    ((cowlj(ixnotot+1)/coobkomss)**(exfrss-1.)))
      end if

c                          calculation for wb will be done after the
c                          calculation of wl, (to prevent simultaneous
c                          solution for wl and wb, which will be very
c                          extensive), so instead of c_lb at j+1/2 and
c                          for k=1 the term c_lb at j for k=1
c                          (cowbjkis1(1-nxnotot)) will be
c                          used in the rv term (see also section 6.9
c                          of SC-DLO report 90)

      rvai           = ktl*witopjph(ixnotot)*(coair/khe)*deltwl
      rvwbdf         = ((2.*por(1)*kdfwbmh(1)*pez0hw)/delz(1)) *
     &                 cowbjkis1(ixnotot) * deltwl
      rvwbad         = (leplot/pez0hw)*qseifjph*pez0hw*cowbjkis1(ixnotot)*
     &                 deltwl

      return

      end


      subroutine wlsolb

** SUBROUTINE:
**    wlsolb.for - the lod, ldd etc terms of the matrices for the lower
**                 boundary of the water subsystem of the TOXSWA
**                 program are composed here
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The matrix for the last node or end buffer will be composed for
**    this timestep. (Only the nonzero terms of the left-hand and
**    right-hand tridiagonal matrices will be stored, they form two
**    times three vectors; notice that lod(1), lbd(nxnotot), rod(1)
**    and rbd(nxnotot) are undefined and not referenced in the
**    subroutines.)
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

c                                common variables
      include 'common.for'
c                                local variables

      if (ixnotot.eq.nxnofb+nxnodit .and. nxnoeb.ne.0) then
c                                normal internal node of core
         call wlsocr
      else
         if (ixnotot.eq.nxnofb+nxnodit .and. nxnoeb.eq.0) then
c                                last node of ditch
            if (qseifjph.ge.zero) then
               if (ueb.ge.zero) then
                  lod(ixnotot) = -lodad - lodds
                  ldd(ixnotot) = -lddadmh +
     &                           (qvojphiph(ixnotot)/delx(ixnotot))*
     &                           (1.-thetawl)*deltwl
     &                           * (1.+coss*kdfrss(ixnotot)*
     &                           ((oldcowljp1(ixnotot)/coobkomss)**
     &                                              (exfrss-1.)))
     &                           + ldddsmh +
     &                           lddca +lddtf + lddai + lddwbad + lddwbdf
                  rod(ixnotot) = rodad + rodds
                  rdd(ixnotot) = rddadmh -
     &                            (qvojphiph(ixnotot)/delx(ixnotot))
     &                            *thetawl*deltwl *
     &                            (1.+coss*kdfrss(ixnotot)*
     &                            ((cowlj(ixnotot)/coobkomss)**
     &                                                  (exfrss-1.)))
     &                            - rdddsmh +
     &                            rddca - rddtf - rddai -rddwbad -rddwbdf
                  rv(ixnotot)  = rvai + rvwbdf
               else
c                                now ueb less than zero
                  lod(ixnotot) = -lodad - lodds
                  ldd(ixnotot) = -lddadmh + ldddsmh +
     &                           lddca +lddtf + lddai + lddwbad + lddwbdf
                  rod(ixnotot) = rodad + rodds
                  rdd(ixnotot) = rddadmh - rdddsmh +
     &                            rddca - rddtf - rddai -rddwbad -rddwbdf
                  rv(ixnotot)  = rvai + rvwbdf
               end if
```

```
         else
C                        now qseifjph less than zero

            if (ueb.ge.zero) then
               lod(ixnotot) = -lodad - lodds
               ldd(ixnotot) = -lddadmh +
     &                       (qvojphiph(ixnotot)/delx(ixnotot))*
     &                       (1.-thetawl)*deltwl
     &                       * (1.+coss*kdfrss(ixnotot)*
     &                       ((oldcowljp1(ixnotot)/coobkomss)**
     &                                           (exfrss-1.)))
     &                       + ldddsmh +
     &                       lddca +lddtf + lddai + lddwbdf
               rod(ixnotot) = rodad + rodds
               rdd(ixnotot) = rddadmh -
     &                        (qvojphiph(ixnotot)/delx(ixnotot))
     &                        *thetawl*deltwl *
     &                        (1.+coss*kdfrss(ixnotot)*
     &                        ((cowlj(ixnotot)/coobkomss)**
     &                                             (exfrss-1.)))
     &                        - rdddsmh +
     &                        rddca - rddtf - rddai - rddwbdf
               rv(ixnotot)  = rvai + rvwbdf - rvwbad
            else
C                        now ueb less than zero
               lod(ixnotot) = -lodad - lodds
               ldd(ixnotot) = -lddadmh + ldddsmh +
     &                       lddca +lddtf + lddai + lddwbdf
               rod(ixnotot) = rodad + rodds
               rdd(ixnotot) = rddadmh - rdddsmh +
     &                        rddca - rddtf - rddai - rddwbdf
               rv(ixnotot)  = rvai + rvwbdf - rvwbad
            end if

         end if
      end if
   end if


   if (ixnotot.eq.nxnotot .and. nxnoeb.ne.0) then
C                        last row elements in matrix for lower
C                        boundary; in case of an end buffer
C                        first determine direction of flow u

      if (ueb.ge.zero) then
         lod(ixnotot) = -lodad - lodds
         ldd(ixnotot) = -lddadmh +
     &                 (qvojphiph(ixnotot)/delx(ixnotot))*
     &                 (1.-thetawl)*deltwl
     &                 * (1.+coss*kdfrss(ixnotot)*
     &                ((oldcowljp1(ixnotot)/coobkomss)**(exfrss-1.)))
     &                 + ldddsmh
     &                 + (1.+coss*kdfrss(ixnotot)*
     &                ((oldcowljp1(ixnotot)/coobkomss)**(exfrss-1.)))
     &                 *warjp1(ixnotot)
         rod(ixnotot) = rodad + rodds
         rdd(ixnotot) = rddadmh -
     &                 (qvojphiph(ixnotot)/delx(ixnotot))*thetawl*
     &                 deltwl *
     &                 (1.+coss*kdfrss(ixnotot)*
     &                 ((cowlj(ixnotot)/coobkomss)**(exfrss-1.)))
     &                 - rdddsmh +
     &                 (1.+coss*kdfrss(ixnotot)*
     &                 ((cowlj(ixnotot)/coobkomss)**(exfrss-1.)))
     &                 *warj(ixnotot)
         rv(ixnotot)  = zero
      else
         if (ueb.lt.zero) then
            lod(ixnotot) = -lodad - lodds
            ldd(ixnotot) = -lddadmh + ldddsmh +
     &                     (1.+coss*kdfrss(ixnotot)*
     &                     ((oldcowljp1(ixnotot)/coobkomss)**
     &                     (exfrss-1.))) *warjp1(ixnotot)
            rod(ixnotot) = rodad + rodds
            rdd(ixnotot) = rddadmh - rdddsmh +
     &                     (1.+coss*kdfrss(ixnotot)*
     &                     ((cowlj(ixnotot)/coobkomss)**(exfrss-1.)))
     &                     *warj(ixnotot)
            rv(ixnotot)  = zero
```

```
            end if
         end if

      else
         if (ixnotot.ne.nxnofb+nxnodit) then
C                        terms lod, ldd, lbd, rod, rdd, rbd and rv
C                        for grid points m+1 until ebt-1 included
C                        in the end buffer
            lod(ixnotot) = -lodad - lodds
            ldd(ixnotot) = -lddadmh + lddadph + ldddsmh + ldddsph +
     &                   (1.+coss*kdfrss(ixnotot)*
     &                   ((oldcowljp1(ixnotot)/coobkomss)**
     &                           (exfrss-1.)))*warjp1(ixnotot)
            lbd(ixnotot) = lbdad - lbdds
            rod(ixnotot) = rodad + rodds
            rdd(ixnotot) = rddadmh - rddadph - rdddsmh - rdddsph +
     &                   (1.+coss*kdfrss(ixnotot)*
     &                   ((cowlj(ixnotot)/coobkomss)**(exfrss-1.)))
     &                   *warj(ixnotot)
            rbd(ixnotot) = -rbdad + rbdds
            rv(ixnotot)  = zero
         end if

      end if

      return


      end



      subroutine wlsoub

** SUBROUTINE:
**    wlsoub.for - the lod, ldd etc terms of the matrices for the upper
**                 boundary of the water layer of the TOXSWA program
**                 are composed here
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The matrices for the first node or front buffer of the water
**    layer will be composed for this timestep. (Only the nonzero terms
**    of the left-hand and right-hand tridiagonal matrices will be
**    stored, they form two times three vectors; notice that lod(1),
**    lbd(nxnotot), rod(1) and rbd(nxnotot) are undefined and not
**    referenced in the subroutines.)
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                        common variables
      include 'common.for'
C                        local variables


C                        flow velocity has been positive since start
C                        calculations: no front buffer needed,
C                        nxnofb=0 and first row core matrix is
C                        composed
      if (nxnofb.eq.0 .and. ixnotot.eq.1 .and. ufb.ge.zero) then
         if (qseifjph.ge.zero) then
            ldd(1)       = lddadph + ldddsph + lddca +lddtf + lddai +
     &                     lddwbad + lddwbdf
            lbd(1)       = lbdad - lbdds
            rdd(1)       = -rddadph - rdddsph + rddca - rddtf - rddai -
     &                     rddwbad - rddwbdf
            rbd(1)       = -rbdad + rbdds
            rv(1)         = rvai + rvwbdf
         else
            if (qseifjph.lt.zero) then
               ldd(1)       = lddadph + ldddsph + lddca +lddtf + lddai +
     &                        lddwbdf
               lbd(1)       = lbdad - lbdds
```

```
            rdd(1)        = -rddadph - rdddsph + rddca - rddtf -
     &                      rddai - rddwbdf
            rbd(1)        = -rbdad + rbdds
            rv(1)          = rvai + rvwbdf - rvwbad
          end if
        end if
      end if

c                           flow velocity has been negative since start
c                           calculations: no front buffer needed,
c                           nxnofb=0 and first row core matrix is
c                           composed
      if (nxnofb.eq.0 .and. ixnotot.eq.1 .and. ufb.lt.zero) then
        if (qseifjph.ge.zero) then
          ldd(1)        = lddadph  -(qvojphimh(ixnotot)/delx(ixnotot))*
     &                 (1.-thetawl)*deltwl *(1.+coss*kdfrss(ixnotot)*
     &                 ((oldcowljpl(ixnotot)/coobkomss)**(exfrss-1.)))
     &                   + ldddsph + lddca +lddtf + lddai +
     &                  lddwbad + lddwbdf
          lbd(1)        = lbdad - lbdds
          rdd(1)        = -rddadph
     &                    + (qvojphimh(ixnotot)/delx(ixnotot))*thetawl*
     &                  deltwl* (1.+coss*kdfrss(ixnotot)*
     &                  ((cowlj(ixnotot)/coobkomss)**(exfrss-1.)))
     &                  - rdddsph + rddca - rddtf - rddai -
     &                  rddwbad - rddwbdf
          rbd(1)        = -rbdad + rbdds
          rv(1)          = rvai + rvwbdf
        else
          if (qseifjph.lt.zero) then
            ldd(1)        = lddadph
     &                      -(qvojphimh(ixnotot)/delx(ixnotot))*
     &                 (1.-thetawl)*deltwl *(1.+coss*kdfrss(ixnotot)*
     &                 ((oldcowljpl(ixnotot)/coobkomss)**(exfrss-1.)))
     &                     + ldddsph + lddca +lddtf + lddai +
     &                     lddwbdf
            lbd(1)        = lbdad - lbdds
            rdd(1)        = -rddadph
     &                  +(qvojphimh(ixnotot)/delx(ixnotot))*thetawl*
     &                   deltwl* (1.+coss*kdfrss(ixnotot)*
     &                   ((cowlj(ixnotot)/coobkomss)**(exfrss-1.)))
     &                     - rdddsph + rddca - rddtf -
     &                     rddai - rddwbdf
            rbd(1)        = -rbdad + rbdds
            rv(1)          = rvai + rvwbdf - rvwbad
          end if
        end if
      end if


c                           flow velocity has been negative during at
c                           least one timestep of the calculations:
c                           front buffer is needed and its matrix is
c                           composed

      if (nxnofb.ne.0 .and. ixnotot.gt.1 .and. ixnotot.lt.nxnofb+1) then
c                           now elements front buffer for gridpoints 2
c                           until fbt included
        lod(ixnotot) = -lodad - lodds
        ldd(ixnotot) = -lddadmh + lddadph + ldddsmh + ldddsph +
     &               (1.+coss*kdfrss(ixnotot)*
     &               ((oldcowljpl(ixnotot)/coobkomss)**(exfrss-1.)))
     &               *warjpl(ixnotot)
        lbd(ixnotot) = lbdad - lbdds
        rod(ixnotot) = rodad + rodds
        rdd(ixnotot) = rddadmh - rddadph - rdddsmh - rdddsph +
     &                (1.+coss*kdfrss(ixnotot)*
     &                ((cowlj(ixnotot)/coobkomss)**(exfrss-1.))) *
     &                warj(ixnotot)
        rbd(ixnotot) = -rbdad + rbdds
        rv(ixnotot)  = zero
      end if

      if (nxnofb.ne.0 .and. ixnotot.eq.1 .and. ufb.lt.0) then
c                           now elements first row, fb=1, of matrix in
c                           front buffer for negative flow direction
        ldd(1)        = -(qvojphimh(ixnotot)/delx(ixnotot))*
     &               (1.-thetawl)*deltwl *(1.+coss*kdfrss(ixnotot)*
     &               ((oldcowljpl(ixnotot)/coobkomss)**(exfrss-1.)))
     &               + lddadph + ldddsph +
     &               (1.+coss*kdfrss(ixnotot)*
     &               ((oldcowljpl(ixnotot)/coobkomss)**(exfrss-1.)))*
     &               warjpl(ixnotot)
        lbd(1)        = lbdad - lbdds
        rdd(1)        = (qvojphimh(ixnotot)/delx(ixnotot))*thetawl*
     &               deltwl* (1.+coss*kdfrss(ixnotot)*
     &               ((cowlj(ixnotot)/coobkomss)**(exfrss-1.))) -
     &               rddadph - rdddsph +
     &               (1.+coss*kdfrss(ixnotot)*
     &               ((cowlj(ixnotot)/coobkomss)**(exfrss-1.)))
     &               * warj(ixnotot)
        rbd(1)        = -rbdad + rbdds
        rv(1)         = zero
      end if

      if (nxnofb.ne.0 .and. ixnotot.eq.1 .and. ufb.gt.zero) then
c                           now elements first row, fb=1, of matrix in
c                           front buffer for positive flow direction
        ldd(1)        = lddadph + ldddsph +
     &               (1.+coss*kdfrss(ixnotot)*
     &               ((oldcowljpl(ixnotot)/coobkomss)**(exfrss-1.)))*
     &               warjpl(ixnotot)
        lbd(1)        = lbdad - lbdds
        rdd(1)        = -rddadph - rdddsph +
     &               (1.+coss*kdfrss(ixnotot)*
     &               ((cowlj(ixnotot)/coobkomss)**(exfrss-1.)))
     &               *warj(ixnotot)
        rbd(1)        = -rbdad + rbdds
        rv(1)         = zero
      else
        if (nxnofb.ne.0 .and. ixnotot.eq.1) then
c                           now elements first row for no flow, ufb=0
        ldd(1)        = -(qvojphimh(ixnotot)/delx(ixnotot))*(1.-thetawl)
     &               *deltwl *(1.+coss*kdfrss(ixnotot)*
     &               ((oldcowljpl(ixnotot)/coobkomss)**(exfrss-1.)))
     &               + lddadph + ldddsph +
     &               (1.+coss*kdfrss(ixnotot)*
     &               ((oldcowljpl(ixnotot)/coobkomss)**(exfrss-1.)))*
     &               warjpl(ixnotot)
        lbd(1)        = lbdad - lbdds
        rdd(1)        = (qvojphimh(ixnotot)/delx(ixnotot))*thetawl*
     &               deltwl* (1.+coss*kdfrss(ixnotot)*
     &               ((cowlj(ixnotot)/coobkomss)**(exfrss-1.))) -
     &               rddadph - rdddsph +
     &               (1.+coss*kdfrss(ixnotot)*
     &               ((cowlj(ixnotot)/coobkomss)**(exfrss-1.)))
     &               * warj(ixnotot)
        rbd(1)        = -rbdad + rbdds
        rv(1)         = zero
        end if
      end if

      if (nxnofb.ne.0 .and. ixnotot.eq.nxnofb+1) then
        call wlsocr
      end if
c                           now element nxnofb+1 will be defined in
c                           subroutine  wlsocr (as it is no first row
c                           of the entire matrix and it does not belong
c                           to the front buffer)

      return

      end


      subroutine wlsocr

** SUBROUTINE:
**    wlsocr.for - the core parts of the matrices, consisting of the
**                 numerical equations to solve the mass conservation
**                 equation for the water layer of the TOXSWA program
**                 are composed here
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The core parts of the matrices will be composed for this timestep.
**    (Only the nonzero components of the left-hand and right-hand
**    tridiagonal matrices will be stored, they form two times three
```

```fortran
**    vectors; notice that lod(1), lbd(nxnotot), rod(1) and rbd(nxnotot)
**    are undefined and not referenced in the subroutines.)
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                              common variables
      include 'common.for'
C                              local variables

      if (qseifjph.ge.zero) then
         lod(ixnotot) = -lodad - lodds
         ldd(ixnotot) = -lddadmh + lddadph + ldddsmh + ldddsph + lddca +
     &                  lddtf + lddai + lddwbad +lddwbdf
         lbd(ixnotot) = lbdad - lbdds

         rod(ixnotot) = rodad + rodds
         rdd(ixnotot) = rddadmh - rddadph - rdddsmh - rdddsph + rddca -
     &                  rddtf - rddai - rddwbad - rddwbdf
         rbd(ixnotot) = -rbdad + rbdds

         rv(ixnotot)  = rvai + rvwbdf
      else
C                              qseifjph is negative
         lod(ixnotot) = -lodad - lodds
         ldd(ixnotot) = -lddadmh + lddadph + ldddsmh + ldddsph + lddca +
     &                  lddtf + lddai + lddwbdf
         lbd(ixnotot) = lbdad - lbdds

         rod(ixnotot) = rodad + rodds
         rdd(ixnotot) = rddadmh - rddadph - rdddsmh - rdddsph + rddca -
     &                  rddtf - rddai - rddwbdf
         rbd(ixnotot) = -rbdad + rbdds

         rv(ixnotot)  = rvai + rvwbdf - rvwbad
      end if

      return


      end


      subroutine tridag (a, b, c, r, u, n)

** SUBROUTINE:
**    tridag.for - solves an equation A.u = r where A is a tridiagonal
**                 matrix and u and r are vectors (Numerical recipes
**                 in Fortran, 1992); this algoritm is used to solve
**                 the mass conservation equation for the water layer
**                 of the TOXSWA program
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The equation to be solved is A.u = r;
**    A is the tridiagonal matrix of which only the nonzero elements are
**    stored as three vectors a(1:n), b(1:n) and c(1:n) representing
**    respectively the band under the diagonal, the diagonal itself and
**    the band above the diagonal. u and r are vectors of length n. So,
**    the subroutine tridag solves for a vector u the tridiagonal
**    linear set given by the equation above. a, b, c and r are input
**    vectors and are not modified.
**    Parameter: nmax is the maximum expected value of n
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

      implicit none
      include 'param.for'
C                              local variables
      integer    n, j, nmax
      real       a(n), b(n), c(n), r(n), u(n)
      parameter  (nmax = mxnxnotot)
      real       bet, gam(nmax)
C                              one vector of workspace, gam is needed

      if (abs(b(1)).lt.1.0E-06) pause ' tridag: rewrite equations'
C                              If this happens then you should rewrite
C                              your equations as a set of order n-1, with
C                              u2 trivially eliminated.

      bet = b(1)
      u(1) = r(1)/bet

      do 10 j = 2, n
C                              decomposition and forward substitution
         gam(j) = c(j-1)/bet
         bet = b(j)-a(j)*gam(j)
         if (abs(bet).lt.1.0E-06) pause ' tridag failed'
C                              Algorithm fails, see Num. recipes
         u(j) = (r(j)-a(j)*u(j-1))/bet
   10 continue

      do 20 j = n-1, 1, -1
C                              backsubstitution
         u(j) = u(j)-gam(j+1)*u(j+1)
   20 continue

      return


      end


      subroutine wlmb

** SUBROUTINE:
**    wlmb.for - the mass balance is checked for the water layer of the
**               TOXSWA program
**    author   - P.I. Adriaanse
**
** DESCRIPTION:
**    All the incoming and outcoming mass terms are calculated for the
**    water layer and it is checked that no mass is lost due to errors
**    in the calculation process. The user receives an error message
**    when more than 0.1% of the originally applied mass plus the mass
**    incoming from the sediment is lost.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                              common variables
      include 'common.for'
C                              local variables
      integer    i, k, ittlo, helpi, imbout(21)
      real       sossj(mxnxnotot), cawlj(mxnxnotot), warst
      real       totrsinwb, rsinwb, totrsoutwb, totrsvol,
     &           rsvol, totrstf, rsouteb, rsoutfb,
     &           totmwlst, totmwl,
     &           cuinwb, cuoutwb, cuvol, cutf, cuouteb, cuoutfb,
     &           qumwl, qumwlprc
      logical    wantoutp1, wantoutp2

C---- Calculation of incoming substance from sediment and of
C     substance penetrating in sediment
C     (this is total P*Jwb*delx at time j and location i)
C     This takes only place in ditch, not in front and end buffer

      totrsinwb = 0.0
```

```
      totrsoutwb = 0.0
      do 10 ixnotot = nxnofb+1, nxnofb+nxnodit
         if (qseifj.ge.zero) then
            rsinwb = leplot*qseifj*cowlj(ixnotot)*delx(ixnotot) -
     &               por(1)*kdfwbmh(1)*delx(ixnotot)*pez0hw*
     &               ((cowbjkisl(ixnotot)-cowlj(ixnotot))
     &               /(0.5*delz(1)))
         else
C                           qseifj is negative
            rsinwb = leplot*qseifj*cowbjkisl(ixnotot)*delx(ixnotot) -
     &               por(1)*kdfwbmh(1)*delx(ixnotot)*pez0hw*
     &               ((cowbjkisl(ixnotot)-cowlj(ixnotot))
     &               /(0.5*delz(1)))
         end if
         totrsinwb = totrsinwb + amin1(0.0, rsinwb)
         totrsoutwb = totrsoutwb + amax1(0.0, rsinwb)
   10 continue

C---- Calculation of substance volatilisation to the air
C     (this is Ox*Jwa*delx at time j and location i)
C     This takes only place in ditch, not in front and end buffer

      totrsvol = 0.0
      do 20 ixnotot = nxnofb+1, nxnofb+nxnodit
         rsvol = -ktl*witopj(ixnotot)*delx(ixnotot)*
     &          (cowlj(ixnotot)-(coair/khe))
         totrsvol = totrsvol + amin1(0.0, rsvol)
   20 continue

C---- Calculation of transformation of substance
C     This takes only place in ditch, not in front and end buffer

      totrstf = 0.0
      do 30 ixnotot = nxnofb+1, nxnofb+nxnodit
         cawlj(ixnotot) = cowlj(ixnotot)* (1.+
     &                   (dwmp*pez0hw/warj(ixnotot))*kdmp(ixnotot) +
     &                   coss*raomss*kdomss(ixnotot)*
     &                   ((cowlj(ixnotot)/coobkomss)**(exfrss-1.)))
         totrstf = totrstf - krtfwl*cawlj(ixnotot)*
     &             warj(ixnotot)*delx(ixnotot)
   30 continue

C---- Calculation of substance flowing out of ditch
C     (this is J*A at time j and location ebt+1/2, as well as
C     at location 1/2 [negative flow!])

      sossj(nxnotot) = raomss*kdomss(nxnotot)*coobkomss*
     &                ((cowlj(nxnotot)/coobkomss)**exfrss)
      rsouteb = qvojiph*(cowlj(nxnotot)+coss*sossj(nxnotot))
      rsouteb = amax1(0.0, rsouteb)

      sossj(1) = raomss*kdomss(1)*coobkomss*
     &                ((cowlj(1)/coobkomss)**exfrss)
      rsoutfb = qvojimh*(cowlj(1)+coss*sossj(1))
      rsoutfb = amin1(0.0, rsoutfb)

C---- Initial conditions for mass balance wl subsystem

      totmwlst = 0.0
      do 40 ixnotot = 1, nxnotot
         warst = wibot*wdhst(ixnotot) + (wdhst(ixnotot)**2)*sisl
         totmwlst = totmwlst + castwl(ixnotot)*warst*
     &              delx(ixnotot)
   40 continue

      if (its.eq.1) then
         cuinwb = 0.0
         cuoutwb = 0.0
         cuvol = 0.0
         cutf = 0.0
         cuouteb = 0.0
         cuoutfb = 0.0
      end if

C---- Actual total substance mass in wl subsystem
C     at end of timestep j

      totmwl = 0.0
      do 50 ixnotot = 1, nxnotot
         totmwl = totmwl + cawljp1(ixnotot)*warjp1(ixnotot)*
     &                delx(ixnotot)
   50 continue

C---- Integration with respect to time

      cuinwb = cuinwb + deltwl*totrsinwb
      cuoutwb = cuoutwb + deltwl*totrsoutwb
      cuvol = cuvol + deltwl*totrsvol
      cutf = cutf + deltwl*totrstf
      cuouteb = cuouteb + deltwl*rsouteb
      cuoutfb = cuoutfb + deltwl*rsoutfb

C---- Mass balance checked
C     (Quantity (g) to check mass balance wl subsystem)

      qumwl = totmwlst - cuinwb - cuoutwb + cuvol +
     &        cutf - cuouteb + cuoutfb - totmwl


C---- Output
C                           first message to screen if quantity to
C                           check mass balance has become too big
C                           (i.e. >0.1% of application+accumulated
C                           mass incoming from sediment)

      if ((totmwlst-cuinwb).gt.zero) then
         qumwlprc = (qumwl/(totmwlst-cuinwb))*100.
      else
         qumwlprc = 0.01
      end if

      if (its.eq.1) then
         ittlo = 0
      end if

      if (abs(qumwlprc).gt.0.1 .and. ittlo.lt.20) then
         print *,  ' Warning: '
         print *,  ' More than 0.1% mass missing in '
         print *,  ' mass balance water layer; '
         print *,  ' see message.out '
         write (uoer, 63) qumwlprc, its, its*deltwl/86400., qumwl,
     &                   (totmwlst-cuinwb)
   63 format (//,' Warning:',E12.5, '% of application plus accumulated',
     &        ' mass incoming from sediment', /, ' is missing in the',
     &        ' mass balance of the water layer', /,
     &        ' at timestep', I8, ' (', F6.2, ' day)', /,
     &        ' qumwl =', E10.3, ' g and', /,
     &        ' totmwlst-cuinwb =', E10.3, ' g', /,
     &        ' See output in wlmb.out', /)
         ittlo = ittlo+1
      end if

C                           calculation of 21 timesteps at which output
C                           is wanted
      helpi = nint(ntswl/20.)-1
      imbout(1) = 1
      do 60 i = 2, 21
         imbout(i) = (i-1)*helpi
   60 continue

C                           selection of wished timesteps
      wantoutp1 = .false.
      do 70 i = 1, 21
         if (its.eq.imbout(i)) then
            wantoutp1 = .true.
         end if
   70 continue

      wantoutp2 = .false.
      do 80 k = 1, nditout
         if (its.eq.itsout(k)) then
            wantoutp2 = .true.
         end if
   80 continue

      wantoutp = (wantoutp1 .or. wantoutp2)


      if (op_wlmb.eq.1) then
         if (its.eq.1) then
```

```
         open (unit = uomw, file = 'wlmb.out', status = 'unknown')
         write (uomw, 85) deltwl
   85 format (/, ' Output from subroutine wlmb', /,
     &        ' (timestep length deltwl = ', F8.1, ' s )')
      end if

      if (wantoutp) then
         write (uomw, 95) its, its*deltwl/86400., qumwl, qumwlprc,
     &                    totmwlst,
     &                    totmwl, abs(cuinwb), -cuoutwb, cuvol,
     &                    cutf, -cuouteb, cuoutfb
   95 format(/, /, ' timestep:', I8, '  (', F8.4, ' d)', /,
     &   ' mass balance for wl subsystem ', /,
     &   '    qumwl = ', E10.3, ' g for entire ditch', /,
     &   '    ( ', E12.5, '% of initial mass + incoming mass wb)', /,
     &   '    positive mb-terms:        negative mb-terms:', /,
     &   '    totmwlst =', E10.3, ' g, totmwl =', E10.3, ' g', /,
     &   '    cuinwb   =', E10.3, ' g, cuoutwb=', E10.3, ' g', /,
     &              24X,      '     cuvol   =', E10.3, ' g', /,
     &              24X,      '     cutf    =', E10.3, ' g', /,
     &              24X,      '     cuouteb=', E10.3, ' g', /,
     &              24X,      '     cuoutfb=', E10.3, ' g')
      end if

      if (its.eq.ntswl) then
         close (unit = uomw)
      end if
    end if

    return


    end



    subroutine wlmbnode

** SUBROUTINE:
**    wlmbnode.for - the mass balance is checked for a selected segment
**                   of the water layer of the TOXSWA program
**    author       - P.I. Adriaanse
**
** DESCRIPTION:
**    All the incoming and outcoming mass terms are calculated for one
**    segment of the water layer and it is checked that no mass is lost
**    due to errors in the calculation process. The user receives an
**    error message when more than 0.1% of the originally applied mass
**    plus the mass incoming from the sediment subsystem located under
**    the segment concerned is lost.
**    This subroutine was made to enable output of the mass balance for
**    selected nodes of the water layer (and corresponding sediment
**    subsystems).
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

c                           common variables
    include 'common.for'
c                           local variables
    integer    k, ittlo
    real       sossj(mxnxnotot), cawlj(mxnxnotot), warst
    real       rsinfn, rsinen, rsouten, rsoutfn, rsinwbn, rsoutwbn,
   &           rsvoln, rstfn,
   &           mwlstn, mwln,
   &           cuinfn(mxnxnotot), cuinen(mxnxnotot),
   &           cuouten(mxnxnotot), cuoutfn(mxnxnotot),
   &           cuinwbn(mxnxnotot), cuoutwbn(mxnxnotot),
   &           cuvoln(mxnxnotot), cutfn(mxnxnotot),
   &           qumn, qumnper, hx
    character  finame*12


    do 110 ixnotot = nxnofb+1, nxnofb+nxnodit
```

```
         finame = 'wlmb###.out'
         do 100 k = 1, nwbsy
            if (ixnotot.eq.iwbsy(k)) then
               if (op_wlmbnodenr.eq.1) then
                  uomn = 49+k
                  write (finame(5:7), '(I3.3)') ixnotot
                  open (unit = uomn, file = finame, status = 'unknown')
               end if

C---- Calculation of substance flowing into and out of selected segment
C     (this is J*A at time j at interfaces ixnotot-1/2 and ixnotot+1/2)

C                             first calculation of concentration at susp.
C                             solids
         if (ixnotot.ne.1) then
            sossj(ixnotot-1) = raomss*kdomss(ixnotot-1)*coobkomss*
     &                         ((cowlj(ixnotot-1)/coobkomss)**
     &                         exfrss)
         end if
         sossj(ixnotot) = raomss*kdomss(ixnotot)*coobkomss*
     &                    ((cowlj(ixnotot)/coobkomss)**
     &                    exfrss)
         if (ixnotot.ne.nxnotot) then
            sossj(ixnotot+1) = raomss*kdomss(ixnotot+1)*coobkomss*
     &                         ((cowlj(ixnotot+1)/coobkomss)**
     &                         exfrss)
         end if


         if (qvojimh.ge.zero) then
            if (ixnotot.eq.1) then
               rsinfn = 0.0
            else
               rsinfn = qvojimh*
     &                  (cowlj(ixnotot-1)+coss*sossj(ixnotot-1)) -
     &                  warjimh(ixnotot)*kdswcljimh(ixnotot)*(
     &                  cowlj(ixnotot)+coss*sossj(ixnotot) -
     &                  (cowlj(ixnotot-1)+coss*sossj(ixnotot-1)))/
     &                  (0.5*(delx(ixnotot)+delx(ixnotot-1)))
            end if
         else
C                             now discharge at interface ixnotot-1/2
C                             negative, so there is a front buffer
            if (ixnotot.eq.1) then
               rsinfn = qvojimh*
     &                  (cowlj(ixnotot)+coss*sossj(ixnotot))
            else
               rsinfn = qvojimh*
     &                  (cowlj(ixnotot)+coss*sossj(ixnotot)) -
     &                  warjimh(ixnotot)*kdswcljimh(ixnotot)*(
     &                  cowlj(ixnotot)+coss*sossj(ixnotot) -
     &                  (cowlj(ixnotot-1)+coss*sossj(ixnotot-1)))/
     &                  (0.5*(delx(ixnotot)+delx(ixnotot-1)))
            end if
         end if
         rsoutfn = amin1(0.0, rsinfn)
         rsinfn = amax1(0.0, rsinfn)


         if (qvojiph.ge.zero) then
            if (ixnotot.eq.nxnotot) then
               rsouten = qvojiph*
     &                   (cowlj(ixnotot)+coss*sossj(ixnotot))
            else
               rsouten = qvojiph*
     &                   (cowlj(ixnotot)+coss*sossj(ixnotot)) -
     &                   warjiph(ixnotot)*kdswcljiph(ixnotot)*(
     &                   cowlj(ixnotot+1)+coss*sossj(ixnotot+1) -
     &                   (cowlj(ixnotot)+coss*sossj(ixnotot)))/
     &                   (0.5*(delx(ixnotot+1)+delx(ixnotot)))
            end if
         else
C                             now discharge at interface ixnotot+1/2
C                             negative
            if (ixnotot.eq.nxnotot) then
               rsouten = 0.0
            else
               rsouten = qvojiph*
     &                   (cowlj(ixnotot+1)+coss*sossj(ixnotot+1))-
     &                   warjiph(ixnotot)*kdswcljiph(ixnotot)*(
```

```
     &                       cowlj(ixnotot+1)+coss*sossj(ixnotot+1) -
     &                       (cowlj(ixnotot)+coss*sossj(ixnotot)))/
     &                       (0.5*(delx(ixnotot+1)+delx(ixnotot)))
               end if
            end if
            rsinen = amin1(0.0, rsouten)
            rsouten = amax1(0.0, rsouten)


C---- Calculation of incoming substance from sediment and of substance
C     penetrating in sediment in selected segment
C     (this is P*Jwb*delx at time j and location ixnotot)

            if (ixnotot.ge.nxnofb+1 .and. ixnotot.le.nxnofb+nxnodit)
     &                                                           then
               if (qseifj.ge.zero) then
                  rsinwbn = leplot*qseifj*cowlj(ixnotot)*
     &                      delx(ixnotot)-
     &                      por(1)*kdfwbmh(1)*delx(ixnotot)*pez0hw*
     &                      ((cowbjkis1(ixnotot)-cowlj(ixnotot))
     &                      /(0.5*delz(1)))
               else
C                        qseifj is negative
                  rsinwbn = leplot*qseifj*cowbjkis1(ixnotot)*
     &                      delx(ixnotot)-
     &                      por(1)*kdfwbmh(1)*delx(ixnotot)*pez0hw*
     &                      ((cowbjkis1(ixnotot)-cowlj(ixnotot))
     &                      /(0.5*delz(1)))
               end if
               rsoutwbn = amax1(0.0, rsinwbn)
               rsinwbn = amin1(0.0, rsinwbn)
            else
               rsoutwbn = zero
               rsinwbn = zero
            end if

C---- Calculation of substance volatilisation to the air in selected
C     segment
C     (this is Ox*Jwa*delx(ixnotot) at time j and location ixnotot)

            if (ixnotot.ge.nxnofb+1 .and. ixnotot.le.nxnofb+nxnodit)
     &                                                           then
               rsvoln = -ktl*witopj(ixnotot)*delx(ixnotot)*
     &                  (cowlj(ixnotot)-(coair/khe))
               rsvoln = amin1(0.0, rsvoln)
            else
               rsvoln = zero
            end if

C---- Calculation of transformation of substance in selected segment

            if (ixnotot.ge.nxnofb+1 .and. ixnotot.le.nxnofb+nxnodit)
     &                                                           then
               cawlj(ixnotot) = cowlj(ixnotot)* (1.+
     &                          (dwmp*pez0hw/warj(ixnotot))*
     &                          kdmp(ixnotot) +
     &                          coss*raomss*kdomss(ixnotot)*
     &                          ((cowlj(ixnotot)/coobkomss)**
     &                          (exfrss-1.)))
               rstfn = - krtfwl*cawlj(ixnotot)*warj(ixnotot)
     &                 *delx(ixnotot)
            else
               rstfn = zero
            end if

C---- Initial conditions for mass balance selected segment

            warst = wibot*wdhst(ixnotot) + (wdhst(ixnotot)**2)*sisl
            mwlstn = castwl(ixnotot)*warst*delx(ixnotot)

            if (its.eq.1) then
               cuinfn(ixnotot) = 0.0
               cuinen(ixnotot) = 0.0
               cuouten(ixnotot) = 0.0
               cuoutfn(ixnotot) = 0.0
               cuinwbn(ixnotot) = 0.0
               cuoutwbn(ixnotot) = 0.0
               cuvoln(ixnotot) = 0.0
               cutfn(ixnotot) = 0.0
            end if

C---- Actual total substance in selected segment at end of timestep j

            mwln = cawljp1(ixnotot)*warjp1(ixnotot)*delx(ixnotot)


C---- Integration with respect to time

            cuinfn(ixnotot) = cuinfn(ixnotot) + deltwl*rsinfn
            cuinen(ixnotot) = cuinen(ixnotot) + deltwl*rsinen
            cuouten(ixnotot) = cuouten(ixnotot) + deltwl*rsouten
            cuoutfn(ixnotot) = cuoutfn(ixnotot) + deltwl*rsoutfn
            cuinwbn(ixnotot) = cuinwbn(ixnotot) + deltwl*rsinwbn
            cuoutwbn(ixnotot) = cuoutwbn(ixnotot) + deltwl*rsoutwbn
            cuvoln(ixnotot) = cuvoln(ixnotot) + deltwl*rsvoln
            cutfn(ixnotot) = cutfn(ixnotot) + deltwl*rstfn

C---- Mass balance checked
C     (Quantity (g) to check mass balance selected segment)

            qumn = mwlstn + cuinfn(ixnotot) - cuinen(ixnotot) -
     &             cuinwbn(ixnotot) - cuouten(ixnotot) +
     &             cuoutfn(ixnotot) - cuoutwbn(ixnotot) +
     &             cuvoln(ixnotot) + cutfn(ixnotot)
     &             - mwln

            if ((mwlstn+cuinfn(ixnotot)-cuinen(ixnotot)-
     &                     cuinwbn(ixnotot)).gt.zero) then
               qumnper = (qumn/(mwlstn+cuinfn(ixnotot)-
     &                   cuinen(ixnotot)-cuinwbn(ixnotot)))*100.
            else
               qumnper = 0.01
            end if

C---- Output
C                         output in message.out if error

            if (its.eq.1) then
               ittlo = 0
            end if

            if (abs(qumnper).gt.0.1 .and. ittlo.lt.20) then
               write (uoer, 63) qumnper, ixnotot, its,
     &                          its*deltwl/86400.,
     &                          qumn, (mwlstn+cuinfn(ixnotot)-
     &                                 cuinen(ixnotot)-cuinwbn(ixnotot))
   63 format (//,' Warning:', E12.5, ' % of application plus incoming',
     &        ' mass ', /, ' is missing in the',
     &        ' mass balance of node ', I8, 'of the water layer', /,
     &        ' at timestep', I8, ' (', F6.2, ' day)', /,
     &        ' qumn =', E10.3, ' g and', /,
     &        ' (mwlstn+cuinfn-cuinen-cuinwbn) =', E10.3, ' g', /,
     &        ' See output in wlmb###.out', /)
               ittlo = ittlo+1
            end if


C                         output in file wlmb###.out
            if (op_wlmbnodenr.eq.1) then
               if (its.eq.1) then
                  write (uomn, 60) deltwl
   60 format (/, ' Output from subroutine wlmb for selected node', /,
     &        ' per running meter ditch', /,
     &        ' (timestep length deltwl = ', F8.1, ' s )')
               end if

               if (wantoutp) then
                  hx = delx(ixnotot)
                  write (uomn, 69) its, its*deltwl/86400.,
     &                             ixnotot, qumn/hx,
     &                             qumnper, mwlstn/hx, mwln/hx,
     &                             abs(cuinfn(ixnotot))/hx,
     &                             -cuouten(ixnotot)/hx,
     &                             abs(cuinen(ixnotot))/hx,
     &                             cuoutfn(ixnotot)/hx,
     &                             abs(cuinwbn(ixnotot))/hx,
     &                             -cuoutwbn(ixnotot)/hx,
     &                             cuvoln(ixnotot)/hx,
     &                             cutfn(ixnotot)/hx
   69 format(/, /, ' timestep:', I8, '  (', F8.4, ' d)', /,
     &   '  mass balance for wl subsystem per m'' at node', I8, /,
```

```
     &    '     qumwn = ', E10.3, ' g per m''', /,
     &    '     ( ', E12.5, '% of initial mass + incoming mass wb ',
     &        'and neighbouring nodes)', /,
     & '    positive mb-terms:          negative mb-terms:', /,
     & '    mwlstn   =', E10.3, ' g/m'', mwln    =', E10.3, ' g/m''', /,
     & '    cuinfn   =', E10.3, ' g/m'', cuouten =', E10.3, ' g/m''', /,
     & '    cuinen   =', E10.3, ' g/m'', cuoutfn =', E10.3, ' g/m''', /,
     & '    cuinwbn  =', E10.3, ' g/m'', cuoutwbn=', E10.3, ' g/m''', /,
     &                  24X,    '       cuvoln  =', E10.3, ' g/m''', /,
     &                  24X,    '       cutfn   =', E10.3, ' g/m''')
              end if

              if (its.eq.ntswl) then
                 close (unit = uomn)
              end if
           end if

        end if
  100   continue
  110 continue


      return


      end


      subroutine wbso

** SUBROUTINE:
**    wbso.for - solves the mass conservation equation for the sediment
**               layer of the TOXSWA program
**    author   - P.I. Adriaanse
**
** DESCRIPTION:
**    The mass conservation equation for the sediment subsystem is
**    solved by solving the matrix equation, containing two tridiagonal
**    matrices describing the pesticide concentration in the liquid
**    phase of the sediment (Eq. (6.39) in SC-DLO report 90). The
**    equation can be solved in an implicit or in an explicit way and
**    due to the sorption description according to the Freundlich
**    equation an iteration takes place to calculate the final
**    pesticide concentration.
**    The following subroutines are called:
**    # wbsods, in which:
**     - the (physical) dispersion coefficient is corrected for the
**       numerical dispersion;
**    # wbsofd, in which:
**     - it is prevented that the calculated dispersion would annulate
**       the advection flux in the sediment;
**    # wbsoem, in which:
**     - the elements of the tridiagonal matrices are defined, which are
**       used in the subroutines wbsoub, wbsolb and wbsocr;
**    # wbsoub, in which:
**     - the lod, ldd etc terms of the matrices for the upper boundary of
**       the sediment layer are composed;
**    # wbsolb, in which:
**     - the lod, ldd etc terms of the matrices for the lower boundary of
**       the sediment layer are composed;
**    # wbsocr, in which:
**     - the lod, ldd etc terms of the core parts of the matrices for the
**       sediment layer are composed;
**    # btridag, in which:
**     - the tridiagonal matrix for the sediment layer is inverted.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                             common variables
      include 'common.for'
C                             local variables
      character   finame*12
      logical     loghelp, convergency_OK
      integer     ittlo, poscounter, k, kk, kn
      real        delztop, cawblintop, cowblintop, sowblintop,
     &            cawbpertop, cowbpertop, sowbpertop

C                             determination of first node number in ditch
      if (nxnofb.eq.0) then
         noldit = 1
      else
         noldit = nxnofb+1
      end if

C                             calculate dispersion: Ecalculation =
C                             Ephysical - Enumerical
      do 20 kznotot = 1, nznotot
         call wbsods
   20 continue

C---- Test condition dispersion would annulate advection

      do 30 kznotot = 1, nznotot
         call wbsofd
   30 continue

      if (its.eq.1) then
         poscounter = 1
      end if
C                              poscounter counts the number of times that
C                              the positivity conditions are not
C                              fulfilled

      ittlo = 1

C---- Loop while no final value of cowbjp1  (So start of iteration
C     procedure due to sorption according to Freundlich equation to
C     find cb at time j+1)
   40 continue


      do 50 kznotot = 1, nznotot

C---- Make estimation for clb at time j+1 to calculate ldd term

         if (ittlo.eq.1) then
            oldcowbjp1(kznotot) = cowbj(kznotot)
         else
            oldcowbjp1(kznotot) = cowbjp1(kznotot)
         end if


C---- Compose left-hand and right-hand matrices

C                             prevent undefined exponentiation in lddca
C                             and rddca terms
         if (cowbj(kznotot).le.zero) then
            cowbj(kznotot) = tiny
         end if
         if (oldcowbjp1(kznotot).le.zero) then
            oldcowbjp1(kznotot) = tiny
         end if

C                             definition separate terms of elements
C                             matrices
         call wbsoem
         if (kznotot.eq.1) then

C                             compose elements matrices at upper boundary
            call wbsoub
         end if

C                             no end buffer defined
         if (nznoebb.eq.0 .and. kznotot.ne.1) then
            if (kznotot.lt.nznowb) then
               call wbsocr
            else
               if (kznotot.eq.nznowb) then
                  call wbsolb
               end if
            end if
```

```fortran
      end if
C                        end buffer defined
      if (nznoebb.ne.0 .and. kznotot.ne.1) then
         if (kznotot.gt.nznowb) then
C                        compose elements matrices in end buffer wb
            call wbsolb
         else
C                        compose core matrices (wb excluded upper
C                        boundary and end buffer)
            call wbsocr
         end if
      end if
C                        complete matrices have now been composed

C---- Output positivity conditions for first timestep

      if (op_icwb.eq.1) then
         if (its.eq.1 .and. ixnotot.eq.noldit .and. kznotot.eq.1
     &      .and. ittlo.eq.1) then
         write (uowb, 41)
   41 format (//, ' Results from subroutine wbso')
         write (uowb, 42)
   42 format (/, ' positivity conditions are: blbd =< 0    brbd >= 0',
     &        /, '                              bldd > 0     brdd > 0',
     &        /, '                              blod =< 0    brod >= 0')
      end if
      if (its.eq.1  .and. ixnotot.eq.noldit .and. ittlo.eq.1) then
         write (uowb, 43)
   43 format (/, ' blbd    brbd    timestep   nodenumber', /,
     &           ' bldd    brdd', /,
     &           ' blod    brod', /,
     &           '     brv')
         write (uowb, *) blbd(kznotot), brbd(kznotot), its, kznotot
         write (uowb, *) bldd(kznotot), brdd(kznotot)
         write (uowb, *) blod(kznotot), brod(kznotot)
         write (uowb, *) brv(kznotot)
      end if


      if (ixnotot.eq.nxnofb+nxnodit) then
         close (unit = uowb)
      end if
   end if


C---- Test positivity conditions

      loghelp = (blbd(kznotot).gt.zero .or. bldd(kznotot).lt.zero
     &          .or. blod(kznotot).gt.zero .or. brbd(kznotot).lt.zero
     &          .or. brdd(kznotot).lt.zero .or. brod(kznotot).lt.zero)
      if (loghelp) then

         poscounter = poscounter+1
         if (poscounter.eq.50) then
            print*, ' Error:'
            print*, ' Positivity conditions wb not fulfilled;'
            print*, ' see message.out'
            print*, ' Program stops'
            stop
         end if

         write (uoer, 44)
   44 format (//, ' Message from subroutine wbso:')
         write (uoer, 45)
   45 format (/, ' Positivity conditions are: blbd =< 0    brbd >= 0',
     &        /, '                              bldd > 0     brdd > 0',
     &        /, '                              blod =< 0    brod >= 0')
         write (uoer, 46)
   46 format (/, ' blbd   brbd   timestep  nodewl  nodewb', /,
     &           ' bldd   brdd', /,
     &           ' blod   brod', /,
     &           '     brv')
      write (uoer, *) blbd(kznotot), brbd(kznotot), its,
     &                ixnotot, kznotot
      write (uoer, *) bldd(kznotot), brdd(kznotot)
      write (uoer, *) blod(kznotot), brod(kznotot)
      write (uoer, *) brv(kznotot)
      write (uoer, 47)
   47 format (/, ' So positivity conditions have not been fulfilled',
     &        /, ' are solutions given and are these OK ?')
      end if

   50 continue

C---- Split solution of matrix equations in solution with explicit
C     calculation scheme and with implicit calculation scheme

      if (abs(thetawb-1.).lt.1.0E-06) then
C                        thetawb = 1;
C                        no inversion of left-hand matrix needed
C                        to solve clbj+1,k;
C                        terms blod and blbd are zero in this case
         do 60 kznotot = 1, nznotot
            if (kznotot.eq.1) then
               cowbjp1(kznotot) = (1/bldd(kznotot))*(
     &                            brdd(kznotot)*cowbj(kznotot)+
     &                            brbd(kznotot)*cowbj(kznotot+1)+
     &                            brv(kznotot))
            else
               if (kznotot.eq.nznotot) then
                  cowbjp1(kznotot) = (1/bldd(kznotot))*(
     &                               brod(kznotot)*cowbj(kznotot-1)+
     &                               brdd(kznotot)*cowbj(kznotot)+
     &                               brv(kznotot))
               else
                  cowbjp1(kznotot) = (1/bldd(kznotot))*(
     &                               brod(kznotot)*cowbj(kznotot-1)+
     &                               brdd(kznotot)*cowbj(kznotot)+
     &                               brbd(kznotot)*cowbj(kznotot+1)+
     &                               brv(kznotot))
               end if
            end if
   60    continue
      else
         if (thetawb.ge.zero .and. abs(thetawb-1.).gt.1.0E-06) then
C                        thetawb is smaller than 1 and equal or
C                        bigger than 0;
C                        inversion of left-hand tridiagonal matrix
C                        needed
            do 70 kznotot = 1, nznotot
C                        compose right-hand of matrix equation
               if (kznotot.eq.1) then
                  brh(kznotot) = brdd(1)*cowbj(kznotot)+
     &                           brbd(1)*cowbj(kznotot+1)+brv(1)
               else
                  if (kznotot.eq.nznotot) then
                     brh(kznotot) = brod(nznotot)*cowbj(kznotot-1)+
     &                              brdd(nznotot)*cowbj(kznotot)+
     &                              brv(nznotot)
                  else
                     brh(kznotot) = brod(kznotot)*cowbj(kznotot-1)+
     &                              brdd(kznotot)*cowbj(kznotot)+
     &                              brbd(kznotot)*cowbj(kznotot+1)+
     &                              brv(kznotot)
                  end if
               end if
   70       continue
            call btridag (blod, bldd, blbd, brh, cowbjp1, nznotot)
         end if
      end if

C---  First iteration due to sorption according to Freundlich equation
C     has been done now, so first value of vector cowbjp1(kznotot) has
C     been found

C                        test whether more iterations are needed
      kznotot= 1
      convergency_OK = .true.
   80 if (convergency_OK .and. (kznotot.le.nznotot)) then
         convergency_OK = (abs(cowbjp1(kznotot)-oldcowbjp1(kznotot)).lt.
     &                     abs(0.00001*cowbjp1(kznotot)))
         kznotot = kznotot+1
         go to 80
      end if

      if (.not. convergency_OK) then
C                        give error message
         if (ittlo.eq.50) then
```

```
          write (uoer, 81)
   81 format (//, ' Message from subroutine wbso:')
          write (uoer, 82) ittlo, kznotot, its, ixnotot
   82 format (/, ' More than', /,
     &        I6, ' iteration loops will be needed to calculate clbj+1',
     &       /, ' for gridpoint:', I6, ' at timestep:', I8,
     &        /, ' in the sediment subsystem',
     &       /, ' under node waterlayer:', I8)
          write (uoer, 83) oldcowbjp1(kznotot), cowbjp1(kznotot)
   83 format (/, ' Iteration values oldcowbjp1 and cowbjp1 are:', E10.4,
     &        2X, E10.4, /, ' Never ending loop !')
           print*, ' Error:'
           print*, ' Positivity conditions wb not fulfilled;'
           print*, ' see message.out'
           print*, ' Program stops'
          stop
        end if
      end if

      if (.not. convergency_OK) then
        ittlo = ittlo+1
        go to 40
      end if
C---- End of loop while no final value of cowbjp1

C---- Now final value of clb at time j+1, cowbjp1 found

C---- Calculate Xb, and cb* at time j+1 for each node

      do 90 kznotot = 1, nznotot

        if (cowbjp1(kznotot).lt.zero) then
          write (uoer, 86)
   86 format (//, ' Message from subroutine wbso:')
          write (uoer, 87) its, kznotot, ixnotot, cowbjp1(kznotot)
   87 format (/, 'Negative concentration in sediment !', /,
     &        ' at timestep:', I6, ' and node: ', I3, /,
     &        ', in sediment sub-system under node wl:', I6,  /,
     &        ' Concentration is:', E10.4)
          cowbjp1(kznotot) = tiny
        end if


C---- When the concentration at an individual node becomes smaller than
C     1E-25 the concentration will be set at 1E-25. (At concentrations
C     in the order of 1E-34 no convergency will be reached in the
C     iteration loop at statement label 80 in this subroutine wbso.for.
C     Also other numerical problems due to computer range limitations
C     will be prevented in this way.)
        if (cowbjp1(kznotot).lt.1.0E-25) then
          cowbjp1(kznotot) = 1.0E-25
        end if


        sowbjp1(kznotot) = kdfrwb(kznotot)*coobkomwb*
     &                     ((cowbjp1(kznotot)/coobkomwb)**exfrwb)
        cawbjp1(kznotot) = por(kznotot)*cowbjp1(kznotot) +
     &                     bdwb(kznotot)*sowbjp1(kznotot)
   90 continue


C---- Concentration in upper segment of sediment subsystem under node
C     ixnotot of water layer segment now known and set aside for
C     calculation exchange wl-wb in wlsoin.for

      cowbjp1kis1(ixnotot) = cowbjp1(1)


C---- Output
C     Output of calculated concentrations; lineic mass and percentages
C     of mass total present under selected nodes wl

      finame = 'wbsco###.out'

      do 105 k = 1, nwbsy
        if (ixnotot.eq.iwbsy(k)) then
          if (op_wbsconodenr.eq.1) then
            uocb = 59 + k
            write (finame(6:8), '(I3.3)') ixnotot
            open (unit = uocb, file = finame, status = 'unknown')
            if (its.eq.1) then
              write (uocb, 104) iwbsy(k), xcd(iwbsy(k))
  104 format (/, ' Output from subroutine wbso:', /,
     &        ' Output for sediment subsystem under node', I6,
     &        ' of water layer', /, ' (at', F8.1, ' m in ditch)')
            end if
          end if
        end if
  105 continue

        do 140 kn = 1, nwbsy
          if (ixnotot.eq.iwbsy(kn) .and. op_wbsconodenr.eq.1) then
            do 130 kk = 1, 9
              if (its.eq.itsout(kk)) then
                write (uocb, 110) its, itsout(kk)*deltwb/86400.
  110 format (/,
     &        ' concentrations, lineic mass (mass per running metre',
     &        ' ditch) and', /, ' percentages for all nodes at',
     &        ' timestep:', I6, ' (', F8.4, ' day after application)',
     &        / , ' concentrations', /, ' nodenr      total        ',
     &        '    liquid phase sediment    at solid phase sediment',
     &        /, '              cb*                  clb              ',
     &        '        Xb', /, '                  g.m-3        ',
     &        '      g.m-3                   g.g-1', /,
     &        ' lineic mass (g.m''-1)', /, ' percentage of total',
     &        ' present in sediment (%)')
                do 120 k= 1, nznotot
                  write (uocb, 111) k, cawbjp1(k), cowbjp1(k),
     &                             sowbjp1(k),
     &                             cawbjp1(k)*pe(k)*delz(k),
     &                             por(k)*cowbjp1(k)*pe(k)*delz(k),
     &                             bdwb(k)*sowbjp1(k)*pe(k)*delz(k),
     &                             cawbjp1(k)/cawbjp1(k)*100.,
     &                            por(k)*cowbjp1(k)/cawbjp1(k)*100.,
     &                            bdwb(k)*sowbjp1(k)/cawbjp1(k)*100.
  111 format (I5, 4X, E12.4, 4X, E12.4, 13X, E12.4,
     &        /, 11X, E12.4, 4X, E12.4, 13X, E12.4,
     &        /, 13X, F6.2, 10X, F6.2, 19X, F6.2)
  120           continue


C                             output of lineic mass and percentage of
C                             mass total present in top layer sediment
                delztop = 0.
                cawblintop = 0.
                cowblintop = 0.
                sowblintop = 0.
                cawbpertop = 0.
                cowbpertop = 0.
                sowbpertop = 0.
                do 125 k = 1, ktop
                  delztop = delztop + delz(k)
                  cawblintop = cawblintop + cawbjp1(k)*pe(k)*delz(k)
                  cowblintop = cowblintop + por(k)*cowbjp1(k)*pe(k)*
     &                         delz(k)
                  sowblintop = sowblintop + bdwb(k)*sowbjp1(k)*
     &                         pe(k)*delz(k)
                  cawbpertop = cawbpertop + cawbjp1(k)
                  cowbpertop = cowbpertop + por(k)*cowbjp1(k)
                  sowbpertop = sowbpertop + bdwb(k)*sowbjp1(k)
  125           continue
                write (uocb, 126) delztop*1000., its,
     &                          cawblintop, cowblintop, sowblintop,
     &                          cawbpertop/cawbpertop*100.,
     &                          cowbpertop/cawbpertop*100.,
     &                          sowbpertop/cawbpertop*100.
  126 format (/, ' Lineic mass (g.m''-1) of toplayer sediment (', F7.2,
     &      ' mm thick)', /, ' percentage of total present in toplayer',
     &        /, ' timestep    total              liquid phase sediment',
     &        '   at solid phase sediment', /,
     &        I5, 4X, E12.4, 4X, E12.4, 13X, E12.4,
     &        /, 13X, F6.2, 10X, F6.2, 19X, F6.2)
              end if
  130       continue

            if (ixnotot.eq.nxnofb+nxnodit .and. its.eq.ntswl) then
              close (unit = uocb)
            end if

          end if
```

```fortran
  140 continue

      return

      end


      subroutine wbsods

** SUBROUTINE:
**    wbsods.for - the physical dispersion coefficient in the sediment
**                 layer of the TOXSWA program is corrected for the
**                 numerical dispersion
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The physical dispersion coefficient for the sediment layer is
**    corrected for the numerical dispersion. The calculation is done at
**    segment boundaries and at time j+1/2; the calculation dispersion
**    equals the physical dispersion minus the numerical dispersion.
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

c                             common variables
      include 'common.for'
c                             local variables



c                             first calculate clb at time j for
c                             kznotot-1/2 and kznotot+1/2
      if (kznotot.eq.1) then
         cowbjkmh(kznotot) = cowbj(1)
         cowbjkph(kznotot) = (1.-nwbph(kznotot))*cowbj(kznotot) +
     &                        nwbph(kznotot)*cowbj(kznotot+1)
      else
         if (kznotot.eq.nznotot) then
            cowbjkmh(kznotot) = (1.-nwbmh(kznotot))*cowbj(kznotot-1) +
     &                           nwbmh(kznotot)*cowbj(kznotot)
            cowbjkph(kznotot) = cowbj(nznotot)
         else
            cowbjkmh(kznotot) = (1.-nwbmh(kznotot))*cowbj(kznotot-1) +
     &                            nwbmh(kznotot)*cowbj(kznotot)
            cowbjkph(kznotot) = (1.-nwbph(kznotot))*cowbj(kznotot) +
     &                           nwbph(kznotot)*cowbj(kznotot+1)
         end if
      end if

c                             prevent undefined exponentiation in
c                             calculation of Enum
      if (cowbjkmh(kznotot).le.zero) then
         cowbjkmh(kznotot) = tiny
      end if
      if (cowbjkph(kznotot).le.zero) then
         cowbjkph(kznotot) = tiny
      end if


c---- Calculate physical dispersion coefficient Elb,kznotot-1/2 and
c     Elb,kznotot+1/2 in sediment included the buffer at time j+1/2

c                             in sediment included the buffer
      kdswbjkmh(kznotot) = ldis*abs((leplot*qseifj)/
     &                     (pemh(kznotot)*pormh(kznotot)))
      kdswbjkph(kznotot) = ldis*abs((leplot*qseifj)/
     &                     (peph(kznotot)*porph(kznotot)))


c--- numerical dispersion has been calculated at time j instead of at
c    time j+1/2, because clb at time j+1/2 is not yet known
c                             no dispersion term Ekznotot-1/2 exist for
c                             kznotot=1 and no dispersion term
c                             Ekznotot+1/2 exist for kznotot=nznotot
c                             (they do not appear in the two tridiagonal
c                             matrices)
      if (kznotot.gt.1) then
         kdsbnujkmh(kznotot) = (leplot*qseifj)/(2.*pemh(kznotot)) *
     &                         ((1.-2.*nwbmh(kznotot))*
     &                         (zcd(kznotot)-zcd(kznotot-1)) + deltwb*
     &                         (1.-2.*thetawb) * (leplot*qseifj)/
     &                         (pemh(kznotot)*(pormh(kznotot)+
     &                        bdwbmh(kznotot)*kdfrwbmh(kznotot)*exfrwb*
     &                         (cowbjkmh(kznotot)/coobkomwb)**
     &                                                  (exfrwb-1.))))
      end if
      if (kznotot.lt.nznotot) then
         kdsbnujkph(kznotot) = (leplot*qseifj)/(2.*peph(kznotot)) *
     &                         ((1.-2.*nwbph(kznotot))*(zcd(kznotot+1)
     &                        -zcd(kznotot)) + deltwb*(1.-2.*thetawb) *
     &                         (leplot*qseifj)/
     &                         (peph(kznotot)*(porph(kznotot)+
     &                        bdwbph(kznotot)*kdfrwbph(kznotot)*exfrwb*
     &                         (cowbjkph(kznotot)/coobkomwb)**
     &                                                  (exfrwb-1.))))
      end if

      if (kznotot.gt.1) then
         kdsbcljkmh(kznotot) = kdswbjkmh(kznotot) - kdsbnujkmh(kznotot)
      end if
      if (kznotot.lt.nznotot) then
         kdsbcljkph(kznotot) = kdswbjkph(kznotot) - kdsbnujkph(kznotot)
      end if

      if (op_icwb.eq.1) then
         if (its.eq.1 .and. ixnotot.eq.noldit .and. kznotot.eq.1) then
            write (uowb, 20)
   20 format (/, ' Some calculation results from subroutine wbsods:', /,
     &        ' physical, numerical and calculation dispersion for',
     &        ' first timestep and', /, ' first node sediment under',
     &        ' first node water layer', /,
     &  '  kdswbjph(k)  kdsbnujkph(k)   kdsbcljkph(k)  (m2.(day)-1)')
            write (uowb, *) 86400.*kdswbjkph(1), 86400.*kdsbnujkph(1),
     &                     86400.*kdsbcljkph(1)
         end if
      end if


      return


      end


      subroutine wbsofd

** SUBROUTINE:
**    wbsofd.for - prevents that calculated dispersion would cancel out
**                 advection flux (see section 4.2 of SC-DLO report 90)
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    In the sediment very sharp concentration gradients may occur. The
**    dispersion flux, calculated with the aid of this gradient, may
**    cancel out the advection flux. However, in reality this is
**    impossible (dispersion is caused by advection). Therefore, in this
**    subroutine it is prevented that the dispersion flux can cancel out
**    the advection flux. (See section 4.2 of SC-DLO report 90.)
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**
```

```
c                             common variables
      include 'common.for'
c                             local variables
      integer   icountersofd
      logical   t1, t2, t12, t3, good

C---- The condition that the calculated dispersion flux should not
c     cancel out or exceed the advection flux will be implemented at
c     time j (and not j+1/2 as clb at time j+1 is not yet known)

c                             for the flux across the interface
c                             kznotot-1/2

      sofdfactormh(kznotot) = 1.
c                             first determine whether situation 2 plus
c                             condition, Eq. (4.23) occurs (see Fig. 11
c                             of SC-DLO report 90)
      if (kznotot.gt.1) then
         t1 = qseifj.gt.zero
         t2 = ((cowbj(kznotot)-cowbj(kznotot-1))/
     &            (zcd(kznotot)-zcd(kznotot-1))).gt.zero
         t3 = (cowbjkmh(kznotot) -
     &        ldis*(cowbj(kznotot)-cowbj(kznotot-1))/
     &        (zcd(kznotot)-zcd(kznotot-1))).lt.zero
         t12 = t1 .and. t2
         good = t12 .and. t3
      end if

      if ((kznotot.gt.1) .and. good) then
c                             now suppose that dispersion flux and
c                             advection flux have exactly the same
c                             magnitude but they are directed in opposite
c                             directions (so their combined effect is
c                             zero; Elb J+1/2,k-1/2 = 0 and
c                             l.qj+1/2.clb j+1/2,k-1/2 = 0; this last
c                             term will be set to zero with the aid of a
c                             factor sofdfactormh(kznotot) in lod, ldd,
c                             rod and rdd
         kdsbcljkmh(kznotot) = zero
         sofdfactormh(kznotot) = zero
      end if

c                             determine whether situation 4 plus
c                             condition, Eq. (4.24) occurs (see Fig. 11
c                             of SC-DLO report 90)
      if (kznotot.gt.1) then
         t1 = qseifj.lt.zero
         t2 = ((cowbj(kznotot)-cowbj(kznotot-1))/
     &            (zcd(kznotot)-zcd(kznotot-1))).lt.zero
         t3 = (cowbjkmh(kznotot) +
     &        ldis*(cowbj(kznotot)-cowbj(kznotot-1))/
     &        (zcd(kznotot)-zcd(kznotot-1))).lt.zero
         t12 = t1 .and. t2
         good = t12 .and. t3
      end if

      if ((kznotot.gt.1) .and. good) then
c                             Elb J+1/2,k-1/2 = 0 and
c                             l.qj+1/2.clb j+1/2,k-1/2 = 0; this last
c                             term will be set to zero with the aid of a
c                             factor sofdfactormh(kznotot) in lod, ldd,
c                             rod and rdd
         kdsbcljkmh(kznotot) = zero
         sofdfactormh(kznotot) = zero
      end if

c                             for the flux across the interface
c                             kznotot+1/2

      sofdfactorph(kznotot) = 1.
c                             first determine whether situation 2 plus
c                             condition, Eq. (4.23) occurs
      if (kznotot.lt.nznotot) then
         t1 = qseifj.gt.zero
         t2 = ((cowbj(kznotot+1)-cowbj(kznotot))/
     &            (zcd(kznotot+1)-zcd(kznotot))).gt.zero
         t3 = (cowbjkph(kznotot) -
     &        ldis*(cowbj(kznotot+1)-cowbj(kznotot))/
     &        (zcd(kznotot+1)-zcd(kznotot))).lt.zero
         t12 = t1 .and. t2
         good = t12 .and. t3
      end if

      if ((kznotot.lt.nznotot) .and. good) then
c                             Elb J+1/2,k+1/2 = 0 and
c                             l.qj+1/2.clb j+1/2,k+1/2 = 0; this last
c                             term will be set to zero with the aid of a
c                             factor sofdfactorph(kznotot) in ldd, lbd,
c                             rdd and rbd
         kdsbcljkph(kznotot) = zero
         sofdfactorph(kznotot) = zero
      end if

c                             determine whether situation 4 plus
c                             condition, Eq. (4.24) occurs
      if (kznotot.lt.nxnotot) then
         t1 = qseifj.lt.zero
         t2 = ((cowbj(kznotot+1)-cowbj(kznotot))/
     &            (zcd(kznotot+1)-zcd(kznotot))).lt.zero
         t3 = (cowbjkph(kznotot) +
     &        ldis*(cowbj(kznotot+1)-cowbj(kznotot))/
     &        (zcd(kznotot+1)-zcd(kznotot))).lt.zero
         t12 = t1 .and. t2
         good = t12 .and. t3
      end if

      if ((kznotot.lt.nznotot) .and. good) then
c                             Elb J+1/2,k+1/2 = 0 and
c                             l.qj+1/2.clb j+1/2,k+1/2 = 0; this last
c                             term will be set to zero with the aid of a
c                             factor sofdfactorph(kznotot) in ldd, lbd,
c                             rdd and rbd
         kdsbcljkph(kznotot) = zero
         sofdfactorph(kznotot) = zero
      end if


      if (its.eq.1 .and. ixnotot.eq.noldit .and. kznotot.eq.1) then
         icountersofd = 0
      end if
      if (sofdfactormh(kznotot).le.zero .or.
     &          sofdfactorph(kznotot).le.zero) then
         icountersofd = icountersofd + 1
         if (icountersofd.le.50) then
            write (uoer, 20) icountersofd, its, ixnotot, kznotot,
     &                       sofdfactormh(kznotot),sofdfactorph(kznotot)
20 format(//, ' Warning:', /,
     &       ' Phenomenon of dispersion flux cancel out/exceeding',
     &       /, ' advection flux would occur in sediment. (This has', /,
     &       ' been avoided with aid sofdfactors.)', /,
     &       ' Number of ocurrence of this phenomenon:', I6, /,
     &       ' Timestep:', I6, ' under water layer node:', I6, /,
     &       ' sediment node:', I6, /, ' sofdfactormh:', F6.2,
     &       ' sofdfactorph:', F6.2)
         end if
      end if


      return


      end


      subroutine wbsoem

** SUBROUTINE:
**    wbsoem.for - the elements of the tridiagonal matrices for the
**                 sediment layer of the TOXSWA program are
**                 defined here
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The elements composing the two tridiagonal matrices for the
**    sediment layer are defined here; these elements are used in the
**    subroutines wbsolb, wbsoub and wbsocr.
**
** HISTORY:
```

```
**    10 April 1996 - P.I. Adriaanse
**                     version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

c                              common variables
      include 'common.for'
c                              local variables

C---- Elements of the core and buffer matrices will be composed here


c                              the terms blod.., blddadmh, brod.. and
c                              brddadmh will not be defined for kznotot=1,
c                              as well as bldddsdfph, blbd..,
c                              brdddsdfph and brbd.. for kznotot=nznotot,
c                              because they do not appear in the two
c                              tridiagonal matrices;
c                              also bldddsdfmh and brdddsdfmh will not be
c                              defined for kznotot=1, as well as
c                              blddadph, blddca, blddtf, brddadph,
c                              brddca and brddtf for kznotot=nznotot,
c                              because these terms are modified for these
c                              nodes


      if (kznotot.gt.1) then
         blodad = ((leplot*qseifjph)/delz(kznotot)) * (1.-thetawb) *
     &            (1.-nwbmh(kznotot)) * (deltwb/pe(kznotot)) *
     &            sofdfactormh(kznotot)
         bloddsdf = (pemh(kznotot)*pormh(kznotot)*(kdsbcljkmh(kznotot)+
     &              kdfwbmh(kznotot))*(1.-thetawb)*deltwb) /
     &              (delz(kznotot)*0.5*(delz(kznotot)+delz(kznotot-1))*
     &              pe(kznotot))
      end if

      if (kznotot.lt.nznotot) then
         blddadph = ((leplot*qseifjph)/delz(kznotot)) * (1.-thetawb) *
     &              (1.-nwbph(kznotot)) * (deltwb/pe(kznotot)) *
     &              sofdfactorph(kznotot)
         bldddsdfph = (peph(kznotot)*porph(kznotot)*
     &                (kdsbcljkph(kznotot)+kdfwbph(kznotot))*
     &                (1.-thetawb)*deltwb) / (delz(kznotot)*0.5*
     &                (delz(kznotot+1)+delz(kznotot))*pe(kznotot))
      end if
      if (kznotot.gt.1) then
         blddadmh = ((leplot*qseifjph)/delz(kznotot)) * (1.-thetawb) *
     &              nwbmh(kznotot) * (deltwb/pe(kznotot)) *
     &              sofdfactormh(kznotot)
         bldddsdfmh = (pemh(kznotot)*pormh(kznotot)*
     &                (kdsbcljkmh(kznotot)+kdfwbmh(kznotot))*
     &                (1.-thetawb)*deltwb) / (delz(kznotot)*0.5*
     &                (delz(kznotot)+delz(kznotot-1))*pe(kznotot))
      end if
      if (kznotot.lt.nznowb+1) then
         blddca = por(kznotot) + bdwb(kznotot)*kdfrwb(kznotot)*
     &            ((oldcowbjp1(kznotot)/coobkomwb)**(exfrwb-1.))
         blddtf = (por(kznotot) + bdwb(kznotot)*kdfrwb(kznotot)*
     &            ((oldcowbjp1(kznotot)/coobkomwb)**(exfrwb-1.))) *
     &            krtfwb*(1.-thetawb)*deltwb
      end if

      if (kznotot.lt.nznotot) then
         blbdad = ((leplot*qseifjph)/delz(kznotot)) * (1.-thetawb) *
     &              nwbph(kznotot) * (deltwb/pe(kznotot)) *
     &              sofdfactorph(kznotot)
         blbddsdf = (peph(kznotot)*porph(kznotot)*
     &              (kdsbcljkph(kznotot)+kdfwbph(kznotot))*
     &              (1.-thetawb)*deltwb) / (delz(kznotot)*0.5*
     &              (delz(kznotot+1)+delz(kznotot))*pe(kznotot))
      end if

      if (kznotot.gt.1) then
         brodad = ((leplot*qseifjph)/delz(kznotot)) * thetawb *
     &            (1.-nwbmh(kznotot)) * (deltwb/pe(kznotot)) *
     &            sofdfactormh(kznotot)
         broddsdf = (pemh(kznotot)*pormh(kznotot)*(kdsbcljkmh(kznotot)+
     &              kdfwbmh(kznotot))*thetawb*deltwb) /
     &              (delz(kznotot)*0.5*(delz(kznotot)+delz(kznotot-1))*
     &              pe(kznotot))
      end if

      if (kznotot.lt.nznotot) then
         brddadph = ((leplot*qseifjph)/delz(kznotot)) * thetawb *
     &              (1.-nwbph(kznotot)) * (deltwb/pe(kznotot)) *
     &              sofdfactorph(kznotot)
         brdddsdfph = (peph(kznotot)*porph(kznotot)*
     &                (kdsbcljkph(kznotot)+kdfwbph(kznotot))*
     &                thetawb*deltwb) / (delz(kznotot)*0.5*
     &                (delz(kznotot+1)+delz(kznotot))*pe(kznotot))
      end if
      if (kznotot.gt.1) then
         brddadmh = ((leplot*qseifjph)/delz(kznotot)) * thetawb *
     &              nwbmh(kznotot) * (deltwb/pe(kznotot)) *
     &              sofdfactormh(kznotot)
         brdddsdfmh = (pemh(kznotot)*pormh(kznotot)*
     &                (kdsbcljkmh(kznotot)+kdfwbmh(kznotot))*
     &                thetawb*deltwb) / (delz(kznotot)*0.5*
     &                (delz(kznotot)+delz(kznotot-1))*pe(kznotot))
      end if
      if (kznotot.lt.nznowb+1) then
         brddca = por(kznotot) + bdwb(kznotot)*kdfrwb(kznotot)*
     &            ((cowbj(kznotot)/coobkomwb)**(exfrwb-1.))
         brddtf = (por(kznotot) + bdwb(kznotot)*kdfrwb(kznotot)*
     &            ((cowbj(kznotot)/coobkomwb)**(exfrwb-1.))) *
     &           krtfwb*thetawb*deltwb
      end if

      if (kznotot.lt.nznotot) then
         brbdad = ((leplot*qseifjph)/delz(kznotot)) * thetawb *
     &              nwbph(kznotot) * (deltwb/pe(kznotot)) *
     &              sofdfactorph(kznotot)
         brbddsdf = (peph(kznotot)*porph(kznotot)*
     &              (kdsbcljkph(kznotot)+kdfwbph(kznotot))*
     &              thetawb*deltwb) / (delz(kznotot)*0.5*
     &              (delz(kznotot+1)+delz(kznotot))*pe(kznotot))
      end if


      return


      end



      subroutine wbsoub

** SUBROUTINE:
**    wbsoub.for - the blod, bldd etc terms of the matrices for the
**                 upper boundary of the sediment layer of the TOXSWA
**                 program are composed here
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The first row (kznotot=1) for the lefthand and righthand
**    tridiagonal matrices will be composed for this timestep.
**    (Only the nonzero components of the left-hand and right-hand
**    tridiagonal matrices will be stored, they form two times three
**    vectors; notice that blod(1), blbd(nxnotot), brod(1) and
**    brbd(nxnotot) are undefined and not referenced in the
**    subroutines).
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                     version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

c                              common variables
      include 'common.for'
c                              local variables
```

```
      if (qseifjph.ge.zero) then

         bldd(kznotot) = blddadph + bldddsdfph +
     &                   (pemh(1)*pormh(1)*kdfwbmh(1)*(1.-thetawb)
     &                   *deltwb)
     &                   / (0.5*(delz(1)**2.)*pe(1)) +
     &                   blddca + blddtf
         blbd(kznotot) = blbdad - blbddsdf

         brdd(kznotot) = -brddadph - brdddsdfph
     &                   -(pemh(1)*pormh(1)*kdfwbmh(1)*thetawb*deltwb) /
     &                   (0.5*(delz(1)**2.)*pe(1))
     &                   + brddca - brddtf
         brbd(kznotot) = -brbdad + brbddsdf

         brv(kznotot)  = cowlj(ixnotot) *(
     &                   (leplot*qseifjph*thetawb
     &                   *deltwb)/
     &                   (delz(1)*pe(1)) +
     &                   (pemh(1)*pormh(1)*kdfwbmh(1)*
     &                   thetawb*deltwb) / (0.5*(delz(1)**2.)*pe(1))) +
     &                   cowljp1(ixnotot) *(
     &                  (leplot*qseifjph*
     &                   (1.-thetawb)*
     &                   deltwb) / (delz(1)*pe(1)) +
     &                   (pemh(1)*pormh(1)*
     &                   kdfwbmh(1)*(1.-thetawb)*deltwb) / (0.5*
     &                   (delz(1)**2.)*pe(1)))
      else

         bldd(kznotot) = blddadph + bldddsdfph +
     &                   (pemh(1)*pormh(1)*kdfwbmh(1)*(1.-thetawb)
     &                   *deltwb)
     &                   / (0.5*(delz(1)**2.)*pe(1)) +
     &                   blddca + blddtf
     &                   - (leplot*qseifjph*(1.-thetawb)*deltwb) /
     &                   (delz(1)*pe(1))
         blbd(kznotot) = blbdad - blbddsdf

         brdd(kznotot) = -brddadph - brdddsdfph
     &                   -(pemh(1)*pormh(1)*kdfwbmh(1)*thetawb*deltwb) /
     &                   (0.5*(delz(1)**2.)*pe(1))
     &                   + brddca - brddtf
     &                   + (leplot*qseifjph*thetawb*deltwb)/
     &                   (delz(1)*pe(1))
         brbd(kznotot) = -brbdad + brbddsdf

         brv(kznotot)  = cowlj(ixnotot) *((pemh(1)*pormh(1)*kdfwbmh(1)*
     &                   thetawb*deltwb) / (0.5*(delz(1)**2.)*pe(1))) +
     &                   cowljp1(ixnotot) *((pemh(1)*pormh(1)*
     &                   kdfwbmh(1)*(1.-thetawb)*deltwb) / (0.5*
     &                   (delz(1)**2.)*pe(1)))
      end if

      return


      end


      subroutine wbsolb
** SUBROUTINE:
**    wbsolb.for - the blod, bldd etc terms of the matrices for the
**                 lower boundary of the sediment subsystem of the
**                 TOXSWA program are composed here
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The matrix for the last node or end buffer will be composed for
**    this timestep. (Only the nonzero terms of the left-hand and
**    right-hand tridiagonal matrices will be stored, they form two
**    times three vectors; notice that blod(1), blbd(nxnotot), brod(1)
**    and brbd(nxnotot) are undefined and not referenced in the
**    subroutines.)
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

c                             common variables
      include 'common.for'
c                             local variables

c                             end buffer defined
      if (nznoebb.ne.0) then
         if ((kznotot.gt.nznowb) .and. (kznotot.lt.nznotot)) then
c                             nodes in end buffer composed: nodes
c                             nznowb+1 untill nznotot (last node
c                             nznotot excluded)
            blod(kznotot) = -blodad - bloddsdf
            bldd(kznotot) = blddadph + bldddsdfph - blddadmh +
     &                      bldddsdfmh + por(kznotot)
            blbd(kznotot) = blbdad - blbddsdf

            brod(kznotot) = brodad + broddsdf
            brdd(kznotot) = -brddadph - brdddsdfph + brddadmh -
     &                      brdddsdfmh + por(kznotot)
            brbd(kznotot) = -brbdad + brbddsdf
            brv(kznotot)  = zero
         end if

         if (kznotot.eq.nznotot .and. qseifjph.ge.zero) then
c                             now node nznotot with downward seepage
            blod(kznotot) = -blodad - bloddsdf
            bldd(kznotot) = (leplot*qseifjph*(1.-thetawb)*deltwb) /
     &                      (delz(kznotot)*pe(kznotot)) -
     &                      blddadmh + bldddsdfmh +
     &                      por(kznotot)

            brod(kznotot) = brodad + broddsdf
            brdd(kznotot) = - (leplot*qseifjph*thetawb*deltwb) /
     &                      (delz(kznotot)*pe(kznotot))
     &                      + brddadmh - brdddsdfmh +
     &                      por(kznotot)
            brv(kznotot)  = zero
         else
            if (kznotot.eq.nznotot .and. qseifjph.lt.zero) then
c                             now node nznotot with upward seepage
               blod(kznotot) = -blodad - bloddsdf
               bldd(kznotot) = -blddadmh + bldddsdfmh +
     &                         por(kznotot)

               brod(kznotot) = brodad + broddsdf
               brdd(kznotot) = brddadmh - brdddsdfmh +
     &                         por(kznotot)

               brv(kznotot)  = -colotj*(leplot*qseifjph*thetawb*
     &                         deltwb) / (delz(kznotot)*pe(kznotot)) -
     &                         colotjp1*(leplot*qseifjph*
     &                         (1.-thetawb)*deltwb) / (delz(kznotot)*
     &                         pe(kznotot))
            end if
         end if
      end if

c                             no end buffer defined
      if (nznoebb.eq.0) then
         if (kznotot.eq.nznotot .and. qseifjph.ge.zero) then
c                             now node nznotot with downward seepage
            blod(kznotot) = -blodad - bloddsdf
            bldd(kznotot) = (leplot*qseifjph*(1.-thetawb)*deltwb) /
     &                      (delz(kznotot)*pe(kznotot)) -
     &                      blddadmh + bldddsdfmh +
     &                      blddca + blddtf

            brod(kznotot) = brodad + broddsdf
            brdd(kznotot) = - (leplot*qseifjph*thetawb*deltwb) /
     &                      (delz(kznotot)*pe(kznotot))
     &                      + brddadmh - brdddsdfmh +
     &                      brddca - brddtf
            brv(kznotot)  = zero
         else
            if (kznotot.eq.nznotot .and. qseifjph.lt.zero) then
```

```
C                          now node nznotot with upward seepage
             blod(kznotot) = -blodad - bloddsdf
             bldd(kznotot) = -blddadmh + bldddsdfmh +
     &                        blddca + blddtf

             brod(kznotot) = brodad + broddsdf
             brdd(kznotot) = brddadmh - brdddsdfmh +
     &                        brddca - brddtf

             brv(kznotot)  = -colotj*(leplot*qseifjph*thetawb*
     &                        deltwb) / (delz(kznotot)*pe(kznotot)) -
     &                        colotjp1*(leplot*qseifjph*
     &                        (1.-thetawb)*deltwb) / (delz(kznotot)*
     &                        pe(kznotot))
            end if
         end if
      end if

      return


      end


      subroutine wbsocr
** SUBROUTINE:
**    wlsocr.for - the core parts of the matrices, consisting of the
**                 numerical equations to solve the mass conservation
**                 equation for the sediment layer of the TOXSWA
**                 program are composed here
**    author     - P.I. Adriaanse
**
** DESCRIPTION:
**    The core parts of the matrices will be composed for this timestep.
**    (Only the nonzero components of the left-hand and right-hand
**    tridiagonal matrices will be stored, they form two times three
**    vectors; notice that blod(1), blbd(nxnotot), brod(1) and
**    brbd(nxnotot) are undefined and not referenced in the
**    subroutines.)
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                          common variables
      include 'common.for'
C                          local variables


      blod(kznotot) = -blodad - bloddsdf
      bldd(kznotot) = blddadph + bldddsdfph - blddadmh + bldddsdfmh +
     &                blddca + blddtf
      blbd(kznotot) = blbdad - blbddsdf

      brod(kznotot) = brodad + broddsdf
      brdd(kznotot) = -brddadph - brdddsdfph + brddadmh - brdddsdfmh +
     &                brddca - brddtf
      brbd(kznotot) = -brbdad + brbddsdf


      brv(kznotot)  = zero

      return

      end


      subroutine btridag (a, b, c, r, u, n)
** SUBROUTINE:
**    btridag.for - version of tridag for sediment (only local parameter
**                  nmax has been changed)
**    tridag.for  - solves an equation A.u = r where A is a tridiagonal
**                  matrix and u and r are vectors (Numerical recipes
**                  in Fortran, 1992); this algoritm is used to solve
**                  the mass conservation equation for the water layer
**                  of the TOXSWA program
**    author      - P.I. Adriaanse
**
** DESCRIPTION:
**    The equation to be solved is A.u = r;
**    A is the tridiagonal matrix of which only the nonzero elements are
**    stored as three vectors a(1:n), b(1:n) and c(1:n) representing
**    respectively the band under the diagonal, the diagonal itself and
**    the band above the diagonal. u and r are vectors of length n. So,
**    the subroutine btridag solves for a vector u the tridiagonal
**    linear set given by the equation above. a, b, c and r are input
**    vectors and are not modified.
**    Parameter: nmax is the maximum expected value of n
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

      implicit none
      include 'param.for'
C                            local variables
      integer     n, j, nmax
      real        a(n), b(n), c(n), r(n), u(n)
      parameter   (nmax = mxnznotot)
      real        bet, gam(nmax)
C                            one vector of workspace, gam is needed

      if (abs(b(1)).lt.1.0E-06) pause ' tridag: rewrite equations'
C                            If this happens then you should rewrite
C                            your equations as a set of order n-1, with
C                            u2 trivially eliminated.

      bet = b(1)
      u(1) = r(1)/bet

      do 10 j = 2, n
C                            decomposition and forward substitution
         gam(j) = c(j-1)/bet
         bet = b(j)-a(j)*gam(j)
         if (abs(bet).lt.1.0E-06) pause ' tridag failed'
C                              Algorithm fails, see Num. recipes.
         u(j) = (r(j)-a(j)*u(j-1))/bet
   10 continue

      do 20 j = n-1, 1, -1
C                            backsubstitution
         u(j) = u(j)-gam(j+1)*u(j+1)
   20 continue

      return


      end


      subroutine wbmb

** SUBROUTINE:
**    wbmb.for - the mass balance is checked for the sediment layer of
**               the TOXSWA program
**    author   - P.I. Adriaanse
**
** DESCRIPTION:
**    All the incoming and outcoming mass terms are calculated for the
**    sediment layer and it is checked that no mass is lost due to
**    errors in the calculation process. The user receives an error
**    messages when more than 0.1% of the originally applied mass plus
**    the mass incoming from the water layer is lost. Calculations are
**    done per sediment subsystem first and afterwards combined to form
**    the entire sediment layer.
```

```
**
** HISTORY:
**    10 April 1996 - P.I. Adriaanse
**                    version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                            common variables
      include 'common.for'
C                            local variables
      integer    k, kk, ittlo
      real       cawbj(mxnznotot)
      real       rsinwl, rsus, rsoutwl, rsper, totrstf,
     &           totmwbst, totmwb, mwbtop, delztop,
     &           cuinwl(mxnxnotot), cuus(mxnxnotot), cuoutwl(mxnxnotot),
     &           cuper(mxnxnotot), cutf(mxnxnotot), qumwb, qumwbprc
      real       alltotmwbst, alltotmwb, allmwbtop,
     &           allqumwb, allqumwbprc,
     &           allcuinwl, allcuus, allcuoutwl, allcuper, allcutf
      character  finame*12


C---- Calculation of incoming substance from water layer and of
C     substance lost to water layer
C     (this is P*J1b at time j and k=1/2)

      if (qseifj.ge.zero) then
         rsinwl = leplot*qseifj*cowlj(ixnotot) - pemh(1)*pormh(1)*
     &            kdfwbmh(1)*((cowbj(1)-cowlj(ixnotot))/(.5*delz(1)))
      else
C                             qseifj is negative
         rsinwl = leplot*qseifj*cowbj(1) - pemh(1)*pormh(1)*
     &            kdfwbmh(1)*((cowbj(1)-cowlj(ixnotot))/(.5*delz(1)))
      end if
      rsoutwl = amin1(0.0, rsinwl)
      rsinwl = amax1(0.0, rsinwl)

C---- Calculation of incoming substance by upward seepage
C     (this is P*J1b at time j and k=nznotot+1/2)

      rsus = leplot * amin1(0.0, qseifj) * colotj

C---- Calculation of substance percolated out of wb-subsystem
C     (this is P*J1b at time j and location k=nznotot)

      rsper = amax1(qseifj, 0.0)*leplot*cowbj(nznotot)

C---- Calculation of substance lost to water layer
C     (this is P*J1b at time j and k=1/2)

      if (qseifj.ge.zero) then
         rsoutwl = leplot*qseifj*cowlj(ixnotot) - pemh(1)*pormh(1)*
     &             kdfwbmh(1)*((cowbj(1)-cowlj(ixnotot))/(.5*delz(1)))
      else
C                            qseifj is negative
         rsoutwl = leplot*qseifj*cowbj(1) - pemh(1)*pormh(1)*
     &             kdfwbmh(1)*((cowbj(1)-cowlj(ixnotot))/(.5*delz(1)))
      end if
      rsoutwl = amin1(0.0, rsoutwl)

C---- Calculation of transformation of substance

      totrstf = 0.0
      do 10 k = 1, nznowb
         cawbj(k) = por(k)*cowbj(k) + bdwb(k)*kdfrwb(k)*coobkomwb*
     &              ((cowbj(k)/coobkomwb)**exfrwb)
         totrstf = totrstf - krtfwb*cawbj(k)*pe(k)*delz(k)
   10 continue

C---- Initial conditions for mass balance wb-system

      totmwbst = 0.0
      do 20 k = 1, nznotot
         totmwbst = totmwbst + castwb(k)*pe(k)*delz(k)
   20 continue

      if (its.eq.1) then
         cuinwl(ixnotot) = 0.0
         cuus(ixnotot) = 0.0
         cuper(ixnotot) = 0.0
         cuoutwl(ixnotot) = 0.0
         cutf(ixnotot) = 0.0
      end if

C---- Actual lineic mass of substance in selected top layer

      mwbtop = 0.0
      do 30 k = 1, ktop
         mwbtop = mwbtop + cawbjp1(k)*pe(k)*delz(k)
   30 continue

C---- Actual total lineic substance mass in wb-system
C     at end of timestep j

      totmwb = 0.0
      do 40 k = 1, nznotot
         totmwb = totmwb + cawbjp1(k)*pe(k)*delz(k)
   40 continue

C---- Integration with respect to time

      cuinwl(ixnotot) = cuinwl(ixnotot) + deltwb*rsinwl
      cuus(ixnotot) = cuus(ixnotot) + deltwb*rsus
      cuper(ixnotot) = cuper(ixnotot) + deltwb*rsper
      cuoutwl(ixnotot) = cuoutwl(ixnotot) +deltwb*rsoutwl
      cutf(ixnotot) = cutf(ixnotot) +deltwb*totrstf

C---- Mass balance checked
C     (Lineic quantity (g/m') to check mass balance wb)

      qumwb = totmwbst + cuinwl(ixnotot) - cuus(ixnotot) -
     &        cuper(ixnotot) + cuoutwl(ixnotot) + cutf(ixnotot) - totmwb


C---- Mass balance for all wb-subsystems totalized of ditch
C----

C---- Initial conditions
      if (ixnotot.eq.noldit) then
         alltotmwbst = 0.0
         allmwbtop = 0.0
         alltotmwb = 0.0
      end if

C                                cu..terms accumulated over time so
C                                allcu..terms set to zero each next timestep
      if (ixnotot.eq.noldit) then
         allcuinwl = 0.0
         allcuus = 0.0
         allcuper = 0.0
         allcuoutwl = 0.0
         allcutf = 0.0
      end if

C---- Integration in direction of flow in ditch

      alltotmwbst = alltotmwbst + totmwbst*delx(ixnotot)
      allmwbtop = allmwbtop + mwbtop*delx(ixnotot)
      alltotmwb = alltotmwb + totmwb*delx(ixnotot)
      allcuinwl = allcuinwl + cuinwl(ixnotot)*delx(ixnotot)
      allcuus = allcuus + cuus(ixnotot)*delx(ixnotot)
      allcuper = allcuper + cuper(ixnotot)*delx(ixnotot)
      allcuoutwl = allcuoutwl + cuoutwl(ixnotot)*delx(ixnotot)
      allcutf = allcutf + cutf(ixnotot)*delx(ixnotot)

C---- Check of mass balance for all wb-subsystems totalized of ditch

      if (ixnotot.eq. nxnofb+nxnodit) then
         allqumwb = alltotmwbst + allcuinwl - allcuus - allcuper +
     &              allcuoutwl + allcutf - alltotmwb
      end if


C---- Output
C---- Output for selected wb-subsystems

C---- Output
```

```
C                             first message to screen if quantity to
C                             check mass balance has become too big
C                             (i.e. >0.1% of initial mass + accumulated
C                             mass incoming from water layer or by
C                             upward seepage)
      if ((totmwbst+cuinwl(ixnotot)-cuus(ixnotot)).gt.zero) then
         qumwbprc = (qumwb/(totmwbst+cuinwl(ixnotot)-cuus(ixnotot)))*
     &              100.
      else
         qumwbprc = 0.01
      end if

      if (its.eq.1 .and. ixnotot.eq.noldit) then
         ittlo = 0
      end if

      if (abs(qumwbprc).gt.0.1 .and. ittlo.lt.20) then
         print *,  ' Warning:'
         print *,  ' More than 0.1% mass missing in '
         print *,  ' mass balance of one sediment subsystem; '
         print *,  ' see message.out '

         write (uoer, 45) qumwbprc, ixnotot, its, its*deltwb/86400.,
     &                    qumwb,
     &                    (totmwbst+cuinwl(ixnotot)-cuus(ixnotot))
 45 format (//, ' Warning:', E12.5, '% of initial mass plus',
     &        ' accumulated mass incoming', /, ' from water layer',
     &        ' and by upward seepage is missing in the mass balance',
     &        /, ' of the sediment subsystem under node', I8, ' of wl',
     &        /, ' at timestep', I8, ' (', F6.2, ' day)',
     &        /, ' qumwb =', E10.3, ' g and', /,
     &        ' totmwbst+cuinwl-cuus =', E10.3, ' g', /,
     &        ' See or make output in wbmb###.out', /)
         ittlo = ittlo+1
      end if


      if (ixnotot.eq.nxnofb+nxnodit .and. its.eq.ntswl) then
         close (unit = uoer)
      end if


      delztop = 0.0
      do 50 k = 1, ktop
         delztop = delztop + delz(k)
 50 continue


      finame = 'wbmb###.out'
      do 90 k = 1, nwbsy
         if (ixnotot.eq.iwbsy(k) .and. op_wbmbnodenr.eq.1) then
            uomb = 39+k
            write (finame(5:7), '(I3.3)') ixnotot
            open (unit = uomb, file = finame, status = 'unknown')
            if (its.eq.1) then
               write (uomb, 84) deltwl
 84 format (/, ' Output mass balance for sediment subsystem', /,
     &       ' (timestep length deltwl = ', F8.1, 's )')
            end if

            if (wantoutp) then
               write (uomb, 85) its, its*deltwb/86400.,
     &                          ixnotot, qumwb, qumwbprc,
     &                          totmwbst, totmwb, cuinwl(ixnotot),
     &                          cuoutwl(ixnotot), abs(cuus(ixnotot)),
     &                          -cuper(ixnotot), cutf(ixnotot),
     &                          delztop*1000., mwbtop
 85 format(/, /, ' timestep:', I8, '  (', F8.4, ' d)', /,
     &  '  mass balance for wb-subsystem under node', I4,
     &  ' of ditch', /
     &  '    qumwb = ', E10.3, ' g/m'' ditch', /,
     & '    (',E12.5, '% of initial+incoming mass wl & upw.seepage)', /,
     &  '    positive mb-terms:        negative mb-terms:', /,
     &  '    totmwbst =', E10.3, ' g/m'', totmwb =', E10.3, ' g/m''', /,
     &  '    cuinwl   =', E10.3, ' g/m'', cuoutwl=', E10.3, ' g/m''', /,
     &  '    cuus     =', E10.3, ' g/m'', cuper  =', E10.3, ' g/m''', /,
     &  '              ', 10X,    '       cutf   =', E10.3, ' g/m''',
     &  /, /,
     &  '  mass of substance in selected top layer, thickness of',
     &  F8.2, ' mm', /,
     &  '    mwbtop   =', E10.3, ' g/m''')
            end if


            if (ixnotot.eq.nxnofb+nxnodit .and. its.eq.ntswl) then
               close (unit = uomb)
            end if

         end if
 90 continue

C---- Output for all wb-subsystems totalized of ditch

      if (op_wbmball.eq.1) then
         open (unit = uoma, file = 'wbmball.out', status = 'unknown')
      end if

      if ((alltotmwbst+allcuinwl-allcuus).gt.zero) then
         allqumwbprc = (allqumwb/(alltotmwbst+allcuinwl-allcuus))*100.
      else
         allqumwbprc = -1000.
      end if

      if (op_wbmball.eq.1) then
         if (ixnotot.eq.(nxnofb+nxnodit) .and. its.eq.1) then
            write (uoma, 94) deltwl
 94 format (/, ' Output mass balance for entire sediment layer', /,
     &        ' (timestep length deltwl = ', F8.1, ' s )')
         end if

         if (ixnotot.eq.(nxnofb+nxnodit) .and. wantoutp) then
            write (uoma, 95) its, its*deltwb/86400., allqumwb,
     &                       allqumwbprc, alltotmwbst, alltotmwb,
     &                       allcuinwl, allcuoutwl, abs(allcuus),
     &                       -allcuper, allcutf, delztop*1000.,
     &                       allmwbtop
 95 format(/, /, ' timestep:', I8, '  (', F8.4, ' d)', /,
     &  '  mass balance for all wb-subsystems of ditch', /,
     &  '     allqumwb =', E10.3, ' g', /,
     & '    (', E12.5, '% of initial+incoming mass wl & upw.seepage)', /,
     &  '     positive mb-terms:        negative mb-terms:', /,
     &  '     alltotmwbst =', E10.3, ' g, alltotmwb =', E10.3, ' g', /,
     &  '     allcuinwl   =', E10.3, ' g, allcuoutwl=', E10.3, ' g', /,
     &  '     allcuus     =', E10.3, ' g, allcuper  =', E10.3, ' g', /,
     &  '                 ', 10X,   '     allcutf   =', E10.3, ' g',
     &  /, /,
     &  '  mass of substance in selected top layer, thickness of',
     &  F8.2, ' mm', /,
     &  '     allmwbtop  =', E10.3, ' g')
         end if


         if (ixnotot.eq. nxnofb+nxnodit .and. its.eq.ntswl) then
            close (unit = uoma)
         end if
      end if

      return


      end



      subroutine dbout

** SUBROUTINE:
**    dbout.for - output of distribution of pesticide mass between
**                different compartments
**    author    - P.I. Adriaanse
**
** DESCRIPTION:
**    The total mass present in the ditch, including the sediment,
**    and per running metre ditch at a node is calculated, as well
**    as its distribution between the different compartments (dissolved,
**    adsorbed at suspended solids, solid bottom material or
**    macrophytes).
**
** HISTORY:
```

```fortran
**    10 April 1996 - P.I. Adriaanse
**                     version 1.0
**
** COPYRIGHT:
**    DLO Winand Staring Centre for Integrated Land, Soil and Water
**    Research (SC-DLO), 1996.
**

C                         common variables
      include 'common.for'
C                         local variables
      integer    i, k, uodn
      real       cawlj(mxnxnotot), sompj(mxnxnotot), sossj(mxnxnotot),
     &           cawbj(mxnznotot), sowbj(mxnznotot),
     &           lmwlca, lmwlco, lmwlmp, lmwlss, lmwbca, lmwbco, lmwbso,
     &           mwlca(mxnxnotot), mwlco(mxnxnotot), mwlmp(mxnxnotot),
     &           mwlss(mxnxnotot), mwbca(mxnxnotot), mwbco(mxnxnotot),
     &           mwbso(mxnxnotot), totmwlca, totmwlco, totmwlmp,
     &           totmwlss, totmwbca, totmwbco, totmwbso
      character  finame*12

      if (wantoutp) then

C ----First calculation of distribution of mass at time j
         do 10 i = 1, nxnotot
            sossj(i) = raomss*kdomss(i)*coobkomss*
     &                 ((cowlj(i)/coobkomss)**exfrss)
            sompj(i) = kdmp(i)*cowlj(i)
            cawlj(i) = cowlj(i) +
     &                 dwmp*pez0hw*sompj(i)/warj(i) +
     &                 coss*sossj(i)
   10    continue

         do 20 k = 1, nznotot
            sowbj(k) = kdfrwb(k)*coobkomwb*
     &                 ((cowbj(k)/coobkomwb)**exfrwb)
            cawbj(k) = por(k)*cowbj(k) +
     &                 bdwb(k)*sowbj(k)
   20    continue

C ----Calculation of lineic mass for resp. under each node wl

C                         lineic mass water layer
         lmwlca = warj(ixnotot)*cawlj(ixnotot)
         lmwlco = warj(ixnotot)*cowlj(ixnotot)
         lmwlmp = dwmp*pez0hw*sompj(ixnotot)
         lmwlss = warj(ixnotot)*coss*sossj(ixnotot)
C                         lineic mass sediment (end buffer excluded)
         lmwbca = 0.
         lmwbco = 0.
         lmwbso = 0.
         do 40 i = 1, nznowb
            lmwbca = lmwbca+cawbj(i)*pe(i)*delz(i)
            lmwbco = lmwbco+por(i)*cowbj(i)*pe(i)*delz(i)
            lmwbso = lmwbso+bdwb(i)*sowbj(i)*pe(i)*delz(i)
   40    continue

C ----Output of distribution of lineic mass (mass present per
C     running metre ditch) at a selected node of the water layer

         finame = 'db###.out'
         do 50 k = 1, nwbsy
            if (ixnotot.eq.iwbsy(k) .and. op_dbnodenr.eq.1) then
               uodn = 69+k
               write (finame(3:5), '(I3.3)') ixnotot
               open (unit = uodn, file = finame, status = 'unknown')
               if (its.eq.1) then
                  write (uodn, 45) ixnotot, xcd(ixnotot)
   45 format (/, ' Lineic mass (g/m'') in water layer',
     &        ' and in sediment', /,
     &        ' at node', I8, ' of water layer (', F8.2, ' m in ditch)',
     &        /, '          water layer                              ',
     &        ' sediment', /,
     &        ' days      total_wl  dissolved macroph.  susp.solid ',
     &        ' total_wb  dissolved solid bot. timest.')
               end if
               write (uodn, 46) its*deltwl/86400., lmwlca, lmwlco,
     &                          lmwlmp, lmwlss, lmwbca, lmwbco,
     &                          lmwbso, its
   46 format (F8.4, 1X, 4(E10.3), 2X, 3(E10.3), 1X, I6)

               if (ixnotot.eq.nxnofb+nxnodit .and. its.eq.ntswl) then
                  close (unit = uodn)
               end if

            end if
   50    continue

C ----Calculation of mass present in resp. under each section wl

C                         mass section water layer
         mwlca(ixnotot) = lmwlca*delx(ixnotot)
         mwlco(ixnotot) = lmwlco*delx(ixnotot)
         mwlmp(ixnotot) = lmwlmp*delx(ixnotot)
         mwlss(ixnotot) = lmwlss*delx(ixnotot)
C                         mass in wb subsystem under section of wl
         mwbca(ixnotot) = lmwbca*delx(ixnotot)
         mwbco(ixnotot) = lmwbco*delx(ixnotot)
         mwbso(ixnotot) = lmwbso*delx(ixnotot)

C                         initialisation total mass present in ditch
         totmwlca = 0.
         totmwlco = 0.
         totmwlmp = 0.
         totmwlss = 0.
         totmwbca = 0.
         totmwbco = 0.
         totmwbso = 0.

         if (ixnotot.eq.(nxnofb+nxnodit)) then
            do 60 i = nxnofb+1, nxnofb+nxnodit
               totmwlca = totmwlca + mwlca(i)
               totmwlco = totmwlco + mwlco(i)
               totmwlmp = totmwlmp + mwlmp(i)
               totmwlss = totmwlss + mwlss(i)
               totmwbca = totmwbca + mwbca(i)
               totmwbco = totmwbco + mwbco(i)
               totmwbso = totmwbso + mwbso(i)
   60       continue

C ----Output of distribution of mass for the entire ditch (including
C     sediment)

            if (op_dbdit.eq.1) then
               open (unit = uodd, file = 'dbdit.out',status = 'unknown')
               if (its.eq.1) then
                  write (uodd, 65)
   65 format (/, ' Mass (g) in water layer',
     &        ' and in sediment (excl. buffers)',
     &        /, '          water layer                              ',
     &        '  sediment', /,
     &        ' days      total_wl  dissolved macroph.  susp.solid ',
     &        ' total_wb  dissolved solid bot. timest.')
               end if
               write (uodd, 66) its*deltwl/86400., totmwlca, totmwlco,
     &                          totmwlmp, totmwlss, totmwbca, totmwbco,
     &                          totmwbso, its
   66 format (F8.4, 1X, 4(E10.3), 2X, 3(E10.3), 1X, I6)
            end if
         end if

      end if


      if (op_dbdit.eq.1) then
         if (ixnotot.eq.nxnofb+nxnodit .and. its.eq.ntswl) then
            close (unit = uodd)
         end if
      end if


      return


      end
```