University of Twente
Department of Applied Mathematics
Systems and Control Group

Sandra J.C. Konings

Tidal effects on cohesive sediment transport in the Western Scheldt

an identification of model parameters using
the extended Kalman filter

Getijeffecten op slibtransport in de Westerschelde

een identificatie van modelparameters met behulp van
het extended Kalman filter

Gecombineerde Stage- & Doctoraalopdracht

in opdracht van
Ministerie van Verkeer en Waterstaat
Directoraat-Generaal Rijkswaterstaat
Rijksinstituut voor Kust en Zee (RIKZ)
te Middelburg
periode 19 januari t/m 15 september 1994

Begeleiders:
Prof. dr. A. Bagchi (UT)
Dr. ir. B. de Jong (UT)
Ir. A. Langerak (RWS)
Ing. D. van Maldegem (RWS)

Afstudeercommissie tevens:
Dr. J.W. Polderman (UT)
Dr. ir. P. Löhnberg (UT)

Enschede, 16 november 1994

Voorwoord

Deze gecombineerde stage- doctoraalopdracht is uitgevoerd voor het Rijksinstituut voor Kust en Zee (RIKZ), afdeling Onderzoek en Strategie, te Middelburg. Het RIKZ is een dienst van het Ministerie van Verkeer en Waterstaat en levert adviezen en gegevens gericht op duurzaam gebruik van estuaria, kusten en zeeën en bescherming tegen overstroming door de zee. Deze opdracht is uitgevoerd in het kader van het project DYNASTAR (dynamica estuaria). Daarnaast zijn er raakvlakken met de projecten SCHOON, OOSTWEST en TROEBEL.[1]

Hierbij wil ik RIKZ-Middelburg bedanken voor de prima tijd, waarin ik naast mijn werk in de gelegenheid werd gesteld schorren[2] en slikken[3] te bekijken. Daarmee doel ik op de bezoekjes aan de Zuidgors, de slikken van Vianen en de dagexcursie naar het oostelijk deel van het Verdronken Land van Saeftinge. Verder mocht ik mee varen op een boot van de meetdienst van Hansweert tot aan Rupelmonde, alwaar sedimentmetingen werden verricht. Deze uitstapjes hebben er zeker toe bij gedragen dat ik enthousiast met deze materie aan de slag ben gegaan.

Mijn dank gaat uit naar ir. A. Langerak die mij intensief begeleid heeft en deze opdracht heeft mogelijk gemaakt en naar ing. D. van Maldegem voor het creëren van een plaatsje op zijn kamer en voor het wegwijs maken in de overstelpende hoeveelheid materie over het slib en de Westerschelde. Verder dank ik dr. ir. A. Heemink voor het geven van aanwijzingen en ideeën bij het toepassen van het extended Kalman filter, dr. ir. B. de Jong en prof. dr. A.

[1]Meer over deze projecten en over het RIKZ vindt u in appendix B.

[2]Een schorrengebied is over het algemeen een hooggelegen, sterk begroeid areaal waar afzetting van fijn materiaal plaatsvindt.

[3]Slikken zijn lager gelegen gebieden, waar door de gunstige hydraulische omstandigheden afzetting van slibrijk materiaal kan plaatsvinden.

Bagchi voor begeleiding en belangstelling vanaf de universiteit, J. Oostveen en B. Kops voor het kritisch doorlezen van dit rapport en mijn ouders voor de ondersteuning tijdens mijn hele studie.

Enschede, 16 november 1994

Sandra J.C. Konings

Samenvatting

Het verband tussen de slibconcentratie en het getij in een waterkolom in een estuarium, kan beschreven worden aan de hand van een vereenvoudigd massabalansmodel. Dit model bevat enkele onbekende parameters die geschat kunnen worden met behulp van het extended Kalmanfilter.

Bij het uitvoeren van deze schattingen voor de situatie bij Bath, in de Westerschelde, blijkt dat kennis van de systeemruis-covariantiematrix noodzakelijk is om tot goede schattingen te komen. Deze systeemruis-covariantiematrix kan op verschillende manieren worden geïdentificeerd.

In dit rapport wordt een identificatie van deze matrix met behulp van de methode van Mehra, die gebruik maakt van de autocorrelatiefunctie van het innovatie proces, vergeleken met een identificatie met behulp van de Maximum Likelihood methode. Het blijkt dat deze twee methoden ongeveer dezelfde resultaten geven, mits er met een toelaatbare beginvoorwaarde wordt gestart, maar dat de methode van Mehra minder rekentijd vergt.

Na bepaling van de systeemruis covariantiematrix met een combinatie van de twee genoemde methoden, worden de uiteindelijke schattingen verkregen van de onbekende parameters van het massabalans model. Deze schattingen zien er redelijk uit, maar leveren, middels het model, nog geen perfecte beschrijving van de slibconcentraties.

Aanbevolen wordt om het model uit te breiden met een verticale diffusiecoëfficiënt, waardoor het model meer zal overeenkomen met de werkelijkheid.

Abstract

In an estuary, tidal effects on the concentration of cohesive sediment in a column of water, may be described by a simple conservation of mass equation. The unknown parameters of this equation may be estimated by the method of extended Kalman filtering.

After estimating these parameters for the situation near Bath, in the Western Scheldt, it appeared that knowledge of the system noise covariance matrix is necessary to obtain satisfactory estimations. There are several ways to identify this system noise covariance matrix.

We compare a method suggested by Mehra, which makes use of the autocorrelation function of the innovation process, with the Maximum Likelihood method, to identify this matrix. It appears that these two methods give the same results, provided that both are started with an allowable initial value, but the Maximum Likelihood method requires more computer time than the method of Mehra.

After using both methods to identify the system noise covariance matrix, final estimates of the unknown parameters in the conservation of mass equation were obtained. Though these estimates seem to be reasonable, they do not result in a perfect description of the cohesive sediment concentration by the model.

The suggestion is to extend the model with a vertical diffusion coefficient, which may result in a more reliable model.

Contents

Voorwoord iii

Samenvatting v

Abstract vi

1 Introduction 1

2 Formulation of the problem 3
- 2.1 Introduction 3
- 2.2 Trend analysis 3
- 2.3 Available data 4
- 2.4 Tidal effects on the cohesive sediment concentration 7
- 2.5 Conclusions 9

3 Mathematical model 10
- 3.1 Introduction 10
- 3.2 Conservation of Mass 10
- 3.3 Erosion and Sedimentation 11
  - 3.3.1 Erosion 11
  - 3.3.2 Sedimentation 12
- 3.4 Description of the parameters 12
- 3.5 Conclusions 14

4 Estimation method 16
- 4.1 Introduction 16
- 4.2 The Kalman filtering method 16
- 4.3 Application of the method 17
- 4.4 Mathematical description of the method 17

4.4.1 Linear Kalman filter 17
4.4.2 Extended Kalman filter 19
4.5 Conclusions 20

5 Estimation of the unknown parameters 21
5.1 Introduction 21
5.2 Application of the Kalman filter 22
5.3 Mathematical systems 23
5.3.1 Original model 23
5.3.2 Discrete system to estimate the erosion constant M . 24
5.3.3 Discrete system to estimate the settling velocity W_s . 24
5.3.4 Discrete system to estimate the critical bed shear stress for erosion τ_{ce} 25
5.3.5 Discrete system to estimate the critical bed shear stress for sedimentation τ_{cd} 26
5.4 Estimation 27
5.4.1 Preparations 27
5.4.2 Estimation of M and W_s 28
5.4.3 Estimation of τ_{ce} and τ_{cd} 30
5.4.4 Obtained model 32
5.5 Conclusions 33

6 Extension of the model 35
6.1 Introduction 35
6.2 Adaptation of the bed shear stress expression 36
6.2.1 Behaviour of the bed shear stress τ_b 36
6.2.2 Static shift of the bed shear stress $\tau_b(k)$ 36
6.2.3 Dynamic shift of the bed shear stress $\tau_b(k)$ 38
6.3 Distinction between the ebb-stream and the flood-stream . . 39
6.4 Estimation using the extended model 41
6.4.1 Estimation of M and W_s in the cases of ebb and flood 41
6.4.2 Estimation of τ_{ce} and τ_{cd} in the cases of ebb and flood 43
6.4.3 Obtained model 44
6.5 Conclusions 47

7 Identification of the noise statistics 49
7.1 Introduction 49
7.2 The system noise covariance matrix Q 49
7.3 Two methods to identify Q 50

7.3.1 Maximum Likelihood method 50
7.3.2 Method of Mehra 53
7.4 Application of the noise identification 58
7.5 Conclusions 58

8 Parameter and system noise identification **60**
8.1 Introduction 60
8.2 Identification of the system noise covariance matrix Q 61
8.3 Estimation of the unknown parameters using identified Q . . 68
8.3.1 Estimation of M and W_s, using identified Q 68
8.3.2 Estimation of τ_{ce} and τ_{cd}, using identified Q 69
8.3.3 Obtained model using identified Q 70
8.4 Conclusions 72

9 Summary and Conclusions **76**

Bibliography **87**

List of Symbols **90**

Appendix **92**

A Estimation with identified Q **92**
A.1 Identification of the system noise covariance matrix Q 92
A.2 Estimation of τ_{ce} and τ_{cd}, using the identified Q 97

B National Institute for Coastal and Marine Management **99**
B.1 Organization 99
B.2 Projects 101
B.3 Field of activity 101

C Listings **103**

List of Figures

2.1 Some locations in the Dutch Western Scheldt 5
2.2 Turbidity T and water level H over a period of three days in November 1988 at Bath 6
2.3 Turbidity T and absolute value of the horizontal velocity V over a period of three days in November 1988 at Bath 8

3.1 Illustration of the simplified situation, looking at one column of water 11

5.1 Estimates of the erosion constant M and the settling velocity W_s, using the simplified model, over a period of three days . 29
5.2 Estimates of the critical bed shear stress for erosion τ_{ce} and the critical bed shear stress for sedimentation τ_{cd}, using the simplified model, over a period of three days 31
5.3 The model compared with the observations over a period of three days 32

6.1 The measured cohesive sediment concentration C compared with the bed shear stress τ_b (by Eqn (3.16)) over a period of three days in November 1988 at Bath 37
6.2 Estimates of the erosion constant M and the settling velocity W_s, in the cases of ebb-tide and flood-tide, using the extended model, over a period of six days 42
6.3 Estimates of the critical bed shear stress for erosion τ_{ce} and the critical bed shear stress for sedimentation τ_{cd}, in the cases of ebb-tide and flood-tide, using the extended model, over a period of six days 42
6.4 The model compared with the observations using the extended model, over a period of six days 45

6.5 The model compared with the observations, using arbitrary values for the eight unknown parameters, over a period of eighteen days 46

8.1 Results of the identification of $Q_{11}(i)$ by the method of Mehra, for $i = 1 \cdot\cdot 4$ 67

8.2 Estimates of the erosion constant M and the settling velocity W_s, in the cases of ebb-tide and flood-tide, using identified Q, over a period of eighteen days 69

8.3 Estimates of the critical bed shear stresses for erosion and sedimentation, τ_{ce} and τ_{cd}, in the cases of ebb-tide and flood-tide, using identified Q, over a period of eighteen days 71

8.4 The model compared with the observations, using identified Q, over a period of eighteen days 72

9.1 Water column divided into three parts 84

A.1 Results of the identification of $Q_{11}(i)$ by the method of Mehra, for $i = 5 \cdot\cdot 8$ 95

B.1 Survey of the division parts of RIKZ 100

List of Tables

6.1 Date and time of the spring-tide (i.e. full moon, or new moon) and the neap-tide (i.e. first quarter or last quarter), occurred in November and December 1988 38

8.1 Results of the method of Mehra, to estimate the system noise covariance matrices $Q(i)$, $i = 1 \cdot\cdot 4$ 75

A.1 Results of the method of Mehra, to estimate the system noise covariance matrices $Q(i)$, $i = 5 \cdot\cdot 8$ 98

B.1 RIKZ organization chart 100

C.1 Summary of the used FORTRAN programs 103

C.2 Summary of the used MATLAB programs 104

Chapter 1

Introduction

During the last twenty years several measurements were carried out in the Dutch Western Scheldt to gather information about the behaviour of cohesive sediment. There are at least three reasons why we want to know more about this behaviour.

One reason is the fact that cohesive sediment is a carrier of polluting substances. Therefore understanding the distribution and transport mechanisms of this suspended matter will cause insight into the rate of pollution. Another reason why we want to know more about the behaviour of the cohesive sediment, is that we want to know more about the effects on the cohesive sediment concentration of the dredging and dumping activities, which are necessary to keep the navigation channels and the harbours open. These dredging activities are needed, because the Western Scheldt is an important international shipping route. The third reason is the fact that we want to know more about the influence of the discharge of the Scheldt river on the cohesive sediment concentration in the Western Scheldt.

In this investigation we will construct a mathematical model which describes the influence of the tide on the cohesive sediment concentration at a certain location in the Western Scheldt. We start with the description of the problem in chapter 2. In chapter 3 we state the model we want to identify. The unknown parameters of this model will be estimated by the method of extended Kalman filtering. This method is described in chapter 4 and the results of the estimation procedure are given in chapter 5. Because we are not satisfied about these results, we shall examine some extensions of the model in chapter 6. The Kalman filter method does not work well. We want to improve this method by identifying the system noise covariance matrix. On behalf of this, we compare two methods to identify this matrix in chapter 7; the Maximum Likelihood method and a method suggested by Mehra. In chapter 8, we use both methods to identify the system noise covariance matrix. We use the obtained adapted Kalman filter to estimate the four unknown parameters of the model of the cohesive sediment concentration. Finally, in chapter 9, we state the conclusions and give suggestions to improve the used model.

Chapter 2

Formulation of the problem

2.1 Introduction

We want to construct a model, which describes the tidal effects on the cohesive sediment concentration in the Western Scheldt. In this chapter, we will explain what we need this model for, and we will discuss the available data we used to construct the model. Finally, we will look at some tidal effects on the cohesive sediment concentration near Bath in the Western Scheldt. Knowing more about these tidal effects and the behaviour of cohesive sediment, we are able to formulate a mathematical model in the next chapters.

2.2 Trend analysis

Sediment consists of little particles. We distinguish for instance cohesive sediment particles ($< 53\mu$m) and sand particles ($\approx 200\mu$m). In this investigation we will concentrate on cohesive sediment particles. These particles may be suspended easily, because their grain size is very small and their conductivity is high.

We want to construct a model, which describes the tidal effects on the cohesive sediment concentration. Using the constructed model, we may eliminate these tidal effects from a set of cohesive sediment concentration data of for instance twenty years. When we also eliminate the seasonal effects on this data set, the obtained set may show a certain trend of the concentration of cohesive sediment over these twenty years. With the knowledge of this trend and, for instance, the knowledge of the discharge of the Scheldt river, we are able to determine the influence of the discharge of the Scheldt river on the sediment concentration in the Western Scheldt. We may also examine whether or not, the dredging and dumping activities have any influence on the cohesive sediment concentration, using both the knowledge of a trend of the cohesive sediment concentration and the knowledge of these dredging and dumping activities.

In this investigation we will concentrate on the tidal effects on the cohesive sediment concentration. We do not concentrate on other influences as, for instance: temperature, discharge of the Scheldt river, storm, wind, waves, tide, turbulence, grade of chloride, etc. These influences are varying in time and no correlation has been found between these influences and the concentration of cohesive sediment in a column of water in an estuary[1] yet. So we do not take these influences into consideration, to keep the problem simple.

Remark: We mentioned here, that we may examine a global variation over a period of about twenty years, but of course, we may consider many other concentration variations as well,

- Variation over a tidal period.
- Variation over a period of a few days as a result of the wind activity.
- Variation within a month as a result of neap-tide and spring-tide.
- Variation within a year as a result of the variation in the seasons.
- Variation over a longer period, to notice the effects of for instance dredging. (see Maiwald and Verhagen, 1991).

2.3 Available data

To construct a model which describes the tidal effects on the cohesive sediment concentration, we need data of this concentration. During the last twenty years, a lot of measurements were carried out in the Western Scheldt to gather information about this cohesive sediment concentration. One kind of those measurements took place twice a month on several locations in the Western Scheldt. These data may tell something about the global trend during the last twenty years, but it is not possible to derive information about trends over shorter periods from it. This requires more data over one tidal period at settled locations. This is the reason why it was decided to

[1] An estuary is a body of water partially surrounded by land, with a connection to the sea, where sweet water is mixing with salt water.

take continuous measurements during three winters on three locations in the Western Scheldt. With these data we may obtain more information about the influences of tide and of, for instance storms, on the cohesive sediment concentration.

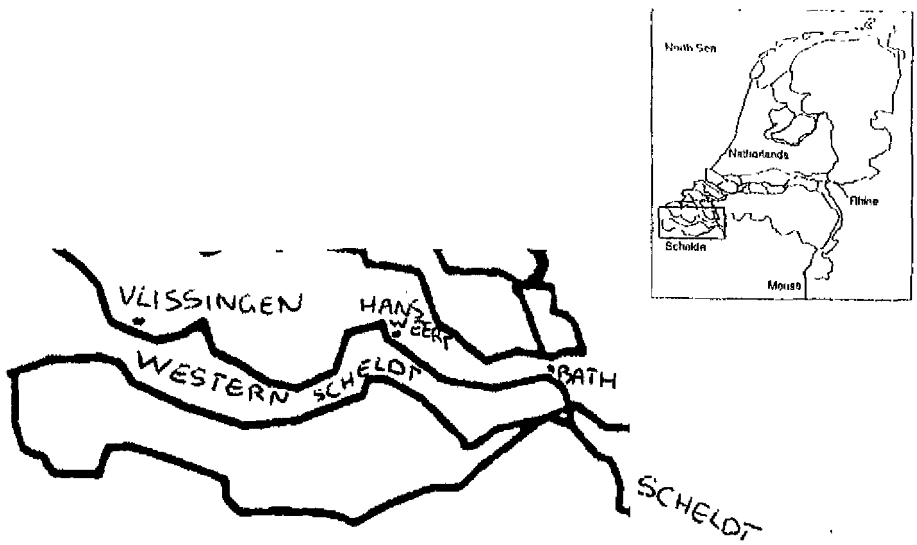


Figure 2.1: Some locations in the Dutch Western Scheldt

The continuous measurements were done with a measuring instrument called "Zeekat". This is a pontoon which may be anchored to the bottom. With this pontoon measurements may be taken on three levels in the water column, even in bad weather conditions. The "Zeekat" checks the water depth and takes water samples at 2 meters above the bottom, at $\frac{6}{10}$ of the water depth and at 1 meter below the water surface. It measures: turbidity, fluorescence, temperature, conduction and velocity (in the two horizontal x and y directions). The turbidity is measured at all three levels, the other quantities are only measured near the free surface. The "Zeekat" takes samples of the turbidity at each level for about three minutes. The turbidity data we use in this investigation are the average values of those samples, for each level.

Once a week, turbidity samples were taken to the laboratory, to determine the calibrations of the relation between turbidity and the cohesive sediment concentration for each level. We will use these calibrations to compute the cohesive sediment concentration out of the turbidity data.

The "Zeekat" was situated at "Middelgat" near Hansweert during the winter '87/'88, near Bath during the winter '88/'89 and near Vlissingen during the winter '89/'90. (locations in figure 2.1). It was decided to consider the measurements carried out near Bath first. Since at this location the influence of the Scheldt river on the cohesive sediment concentration data may be most significant.

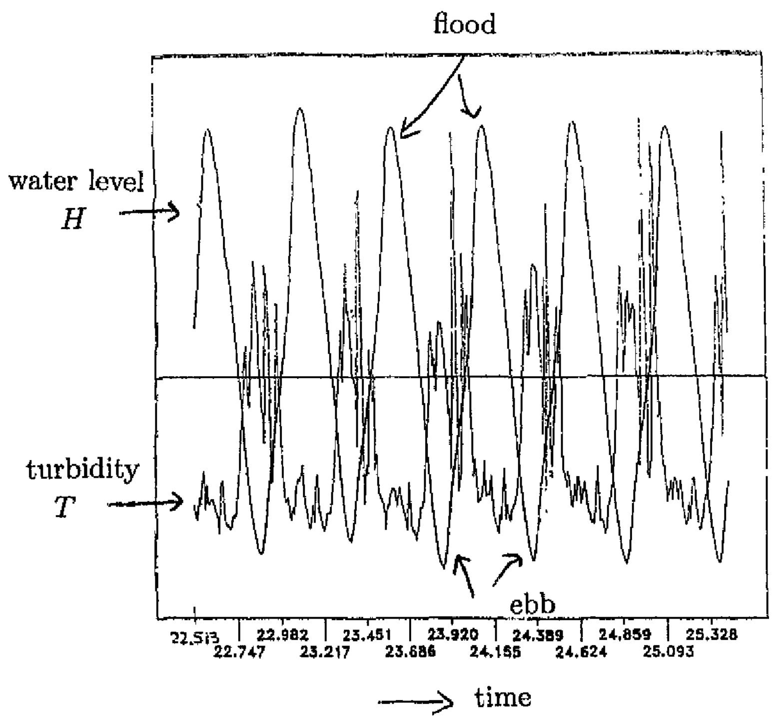


Figure 2.2: Turbidity T and water level H over a period of three days in November 1988 at Bath

where	*22.500*	*means November 22, 12:00:00 h*
	24.750	*means November 24, 18:00:00 h*

2.4 Tidal effects on the cohesive sediment concentration

In this section we examine the behaviour of cohesive sediment as a result of tide, using some conclusions of a literature study by Manni [15]. Tide may be described by the water depth H and the horizontal velocity V.

We examine the turbidity, i.e. the concentration of all sediment, including the cohesive sediment, of a few days in November 1988 at Bath.[2] We concentrate on the peaks of the signal, since we look for an explanation of those peaks. The peaks of the turbidity are shown in figure 2.2 where the turbidity T is shown together with the water level H, over a period of three days in November 1988 at Bath.[3] Note that the turbidity at ebb-tide is higher than the turbidity at flood-tide. This may be explained by the fact that during ebb-tide the water near Bath is coming from inland locations. We know for instance, that near the Dutch-Belgium border are sluices. At that location, a lot of dredging takes place, which brings about a lot of cohesive sediment into the water column. This water with much suspended particles in it, is transported to Bath during ebb-tide, which will cause the high peaks at ebb-tide. The low peaks at flood-tide may be a result of the mud flats upstream from Bath; most of the sediment is left over there, and will not any longer be into the water column near Bath.

So we distinguish between ebb-tide and flood-tide. In ebb-tide the bottom layer of the water is moving in seaward direction, and in flood-tide, it is moving in inland direction. So the velocity near the bottom at flood-tide is in opposite direction to the velocity at ebb-tide. At the turn of the tide, the value of the absolute velocity is zero. In figure 2.3 the value of the absolute velocity is shown together with the turbidity for three days in November 1988 at Bath.

Looking at figure 2.3 we note three peaks in the turbidity after the turn at ebb-tide, i.e. the low water turn (LW-turn), labeled 1,2 and 3. The first peak is caused by a small layer of fresh sediment at the bottom of the water column. This fresh sediment consists of only cohesive sediment particles,

[2]It is allowed to examine the turbidity of November only, because the shape of the figure of this turbidity does not differ from the shape of the turbidity figures of other months in 1988/1989.

[3]We will only use the data obtained at the mean water level, and not the data obtained near the surface or at the bottom of the water column. Because we are only interested in the shape of the concentration figure, and the shapes of the concentration figures of all three levels are the same.

which will erode fast. This causes a high peak in turbidity (1). The second peak is a little lower than the first one. The fresh layer of sediment is already eroded, but the velocity is increased. At this moment the velocity will be high enough to cause the bottom sediment to erode. This will cause the second peak (2). At the end of the period of flood-tide (note that the water level is increasing), the velocities become very big. This causes another part of the sediment to be eroded (3). Because a big part of the sediment has been eroded before, the third peak of turbidity will not be as high as the other peaks.

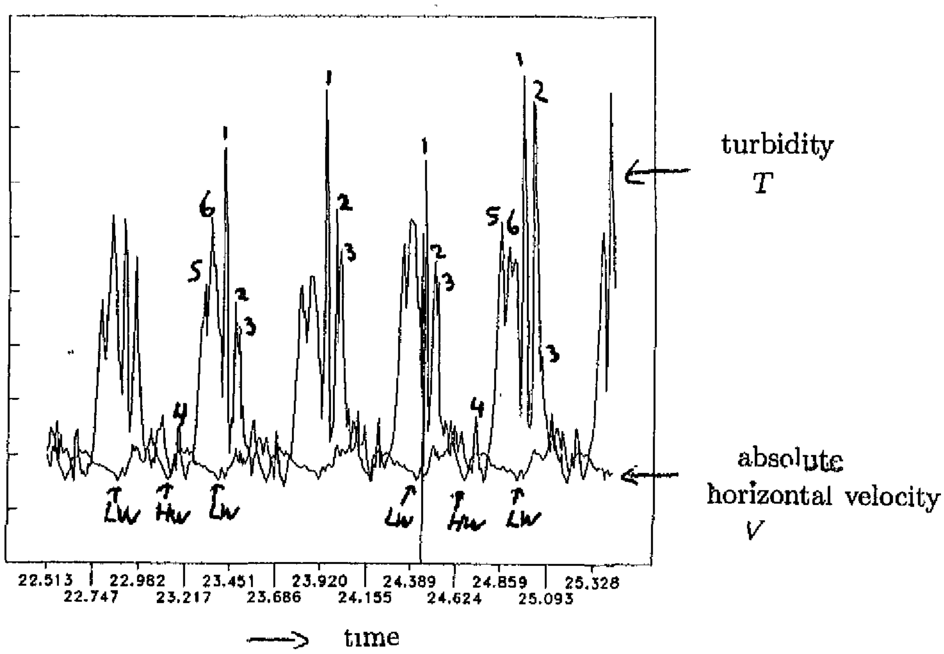


Figure 2.3: Turbidity T and absolute value of the horizontal velocity V over a period of three days in November 1988 at Bath

Remark: In figure 2.3 we notice that a peak in velocity causes a peak in turbidity, with some time delay.

At the turn of flood-tide, i.e. the high water turn (HW-turn), the absolute velocity is zero. However, there is still some cohesive sediment in suspension in the water column. After this turn, we notice a peak of turbidity (4). This peak is caused by the increasing velocity. It is a small peak, in comparison with the peak just after the LW-turn. This may be caused by the fact the water level is still very high. The water is streaming over the marshes and the shoals, which consist of a lot of sand, and not much (fresh) sediment. So the peak (4) will be low. A few moments after peak (4), the water level H is decreased, and we note a high peak in the (absolute) velocity. This velocity peak causes two high turbidity peaks, (5) and (6), since due to the high velocity, all suspended sediment will be eroded from the bottom of the ebb-channels. The erosion of the fresh sediment may cause the high peaks, (5) and (6), of turbidity.

2.5 Conclusions

We want to construct a mathematical model which describes the tidal effects on the cohesive sediment concentration in the Western Scheldt. For this purpose we use turbidity data obtained by an instrument named "Zeekat" at three locations in the Western Scheldt: Vlissingen, Middelgat and Bath. We decide to concentrate on the situation at Bath, since at this location, the influence of the Scheldt river is most significant. The cohesive sediment concentration data are computed out of the turbidity data, using the knowledge of the calibrations, which are determined once a week. Looking at a graph of the turbidity, we notice particular peaks. We related these peaks with the variations of the tide.

Knowledge of the tidal variations, means knowledge of the water depth H and of the horizontal velocity V. The influences of both of them, especially the influence of the horizontal velocity is examined in this chapter. It appeared to be very important.

In the next chapters, we will state a mathematical model, including some unknown parameters, which describes the tidal effects on the cohesive sediment concentration in the Western Scheldt. We will examine the influence of the horizontal velocity V only. In later chapters we will also take the water depth H into consideration, when we shall distinguish between ebb and flood.

Chapter 3

Mathematical model

3.1 Introduction

In the previous chapter we examined the tidal effects on the cohesive sediment concentration near Bath, in the Western Scheldt. It appeared that both the water depth H and the horizontal velocity V have influence on the cohesive sediment concentration.

In this chapter we examine a conservation of mass equation, which describes the influence of tide on the cohesive sediment concentration. We simplify this equation by considering one column of water. We only consider the processes of erosion and sedimentation, and we give the mathematical equations for these processes. We will also give a mathematical expression for the bed shear stress (τ_b). The unknown parameters of these equations will be estimated in the next chapters, using the method of extended Kalman filtering.

3.2 Conservation of Mass

We assume that all suspended cohesive sediment is well mixed throughout the vertical water column. The depth averaged concentration of the suspended cohesive sediment satisfies the conservation of mass equation [22],

$$\frac{\partial C}{\partial t} + u\frac{\partial C}{\partial x} + v\frac{\partial C}{\partial y} = \frac{1}{H}(\frac{\partial}{\partial x}(HD_x\frac{\partial C}{\partial x})) + \frac{1}{H}(\frac{\partial}{\partial y}(HD_y\frac{\partial C}{\partial y})) + \frac{S}{H} \quad (3.1)$$

where C is the depth integrated concentration of the suspended cohesive sediment ($\mathrm{Kg\,m^{-3}}$); u, v are the depth integrated velocity components ($\mathrm{m\,s^{-1}}$);

D_x, D_y are the horizontal dispersion coefficients in the x and y directions ($\mathrm{m^2\,s^{-1}}$); H is the water depth (m) and S represents the source term ($\mathrm{Kg\,m^{-2}\,s^{-1}}$), which may be described as the amount of erosion minus the amount of sedimentation at the bottom.

We start with a simple model. We consider a column of water, with hight H and we assume that the total amount of sediment into the water column is only influenced by erosion and sedimentation at the bottom. So we neither look at the advection terms, nor at the diffusion terms of the cohesive sediment. This situation is shown in figure 3.1. We simplify Eqn (3.1),

$$\frac{\partial C}{\partial t} = \frac{S}{H} = \frac{\partial E}{\partial t} - \frac{\partial d}{\partial t} \tag{3.2}$$

where the source term S is computed from the processes of erosion E and sedimentation d. This will be explained in the next section.

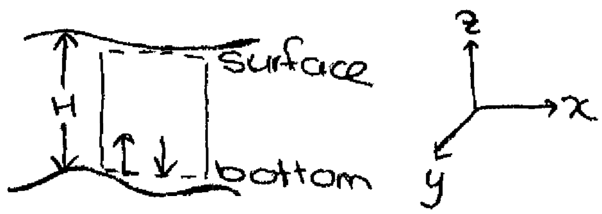


Figure 3.1: Illustration of the simplified situation, looking at one column of water

3.3 Erosion and Sedimentation

3.3.1 Erosion

The rate of erosion E is

$$\frac{\partial E}{\partial t} = M\,(\frac{\tau_b}{\tau_{ce}} - 1), \qquad \tau_b \geq \tau_{ce} \tag{3.3}$$

$$\frac{\partial E}{\partial t} = 0, \qquad \tau_b < \tau_{ce} \tag{3.4}$$

where M is an erosion constant ($\mathrm{Kg\,m^{-2}\,s^{-1}}$), τ_b is the bed shear stress ($\mathrm{N\,m^{-2}}$), and τ_{ce} is the critical bed shear stress above which erosion occurs ($\mathrm{N\,m^{-1}}$). Assume τ_{ce} is a constant.

3.3.2 Sedimentation

The rate of sedimentation d is

$$\frac{\partial d}{\partial t} = C W_s(1 - \frac{\tau_b}{\tau_{cd}}), \qquad \tau_b \leq \tau_{cd} \tag{3.5}$$

$$\frac{\partial d}{\partial t} = 0, \qquad \tau_b > \tau_{cd} \tag{3.6}$$

where W_s is the settling velocity ($\mathrm{m\,s^{-1}}$), τ_b is the bed shear stress ($\mathrm{N\,m^{-2}}$), and τ_{cd} is the critical bed shear stress below which sedimentation occurs ($\mathrm{N\,m^{-2}}$). Assume τ_{cd} is a constant.

We assume the settling velocity W_s to be a constant. But this is not true. Cohesive sediment particles easily stick together. This is called flocculation and is described by van Leussen, 1991 in [11].

Flocks (created by flocculation) are falling faster than single cohesive sediment particles, but when the flocks are too big, they may hinder each other. In that case the flocks are falling slower. So the settling velocity W_s depends on the size of the flocks. The flocculation is depending on, for instance:

- the concentration of the sediment particles
- the concentration of organic elements
- the temperature of the water
- the turbulence of the water

In order to describe the effect of flocculation on the settling velocity, we define three concentration ranges [22],

$$W_s = W_{const}, \qquad C < C_1 \tag{3.7}$$

$$W_s = K_1 C^n, \qquad C_1 \leq C \leq C_2 \tag{3.8}$$

$$W_s = W_{s0}(1 - K_2 C)^\beta, \qquad C > C_2 \tag{3.9}$$

Where W_{const} is a constant value of the settling velocity and K_1, K_2, W_{s0}, n and β are coefficients depending on the sediment type and the salinity. Nevertheless, we assume that W_s has a constant value.

3.4 Description of the parameters

The model we examine is described by Eqns (3.2)–(3.6). We distinguish three situations: the situation of erosion ($\tau_b \geq \tau_{ce}$), the situation of sedimentation ($\tau_b \leq \tau_{cd}$), and the situation in which neither erosion nor sedimentation occurs ($\tau_{cd} < \tau_b < \tau_{ce}$):

$$\frac{\partial C}{\partial t} = \frac{M}{H}\left(\frac{\tau_b}{\tau_{ce}} - 1\right) \qquad \tau_b \geq \tau_{ce} \tag{3.10}$$

$$\frac{\partial C}{\partial t} = 0 \qquad \tau_{cd} < \tau_b < \tau_{ce} \tag{3.11}$$

$$\frac{\partial C}{\partial t} = \frac{CW_s}{H}\left(\frac{\tau_b}{\tau_{cd}} - 1\right) \qquad \tau_b \leq \tau_{cd} \tag{3.12}$$

The data of the cohesive sediment concentration C are available. The water depth H is a known value for every discrete time step k. The parameters M, W_s, τ_{ce} and τ_{cd} are the unknown parameters we shall estimate. We assume them to be constants. The bed shear stress τ_b may be expressed in terms of the horizontal velocity V near the bottom and the Chézy-coefficient Ch. We use a simple equation to express the bed shear stress (Van Leussen III, 1981)

$$\tau_b = \frac{\rho_w\, g\, V^2(\text{bottom})}{(Ch)^2} \tag{3.13}$$

Where we assume

$V(\text{bottom})$	=	horizontal velocity at the bottom			
ρ_w	=	density of water	=	1000	$\text{Kg}\,\text{m}^{-3}$
g	=	gravitation constant	=	9,81	$\text{m}\,\text{s}^{-2}$
Ch	=	Chézy coefficient	=	50	$\text{m}^{-\frac{1}{2}}\,\text{s}^{-1}$

For this bed shear stress expression, we need data of the horizontal velocity near the bottom of the water column, but only the data of the horizontal velocity near the surface are available. So we have to find an expression to compute the necessary velocity from the available data. Mulder [21] gave a parabolic expression of the distribution of the velocity in the vertical direction:

$$V(z) = \bar{V} \cdot (m+1)\left(\frac{z}{H}\right)^m \tag{3.14}$$

where $V(z)$ is the horizontal velocity at water level z, $\bar{V}$ is the depth averaged velocity, m is a constant (≈ 0.15), and H is the water depth. The velocity near the surface $V(\text{surface})$ is known. The water level at which was measured near the surface is 1 meter below the surface, i.e. $z(\text{surface}) = (H - 1)$ m. We like to know the horizontal velocity near the bottom, i.e. 2 meters above

the bottom of the water column: $z(\text{bottom}) = 2$ m.
Keeping this in mind, we get

$$\begin{aligned} V(\text{surface}) &= V(H-1) = \bar{V}\cdot(1.15)\left(\frac{H-1}{H}\right)^{0.15} \\ V(\text{bottom}) &= V(2) = \bar{V}\cdot(1.15)\left(\frac{2}{H}\right)^{0.15} \\ &= V(\text{surface})\left(\frac{2}{H-1}\right)^{0.15} \end{aligned}$$

So at each time step k,

$$V_k^2(\text{bottom}) = V_k^2(\text{surface})\left(\frac{2}{H_k-1}\right)^{0.30} \tag{3.15}$$

We express the bed shear stress at time step k,

$$\begin{aligned} \tau_b(k) &= \frac{\rho_w\, g\, V_k^2(\text{bottom})}{(Ch)^2} \\ &= \frac{\rho_w\, g\, V_k^2(\text{surface})}{(Ch)^2}\left(\frac{2}{H_k-1}\right)^{0.30} \end{aligned} \tag{3.16}$$

3.5 Conclusions

In this chapter we gave a simple conservation of mass equation, which describes the tidal effects on the cohesive sediment concentration in a column of water. We considered three situations: the situation of erosion ($\tau_b \geq \tau_{ce}$), the situation of sedimentation ($\tau_b \leq \tau_{cd}$), and the situation where neither erosion nor sedimentation occurs ($\tau_{cd} < \tau_b < \tau_{ce}$). This is described by the discrete equations,

$$\begin{aligned} C_{k+1} &= C_k + \frac{\alpha M}{H_k}\left(\frac{\tau_b(k)}{\tau_{ce}} - 1\right) && \tau_b(k) \geq \tau_{ce} \\ C_{k+1} &= C_k && \tau_{cd} < \tau_b(k) < \tau_{ce} \\ C_{k+1} &= C_k + \frac{\alpha C_k W_s}{H_k}\left(\frac{\tau_b(k)}{\tau_{cd}} - 1\right) && \tau_b(k) \leq \tau_{cd} \end{aligned}$$

where C_k is the cohesive sediment concentration at time step k, M is the erosion constant, W_s is the settling velocity, τ_{ce} is the critical bed shear stress above which erosion occurs, τ_{cd} is the critical bed shear stress below which sedimentation occurs, and $\alpha = \Delta t$ is the discrete time step. The bed shear stress $\tau_b(k)$ is given by Eqn (3.16).

Using these equations and the data of the cohesive sediment concentrations C_k at each time step k, we shall estimate the unknown parameters M, W_s, τ_{ce} and τ_{cd} appearing in these equations.

To estimate these parameters we will use the method of extended Kalman filtering. This method will be described in the next chapter. In later chapters, we are going to estimate the unknown parameters.

Remark: We consider this problem to be a stochastic filtering problem. This is the first time this approach is used. All previous investigations used correlation and regression analysis techniques to estimate the parameters of the model.[1]

[1] An interesting study is made by Verlaan & Spanhoff, 1992. But they examined the global shape of the turbidity figure, and we are interested in the peaks of this figure.

Chapter 4

Estimation method

4.1 Introduction

Thus far we examined the tidal effects on the cohesive sediment concentration near Bath, in the Western Scheldt (in chapter 2). We also gave a mathematical model (in chapter 3) which describes the tidal effects on the cohesive sediment concentration. Four parameters in this model are unknown. We want to estimate these parameters by the method of extended Kalman filtering. This method will be described in this chapter.

We give a general description of the Kalman filtering method, and we explain why we choose this method for the estimation. After that, we state the mathematical equations of the (extended) Kalman filter. In further chapters we will apply this extended Kalman filtering method to estimate the four unknown parameters of the mathematical model.

4.2 The Kalman filtering method

The Kalman filter is developed in the early sixties by Kalman and Bucy. It gives for linear dynamical systems, the optimal estimator for the state of the dynamical system, in the sense of minimum variance. For nonlinear dynamical systems we need an extension of the Kalman filter, i.e. the extended Kalman filter.

The Kalman filtering problem is that of determining the minimum variance estimator of X_k on the basis of the observations $Y_0, Y_1, \cdots, Y_k$. This really means calculating $E[X_k \,|\mathbf{Y}_k]$; or equivalently, $\hat{X}_{k|k}$. The Kalman filter is recursive, in the sense that once $\hat{X}_k$ (the estimate of X on time k) has been determined on the basis of the measurements $Y_0, Y_1, \cdots, Y_k$, we can determine $\hat{X}_{k+1}$ on the basis of the knowledge of $\hat{X}_k$ and the new measurement Y_{k+1}. This recursive property will be very useful in our estimation problem.

4.3 Application of the method

We apply the Kalman filter to estimate the four unknown parameters of the model. The measurements $Y_0, Y_1, \cdots, Y_k$ are the measured cohesive sediment concentrations $C_0, C_1, \cdots, C_k$ at time steps $0, 1, \cdots, k$. We consider the unknown parameters as random variables and we augment the state vector with these variables. Since the extended Kalman filter yields estimates of the state vector, it also provides estimates of the uncertain parameters.
Remark: A disadvantage of this on-line estimating procedure is that initially the filter uses the wrong parameters, which is detrimental to the performances of the filter. This disadvantage may be overcome by repeating the whole process using the estimated values of the parameters from the first run as initial estimates of the parameters for a second run, and so on. We will use this (off-line) recursive approach.

Apart from the fact that we do not need to store the previous measurements while updating the estimate, the Kalman filtering method is useful, because it takes into account the system and the measurement noise. We have to take into account the system noise, because we used a simplified model to describe the tidal effects on the cohesive sediment concentration near Bath.

4.4 Mathematical description of the method

4.4.1 Linear Kalman filter

The Kalman filtering method is based on the situation of the discrete-time linear stochastic dynamical system (Bagchi, 1993)

$$X_{k+1} = A_k X_k + F_k W_k \tag{4.1}$$

$$Y_k = C_k X_k + V_k, \qquad k \geq 0 \tag{4.2}$$

where the n-dimensional random vector X_k denotes the state at the time-instant k, the r-dimensional random vector W_k is the system disturbance, the m-dimensional random vector Y_k denotes the observation and V_k is the observation error. We assume that X_0, $\{W_k\}$ and $\{V_k\}$ are jointly Gaussian and mutually independent, $k \geq 0$. Assume X_0 has mean $\bar{x}(0)$ and covariance matrix $\bar{P}(0)$ and assume $EW_k = EV_k = 0, k \geq 0$, i.e. $\{W_k\}$ and $\{V_k\}$ are zero mean white Gaussian sequences. We get:

$$E\left[\begin{pmatrix} W_k \\ V_k \end{pmatrix} \begin{pmatrix} W_l^T & V_l^T \end{pmatrix} \right] = \begin{pmatrix} Q(k) & 0 \\ 0 & R(k) \end{pmatrix} \delta_{kl} \tag{4.3}$$

with $Q(k) \geq 0$ and $R(k) > 0$ for all $k \geq 0$ and where δ_{kl} the Kronecker delta function.

We use the following Kalman filter equations (write $\hat{X}_k$ to denote $\hat{X}_{k|k}$):

Measurement update:

$$\hat{X}_{k+1} = \hat{X}_{k+1|k} + K_{k+1}[Y_{k+1} - C_{k+1}\hat{X}_{k+1|k}] \tag{4.4}$$

$$\begin{aligned} K_{k+1} \triangleq\ & P_{k+1|k}C_{k+1}^T[C_{k+1}P_{k+1|k}C_{k+1}^T \\ & +R(k+1)]^{-1} \end{aligned} \tag{4.5}$$

$$\begin{aligned} P_{k+1} =\ & P_{k+1|k} - P_{k+1|k}C_{k+1}^T[C_{k+1}P_{k+1|k}C_{k+1}^T \\ & +R(k+1)]^{-1}C_{k+1}P_{k+1|k} \end{aligned} \tag{4.6}$$

where $P_{k+1|k} \triangleq E[\tilde{X}_{k+1|k}\tilde{X}_{k+1|k}^T]$ and $\tilde{X}_{k+1|k} = X_{k+1} - \hat{X}_{k+1|k}$, and K_{k+1} is the filter gain

Time update:

$$\hat{X}_{k+1|k} = A_kY_k \tag{4.7}$$

$$P_{k+1|k} = A_kP_kA_k^T + F_kQ(k)F_k^T \tag{4.8}$$

where $P_k \triangleq E[\tilde{X}_k\tilde{X}_k^T]$.

Initial conditions:

$$\hat{X}_{0|-1} = \bar{x}(0) \tag{4.9}$$

$$P_{0|-1} = \bar{P}(0) \tag{4.10}$$

4.4.2 Extended Kalman filter

Consider the following non-linear system:

$$\begin{aligned} X_{k+1} &= f(k, X_k) + g(k, X_k)W_k && (4.11)\\ Y_k &= h(k, X_k) + V_k && (4.12) \end{aligned}$$

where $A_k X_k$, F_k and $C_k X_k$ of our earlier linear model are now replaced by $f(k, X_k)$, $g(k, X_k)$ and $h(k, X_k)$, where $f(k, \cdot)$, $h(k, \cdot)$ are now non-linear in general and $g(k, \cdot)$ is not necessarily a constant function. $\{W_k\}$ and $\{V_k\}$ are, as before, and X_0 is a Gaussian random vector. We assume that X_0, $\{W_k\}$ and $\{V_k\}$ are mutually independent, $EW_kW_k^T = Q(k)$, $EV_kV_k^T = R(k)$ and X_0 is Gaussian with mean $\bar{x}(0)$ and covariance matrix $\bar{P}(0)$. Suppose, we have obtained $\hat{X}_{k|k-1}$ and $\hat{X}_k$ by some means and we want to update them by using the new data Y_{k+1}. Define

$$\begin{aligned} A_k &= \left.\frac{\partial f(k,x)}{\partial x}\right|_{x=\hat{X}_k} && (4.13)\\ F_k &= g(k, \hat{X}_k) && (4.14)\\ C_k &= \left.\frac{\partial h(k,x)}{\partial x}\right|_{x=\hat{X}_{k|k-1}} && (4.15) \end{aligned}$$

Assuming smoothness of the functions $f(k, \cdot)$, $g(k, \cdot)$ and $h(k, \cdot)$, we may expand these functions in Taylor series as follows

$$\begin{aligned} f(k, X_k) &= f(k, \hat{X}_k) + A_k(X_k - \hat{X}_k) + \text{higher order terms}\\ g(k, X_k) &= g(k, \hat{X}_k) + \text{higher order terms}\\ h(k, X_k) &= h(k, \hat{X}_{k|k-1}) + C_k(X_k - \hat{X}_{k|k-1}) + \text{higher order terms} \end{aligned}$$

Neglecting higher order terms and assuming the knowledge of $\hat{X}_k$ and $\hat{X}_{k|k-1}$, we get the approximate one-step transition model from Eqns (4.11) and (4.12) as

$$\begin{aligned} X_{k+1} &= [f(k, \hat{X}_k) + A_k(X_k - \hat{X}_k)]X_k + g(k, \hat{X}_k)W_k\\ &= A_kX_k + F_kW_k + L_k && (4.16) \end{aligned}$$

$$\begin{aligned} Y_k &= h(k, \hat{X}_{k|k-1}) + C_k(X_k - \hat{X}_{k|k-1}) + V_k\\ &= C_kX_k + V_k + M_k && (4.17) \end{aligned}$$

where

$$L_k = f(k, \hat{X}_k) - A_k \hat{X}_k \tag{4.18}$$

$$M_k = h(k, \hat{X}_{k|k-1}) - C_k \hat{X}_{k|k-1} \tag{4.19}$$

Note that A_k, F_k, C_k, L_k and M_k are all known at time-instant k.

Let us now apply the Kalman filter equations to this approximate model in Eqns (4.16) and (4.17). We get
Measurement update:

$$\hat{X}_{k+1} = \hat{X}_{k+1|k} + K_{k+1}[Y_{k+1} - h(k+1, \hat{X}_{k+1|k})] \tag{4.20}$$

$$\begin{aligned} P_{k+1} &= P_{k+1|k} - P_{k+1|k} C_{k+1}^T [C_{k+1} P_{k+1|k} C_{k+1}^T \\ &\quad + R(k+1)]^{-1} C_{k+1} P_{k+1|k} \end{aligned} \tag{4.21}$$

$$K_{k+1} = P_{k+1|k} C_{k+1}^T [C_{k+1} P_{k+1|k} C_{k+1}^T + R(k+1)]^{-1} \tag{4.22}$$

Time update:

$$\hat{X}_{k+1|k} = A_k \hat{X}_k + L_k = f_k(\hat{X}_k) \tag{4.23}$$

$$P_{k+1|k} = A_k P_k A_k^T + F_k Q(k) F_k^T \tag{4.24}$$

Initialization:

$$\hat{X}_{0|-1} = \bar{x}(0) \tag{4.25}$$

$$\bar{P}_{0|-1} = \bar{P}(0) \tag{4.26}$$

4.5 Conclusions

We will use the extended Kalman filter to estimate the four unknown parameters of the mathematical model. The extended Kalman filter determines the minimum variance estimator of X_k on the basis of the observations $Y_0, Y_1, \cdots Y_k$. The method is recursive, i.e. we do not need to store the previous measurements while updating the estimate, and it takes into account the system and the measurement noise. So this method is very useful for our estimation problem.

We will use the method of the extended Kalman filter in the next chapters to obtain estimates of the four unknown parameters of the model, which describes the tidal effects on the cohesive sediment concentration near Bath in the Western Scheldt.

Chapter 5

Estimation of the unknown parameters

5.1 Introduction

In the previous chapters we noticed that the cohesive sediment concentration in the Western Scheldt may be influenced by tidal effects. We gave a mathematical model to describe these effects. This model contains four unknown parameters: the erosion constant M, the settling velocity W_s, the critical bed shear stress for erosion τ_{ce}, and the critical bed shear stress for sedimentation τ_{cd}. We want to estimate these unknown parameters by the method of extended Kalman filtering. This method is described in chapter 4. In this chapter we estimate these unknown parameters using the Kalman filtering method.

We will first show, how we apply this Kalman filter to the model we are looking at. After that, we will give the mathematical systems we need to determine the estimates of the four unknown parameters. Using these systems and applying the (extended) Kalman filter, we obtain estimates of the four unknown parameters. These obtained estimates will appear not to be sufficient, so we will extend the model in later chapters, in order to obtain more reliable estimates.

5.2 Application of the Kalman filter

We want to estimate the four unknown parameters M, W_s, τ_{ce} and τ_{cd}, using the Eqns (3.2)–(3.6). For an arbitrary unknown parameter θ we may write

$$\frac{\partial C}{\partial t} = f(\theta, C, H, \tau_b, t) \tag{5.1}$$

according to Eqns (3.10)–(3.12). Assume $f(..)$ is an arbitrary function. The subscript $\cdot_k$ refers to the time $\Delta t \cdot k$.
Or, in discrete version,

$$C_{k+1} = C_k \,+\, \Delta t \cdot F(\theta_k, C_k, H_k, \tau_b(k)) \tag{5.2}$$

where F(..) is a discrete function, or a discretisized continuous function.
We want to estimate the unknown parameter θ_k, while making use of the knowledge of the cohesive sediment concentration C_k, the water depth H_k and the bed shear stress $\tau_b(k)$, all at time k.
Using the Kalman filter Eqns (4.1) and (4.2), we may write the observation

$$Y_k \;=\; C_k$$

i.e. we observe the cohesive sediment concentration C_k. Since we assume a certain observation noise V_k, we write

$$Y_k = C_k \,+\, V_k \tag{5.3}$$

The Kalman filter is developed to estimate X_k. We want to obtain an estimation of θ_k, so we have to include θ_k as a component into the state vector X_k. Since we want to state Y_k as a function of X_k, according to Eqn (4.2), we also include C_k into X_k. Thus we write

$$X_k = \begin{bmatrix} C_k \\ \theta_k \end{bmatrix} \tag{5.4}$$

to obtain

$$X_{k+1} \;=\; A_k X_k \,+\, F_k W_k \tag{5.5}$$
$$Y_k \;=\; C X_k \,+\, V_k \tag{5.6}$$

where W_k denotes the system disturbance, and $C = \begin{bmatrix} 1 & 0 \end{bmatrix}$. Note that these equations have the same shape as Eqns (4.1) and (4.2) of the Kalman

filter, and thus we may apply the Kalman filter to obtain an estimate of X_k, i.e. to obtain $\hat{C}_k$ and $\hat{\theta}_k$.
In the next section we will state this system for specific θ.

Remark: Although we assume that the unknown parameter θ is a constant, we add a certain noise term W_{k_2} to the unknown parameter:

$$\theta_{k+1} = \theta_k + W_{k_2}$$

in order to indicate the uncertainty in the constant value for θ.

5.3 Mathematical systems

5.3.1 Original model

We use Eqns (3.10)–(3.12),

$$\begin{aligned}
\frac{\partial C}{\partial t} &= \frac{M}{H}\left(\frac{\tau_b}{\tau_{ce}} - 1\right) && \tau_b \geq \tau_{ce} \\
\frac{\partial C}{\partial t} &= 0 && \tau_{cd} < \tau_b < \tau_{ce} \\
\frac{\partial C}{\partial t} &= \frac{CW_s}{H}\left(\frac{\tau_b}{\tau_{cd}} - 1\right) && \tau_b \leq \tau_{cd}
\end{aligned}$$

and estimate the erosion constant M and the settling velocity W_s. We assume the critical bed shear stresses for erosion and for sedimentation, τ_{ce} and τ_{cd}, are not varying. We consider three ranges for $\tau_b(k)$, according to Eqns (3.10)–(3.12). When $\tau_b(k) \geq \tau_{ce}$ we estimate the erosion constant M and we do not vary W_s. When $\tau_{cd} < \tau_b(k) < \tau_{ce}$ there is no erosion and no sedimentation; we do not vary both the erosion constant M and the settling velocity W_s. When $\tau_b(k) \leq \tau_{cd}$, we estimate the settling velocity W_s and we do not vary the erosion constant M.[1]

The critical bed shear stresses τ_{ce} and τ_{cd} will be estimated in the same way, not varying the erosion constant M and the settling velocity W_s. Since we want to estimate four parameters, we need four systems to estimate them. We will describe these (discrete) systems in the next sections.

[1]It is clear that this model is a rough approximation of the physical reality, so it will only give a first approximation of the unknown parameters.

5.3.2 Discrete system to estimate the erosion constant M

Consider the continuous model

$$\frac{\partial C}{\partial t} = \frac{1}{H}\frac{\partial E}{\partial t} = \frac{M}{H}\left(\frac{\tau_b}{\tau_{ce}} - 1\right) \qquad \tau_b \geq \tau_{ce} \tag{5.7}$$

and it's discrete version

$$C_{k+1} - C_k = \Delta t \frac{M_k}{H_k}\left(\frac{\tau_b(k)}{\tau_{ce}} - 1\right) \qquad \tau_b(k) \geq \tau_{ce} \tag{5.8}$$

Using this discrete model, we obtain the following system ($\Delta t = \alpha$):

$$X_{k+1} = \begin{bmatrix} 1 & \frac{\alpha}{H_k}(\frac{\tau_b(k)}{\tau_{ce}} - 1) \\ 0 & 1 \end{bmatrix} X_k + F_k W_k \tag{5.9}$$

$$Y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} X_k + V_k \tag{5.10}$$

where $X_k = \begin{bmatrix} C_k \\ M_k \end{bmatrix}$, τ_{ce} is a constant, W_k is the system noise, and V_k is the measurement noise, all at time step k.
Write $\tau_b(k)$ as in Eqn (3.16)

$$\tau_b(k) = \frac{\rho_w\, g\, V_k^2(\text{surface})\, 2^{0.30}}{(Ch)^2\, (H_k - 1)^{0.30}}$$

With system (5.9)–(5.10), we are able to estimate the unknown erosion constant M, using the linear Kalman filter.

5.3.3 Discrete system to estimate the settling velocity W_s

Consider the continuous model

$$\begin{aligned} \frac{\partial C}{\partial t} = -\frac{1}{H}\frac{\partial d}{\partial t} &= -C\frac{W_s}{H}\left(1 - \frac{\tau_b}{\tau_{cd}}\right) \qquad \tau_b \leq \tau_{cd} \\ &= C\frac{W_s}{H}\left(\frac{\tau_b}{\tau_{cd}} - 1\right) \end{aligned} \tag{5.11}$$

and it's discrete version

$$C_{k+1} - C_k = \Delta t\, C_k \frac{W_s(k)}{H_k}\left(\frac{\tau_b(k)}{\tau_{cd}} - 1\right) \qquad \tau_b(k) \leq \tau_{cd} \tag{5.12}$$

This results in the system ($\Delta t = \alpha$)

$$
\begin{aligned}
X_{k+1} &= \begin{bmatrix} C_k + \frac{C_k W_s(k)\alpha}{H_k}\left(\frac{\tau_b(k)}{\tau_{cd}} - 1\right) \\ W_s(k) \end{bmatrix} + F_k W_k \\
&= f(k, X_k) + F_k W_k && (5.13) \\
Y_k &= \begin{bmatrix} 1 & 0 \end{bmatrix} X_k + V_k && (5.14)
\end{aligned}
$$

where $X_k = \begin{bmatrix} C_k \\ W_s(k) \end{bmatrix}$, τ_{cd} is a constant, W_k is the system noise and V_k is the measurement noise, all at time step k.

Define

$$
\begin{aligned}
A_k &\doteq \left.\frac{\partial f(k,x)}{\partial x}\right|_{x=\hat{X}_k} = \begin{bmatrix} 1 + \frac{\hat{W}_s(k)\alpha}{H_k}\left(\frac{\tau_b(k)}{\tau_{cd}} - 1\right) & \frac{\hat{C}_k\alpha}{H_k}\left(\frac{\tau_b(k)}{\tau_{cd}} - 1\right) \\ 0 & 1 \end{bmatrix} && (5.15) \\
L_k &= f(k, \hat{X}_k) - A_k \hat{X}_k = \begin{bmatrix} -\frac{\hat{C}_k \hat{W}_s(k)\alpha}{H_k}\left(\frac{\tau_b(k)}{\tau_{cd}} - 1\right) \\ 0 \end{bmatrix} && (5.16)
\end{aligned}
$$

where $\tau_b(k)$ defined by Eqn (3.16).
We have

$$
\begin{aligned}
X_{k+1} &= A_k X_k + F_k W_k + L_k && (5.17) \\
Y_k &= C_k X_k + V_k && (5.18)
\end{aligned}
$$

Using Eqns (5.12)–(5.16) and the extended Kalman filter, we are able to estimate the unknown settling velocity W_s.

5.3.4 Discrete system to estimate the critical bed shear stress for erosion τ_{ce}

Consider the continuous Eqn (5.7),

$$
\frac{\partial C}{\partial t} = \frac{1}{H}\frac{\partial E}{\partial t} = \frac{M}{H}\left(\frac{\tau_b}{\tau_{ce}} - 1\right) \qquad \tau_b \geq \tau_{ce}
$$

and it's discrete version

$$
C_{k+1} - C_k = \Delta t \, \frac{M}{H_k}\left(\frac{\tau_b(k)}{\tau_{ce}(k)} - 1\right) \qquad \tau_b(k) \geq \tau_{ce}(k) \tag{5.19}
$$

This results in the system ($\Delta t = \alpha$)

$$
\begin{aligned}
X_{k+1} &= \begin{bmatrix} C_k + \frac{\alpha M}{H_k}(\frac{\tau_b(k)}{\tau_{ce}(k)} - 1) \\ \tau_{ce}(k) \end{bmatrix} + F_k W_k \\
&= f(k, X_k) + F_k W_k && (5.20) \\
Y_k &= \begin{bmatrix} 1 & 0 \end{bmatrix} X_k + V_k && (5.21)
\end{aligned}
$$

where $X_k = \begin{bmatrix} C_k \\ \tau_{ce}(k) \end{bmatrix}$, M is a constant, W_k is the system noise, and V_k is the measurement noise, all at time step k.

Define

$$
\begin{aligned}
A_k &= \frac{\partial f(k,x)}{\partial x}\bigg|_{x=\hat{X}_k} = \begin{bmatrix} 1 & -\frac{\alpha M}{H_k}\frac{\tau_b(k)}{(\hat{\tau}_{ce}(k))^2} \\ 0 & 1 \end{bmatrix} && (5.22) \\
L_k &= f(k, \hat{X}_k) - A_k \hat{X}_k = \begin{bmatrix} \frac{\alpha M}{H_k}(2\frac{\tau_b(k)}{\hat{\tau}_{ce}(k)} - 1) \\ 0 \end{bmatrix} && (5.23)
\end{aligned}
$$

where $\tau_b(k)$ defined by Eqn (3.16)
We use Eqns (5.20)–(5.23) and the extended Kalman filter to obtain the estimation of the unknown critical bed shear stress τ_{ce} above which erosion occurs.

5.3.5 Discrete system to estimate the critical bed shear stress for sedimentation τ_{cd}

Consider the continuous Eqn (5.11),

$$
\frac{\partial C}{\partial t} = -\frac{1}{H}\frac{\partial d}{\partial t} = C\frac{W_s}{H}\left(\frac{\tau_b}{\tau_{cd}} - 1\right) \qquad \tau_b \le \tau_{cd}
$$

and it's discrete version

$$
C_{k+1} - C_k = \Delta t\, C_k \frac{W_s}{H_k}\left(\frac{\tau_b(k)}{\tau_{cd}(k)} - 1\right) \qquad \tau_b(k) \le \tau_{cd}(k) \qquad (5.24)
$$

This results in the system ($\Delta t = \alpha$)

$$
\begin{aligned}
X_{k+1} &= \begin{bmatrix} C_k + \frac{C_k \alpha W_s}{H_k}(\frac{\tau_b(k)}{\tau_{cd}(k)} - 1) \\ \tau_{cd}(k) \end{bmatrix} + F_k W_k \\
&= f(k, X_k) + F_k W_k && (5.25) \\
Y_k &= \begin{bmatrix} 1 & 0 \end{bmatrix} X_k + V_k && (5.26)
\end{aligned}
$$

where $X_k = \begin{bmatrix} C_k \\ \tau_{cd}(k) \end{bmatrix}$, W_s is a constant, W_k is the system noise, and V_k is the measurement noise, all at time step k.

Define

$$A_k = \left.\frac{\partial f(k,x)}{\partial x}\right|_{x=\hat{X}_k} = \begin{bmatrix} 1 + \frac{\alpha W_s}{H_k}\left(\frac{\tau_b(k)}{\hat{\tau}_{cd}(k)} - 1\right) & -\hat{C}_k \frac{\alpha W_s}{H_k}\frac{\tau_b(k)}{(\hat{\tau}_{cd}(k))^2} \\ 0 & 1 \end{bmatrix} \quad (5.27)$$

$$L_k = f(k, \hat{X}_k) - A_k \hat{X}_k = \begin{bmatrix} \alpha \frac{W_s \hat{C}_k \tau_b(k)}{H_k \hat{\tau}_{cd}(k)} \\ 0 \end{bmatrix} \quad (5.28)$$

where $\tau_b(k)$ defined by Eqn (3.16). We use Eqns (5.25)–(5.28) and the extended Kalman filter to estimate the unknown critical bed shear stress τ_{cd} below which sedimentation occurs.

5.4 Estimation

5.4.1 Preparations

To estimate the unknown parameters, we need initial values for these parameters. Approximate values for these parameters are, according to van Leussen, [10],

$$\begin{aligned} &\tau_{ce} \approx 0.14\,\mathrm{N\,m^{-2}} \\ &\tau_{cd} \approx 0.08\,\mathrm{N\,m^{-2}} \\ &0.1 \leq M \ \leq 4.0 \cdot 10^{-3}\,\mathrm{Kg\,m^{-2}\,s^{-2}} \\ &0.5 \leq W_s \leq 2.0 \cdot 10^{-3}\,\mathrm{Kg\,m^{-2}\,s^{-2}} \end{aligned}$$

We assume τ_{ce}(cohesive sediment) $\leq$ τ_{ce}(sand) $\approx$ $0.2N\,m^{-2}$, because the cohesive sediment consists of little particles, which will be easier mixed with the water then the sand particles.[2]. Further, we assume $H_k = 1$, in order to obtain a low water column, in which we assume ideal instantaneous mixing in the vertical direction.

[2]More about this can be read in chapter 2

5.4.2 Estimation of M and W_s

Method

We use Eqns (5.9)–(5.10) in the case of erosion, i.e. when $\tau_b(k) \geq \tau_{ce}$, and Eqns (5.13)–(5.16) in the case of sedimentation, i.e. when $\tau_b(k) \leq \tau_{cd}$, for given τ_{ce} and τ_{cd}. In the case of erosion we are going to estimate the erosion constant M, and in the case of sedimentation we shall estimate the settling velocity W_s. We use the (extended) Kalman filter to estimate the unknown parameters, with the initial values,

$$\tau_{ce} = 0.14 \ \mathrm{N\,m^{-2}} \tag{5.29}$$

$$\tau_{cd} = 0.08 \ \mathrm{N\,m^{-2}} \tag{5.30}$$

$$\bar{x}_{\mathrm{er.}}(0) = \begin{bmatrix} C(0) \\ M(0) \end{bmatrix} \tag{5.31}$$

$$\bar{x}_{\mathrm{sed.}}(0) = \begin{bmatrix} C(0) \\ W_s(0) \end{bmatrix} \tag{5.32}$$

$$F = I_2 \tag{5.33}$$

$$\bar{P}_{\mathrm{er.}}(0) = \begin{bmatrix} 2.5 \cdot 10^{-5} & 1 \cdot 10^{-5} \\ 1 \cdot 10^{-5} & 4.0 \cdot 10^{-6} \end{bmatrix} \tag{5.34}$$

$$\bar{P}_{\mathrm{sed.}}(0) = \begin{bmatrix} 2.5 \cdot 10^{-5} & 5.10 \cdot^{-7} \\ 5 \cdot 10^{-7} & 1 \cdot 10^{-8} \end{bmatrix} \tag{5.35}$$

$$R(k) = 2.5 \cdot 10^{-5} \tag{5.36}$$

$$Q(k) = \begin{bmatrix} 5 \cdot 10^{-4} & 0 \\ 0 & 0 \end{bmatrix} \tag{5.37}$$

where $M(0)$ and $W_s(0)$ are initial guesses, and $C(0)$ is the measured cohesive sediment concentration at time $k = -1$.

Results

To obtain estimates of the unknown parameters M and W_s, we use the Kalman filtering method as described in chapter 4. This method is used in the FORTRAN-program 'kfmwc1' (appendix C). We obtain the results shown in figure 5.1, where we assume

$$\begin{aligned} M(0) &= 2.0 \cdot 10^{-3} \ \mathrm{Kg\,m^{-2}\,s^{-2}} \\ W_s(0) &= 3.0 \cdot 10^{-3} \ \mathrm{m\,s^{-1}} \end{aligned}$$

From figure 5.1 we may conclude

$$\begin{aligned} \hat{M} &= 0.2 \cdot 10^{-4} \ \mathrm{Kg\,m^{-2}\,s^{-2}} && (5.38) \\ \hat{W}_s &= 0.3 \cdot 10^{-3} \ \mathrm{m\,s^{-1}} && (5.39) \end{aligned}$$

The next step is to determine τ_{ce} and τ_{cd}.

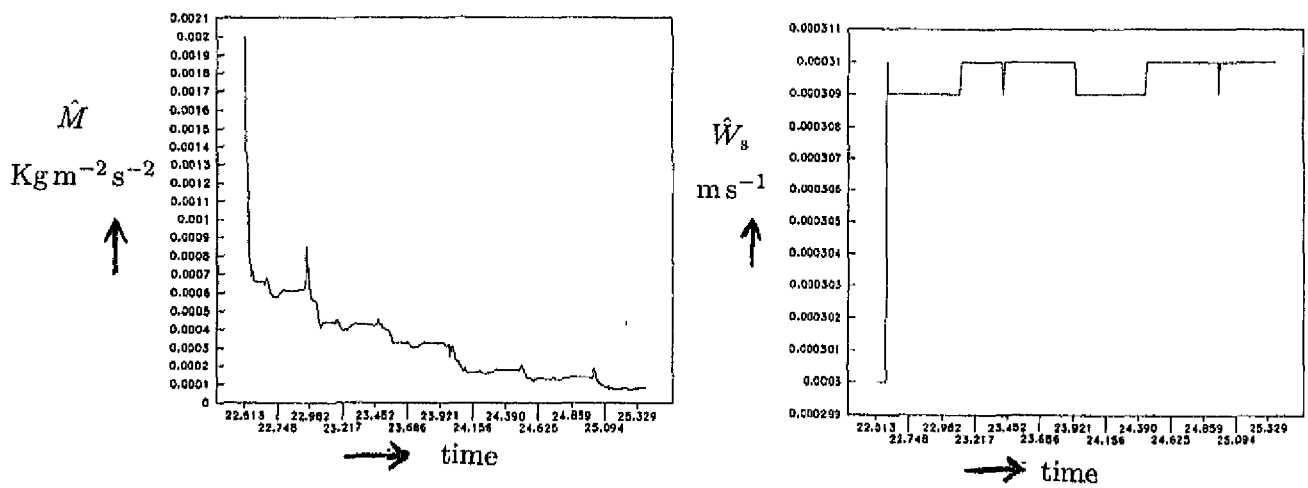


Figure 5.1: Estimates of the erosion constant M and the settling velocity W_s, using the simplified model, over a period of three days

Note: In the figure of the estimated concentrations $\hat{C}$, which is not shown here, I observed some horizontal lines where the figure of the observed concentrations C shows a peak. This may be explained by the fact that I may have chosen the wrong values for τ_{ce} and τ_{cd}. (Note that for $\tau_{cd} \leq \tau_b \leq \tau_{ce}$: $\frac{\partial C}{\partial t} = 0$, according to Eqn (3.11).) Examining the figure of the estimated settling velocity $\hat{W}_s$ and at the figure of the observed cohesive sediment concentration C, I noticed that the peaks in W_s appear at the same time as the highest peaks in $\hat{C}$. This is caused by the Kalman filter: A high peak in the observed concentration C, forces an adaption of the parameter estimation.

To have a notion of the values of the parameters (both in the case of ebb-tide and in the case of of flood-tide), I examined some parts of the time series of November 22, until November 28, where I determined the values of M and

W_s. I found the values,

$$M_{ebb} = 0.1 \cdot 10^{-4} \tag{5.40}$$
$$M_{flood} = 0.1 \cdot 10^{-4} \tag{5.41}$$
$$W_{s,ebb} = 0.3 \cdot 10^{-4} \tag{5.42}$$
$$W_{s,flood} = 0.1 \cdot 10^{-4} \tag{5.43}$$

Notice that the values of ebb-tide are almost the same as the estimated values we obtained until now.
Remark: We distinguish between the case of ebb-tide and that of flood-tide, because these situations are different. More about this is explained in the next chapter.

5.4.3 Estimation of τ_{ce} and τ_{cd}

Method

We use Eqns (5.20)–(5.23) in the case of erosion, i.e. when $\tau_b(k) \geq \tau_{ce}$, and Eqns (5.25)–(5.28) in the case of sedimentation, i.e. when $\tau_b(k) \leq \tau_{cd}$, for given τ_{ce} and τ_{cd}. In the case of erosion we are going to estimate the critical bed shear stress for erosion τ_{ce}, and in the case of sedimentation we shall estimate the critical bed shear stress for sedimentation τ_{cd}. We use the extended Kalman filter to estimate the unknown parameters, with the initial values,

$$M = 0.2 \cdot 10^{-4} \ \mathrm{Kg\,m^{-2}\,s^{-2}}$$
$$W_s = 0.3 \cdot 10^{-3} \ \mathrm{m\,s^{-1}}$$
$$\bar{x}\mathrm{er.}(0) = \begin{bmatrix} C(0) \\ \tau_{ce}(0) \end{bmatrix} \tag{5.44}$$
$$\bar{x}\mathrm{sed.}(0) = \begin{bmatrix} C(0) \\ \tau_{cd}(0) \end{bmatrix} \tag{5.45}$$
$$F = I_2 \tag{5.46}$$
$$\bar{P}\mathrm{er.}(0) = \begin{bmatrix} 2.5 \cdot 10^{-5} & 3 \cdot 10^{-4} \\ 3 \cdot 10^{-4} & 3.6 \cdot 10^{-3} \end{bmatrix} \tag{5.47}$$
$$\bar{P}\mathrm{sed.}(0) = \begin{bmatrix} 2.5 \cdot 10^{-5} & 1.10 \cdot ^{-4} \\ 1 \cdot 10^{-4} & 4 \cdot 10^{-4} \end{bmatrix} \tag{5.48}$$
$$R(k) = 2.5 \cdot 10^{-5} \tag{5.49}$$

$$Q(k) = \begin{bmatrix} 5 \cdot 10^{-4} & 0 \\ 0 & 0 \end{bmatrix} \tag{5.50}$$

where $\tau_{ce}(0)$ and $\tau_{cd}(0)$ are initial guesses, and $C(0)$ is the measured cohesive sediment concentration at time $k = -1$.

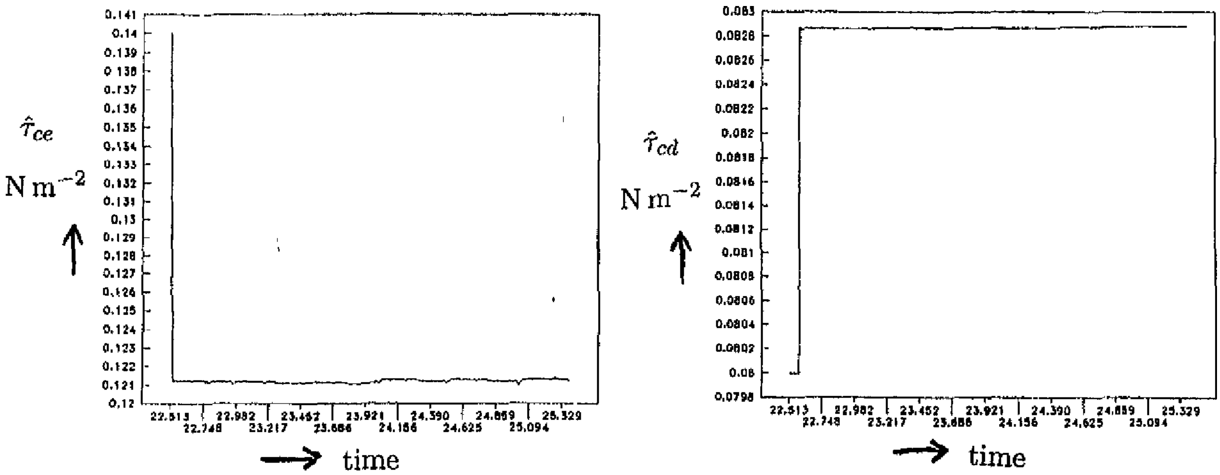


Figure 5.2: Estimates of the critical bed shear stress for erosion τ_{ce} and the critical bed shear stress for sedimentation τ_{cd}, using the simplified model, over a period of three days

Results

To obtain estimates of the unknown parameters τ_{ce} and τ_{cd}, we use the Kalman filtering method as described in chapter 4. This method is used in FORTRAN-program 'kftau1' (appendix C). We obtain the results shown in figure 5.2, where we assumed

$$\begin{aligned} \tau_{ce}(0) &= 0.14 \ \mathrm{N\,m^{-2}} \\ \tau_{cd}(0) &= 0.08 \ \mathrm{N\,m^{-2}} \end{aligned}$$

Note: I still noticed horizontal lines in the figure of the estimated cohesive sediment concentration $\hat{C}_k$, because I did not change the initial τ_{ce} and τ_{cd}. So I had to adapt these initial values to improve the estimated concentration $\hat{C}_k$.

While proceeding the estimation, I noticed that the estimated concentration $\hat{C}_k$ was matching better with the real concentration C_k. The finally obtained estimates of the critical bed shear stresses for erosion and sedimentation, τ_{ce} and τ_{cd}, are

$$\hat{\tau}_{ce} = 0.061 \ \mathrm{N\,m^{-2}} \tag{5.51}$$

$$\hat{\tau}_{cd} = 0.058 \ \mathrm{N\,m^{-2}} \tag{5.52}$$

These values are lying closely together, because in the figure of the observed cohesive sediment concentration, we notice no horizontal lines (compare figure 2.2 in chapter 2). This means $\frac{\partial C}{\partial t} = 0$ only for short periods, i.e. $\tau_{cd} < \tau_b < \tau_{ce}$ only for short periods. Thus $\tau_{cd} \approx \tau_{ce}$.

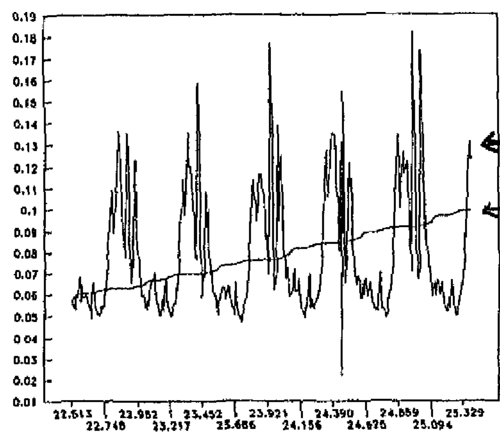


Figure 5.3: The model compared with the observations over a period of three days

using	M	$= 0.2 \cdot 10^{-4}$	$Kg\,m^{-2}\,s^{-2}$
	W_s	$= 0.3 \cdot 10^{-3}$	$m\,s^{-1}$
	τ_{ce}	$= 0.061$	$N m^{-2}$
	τ_{cd}	$= 0.058$	$N m^{-2}$

5.4.4 Obtained model

We have obtained estimates of the four unknown parameters of model (3.10)–(3.12). Substituting the values of these estimates in the model, we obtain figure 5.3. These estimates are characteristic for the location where the measurements were done and for the time of the year. So the model will give an idea of the tidal effects on the cohesive sediment concentration, not considering the vertical diffusion in the water column, the longitudinal advection, the differences between the ebb-stream and the flood-stream, and assuming that the parameters of the model, behave like constants.

Notice that the cohesive sediment concentrations obtained by the model and the observed cohesive sediment concentrations are totally different.

To improve the model, I varied H_k, and I estimated the four unknown parameters in the same way as before. I also estimated M together with τ_{ce}, assuming W_s and τ_{cd} to be constants. I examined the obtained figures, and looked for any correlation between these two parameters. Since I did not found anything particular in these estimations, I decided not to include these estimation procedures into this report.

Since I am not at all satisfied about the estimated values until now, I will extend the model in another way as will be pointed out in the next chapter.

5.5 Conclusions

In this chapter, we applied the (extended) Kalman filter to estimate the unknown parameters of the mathematical model describing the tidal effects on the cohesive sediment concentration in the Western Scheldt. Four parameters of this model are unknown. To estimate these four parameters we need four mathematical discrete systems, one for each of them. Using these systems, we estimated the unknown erosion constant M and the unknown settling velocity W_s, while not varying the critical bed shear stresses for erosion and sedimentation, τ_{ce} and τ_{cd}. We obtained

$$\begin{aligned} \hat{M} &= 0.2 \cdot 10^{-4} \ \mathrm{Kg\,m^{-2}\,s^{-2}} \\ \hat{W}_s &= 0.3 \cdot 10^{-3} \ \mathrm{m\,s^{-1}} \end{aligned}$$

After this we estimated the critical bed shear stress for erosion τ_{ce} and the critical bed shear stress for sedimentation τ_{cd}. We assume the erosion constant M and the settling velocity W_s have values as obtained in the previous estimations. We obtained

$$\begin{aligned} \hat{\tau}_{ce} &= 0.061 \ \mathrm{N\,m^{-2}} \\ \hat{\tau}_{cd} &= 0.058 \ \mathrm{N\,m^{-2}} \end{aligned}$$

Substituting these estimates in the model of the tidal effects we obtain cohesive sediment concentrations which do not agree with the observed cohesive sediment concentrations. So we have to improve the model.

In the next chapter, we shall adapt the bed shear stress equation $\tau_b(k)$, and we will consider the differences between the ebb-stream and the flood-stream, in order to obtain more reliable estimates.

Chapter 6

Extension of the model

6.1 Introduction

In the previous chapter, we estimated the four unknown parameters of the mathematical model, that describes the tidal effects on the cohesive sediment concentration. When we simulate these estimated parameters in the model, we obtained cohesive sediment concentrations, which did not agree with the observed cohesive sediment concentrations. So we want to improve the model in this chapter.

In the previous chapter we used a value for the bed shear stress $\tau_b(k)$, determined by the horizontal velocity of the water near the bottom. We computed $\tau_b(k)$ at each time step k, and we considered no longitudinal convection. In this chapter we are going to examine the bed shear stress $\tau_b(k)$ with a certain time delay, in order to include some kind of longitudinal advection.

Because we will take the tide into consideration, we will distinguish between the ebb-stream (in seaward direction), and the flood-stream (in inland direction).

Using the adapted bed shear stress signal $\tilde{\tau}_b(k)$, and taking into account the differences between the ebb-stream and the flood-stream, we shall estimate the four unknown parameters of the model. These estimates will result in a better model for the cohesive sediment concentrations, but still the concentrations obtained by the model, do not agree with the the observed concentrations.

After all those estimations, we have an idea of the values of the parameters. Substituting these values into the model, we obtain cohesive sediment concentrations calculated from the model, which are agree with the observed cohesive sediment concentrations. From this we may conclude, that the model is correct at that time.

Since the estimates obtained by the Kalman filter, do not satisfy us, and we have found that the model is right, maybe something is wrong in the way we adapt the Kalman filter. We will examine the noise statistics of the used Kalman filtering method in the next chapters.

6.2 Adaptation of the bed shear stress expression

6.2.1 Behaviour of the bed shear stress τ_b

To examine the relation between the observed cohesive sediment concentration C_k and the calculated bed shear stress $\tau_b(k)$ (by Eqn (3.16), we compare the figures of both of them, in figure 6.1. We notice that most of the peaks of the cohesive sediment concentration may be explained by the peaks of the bed shear stress. When we look carefully, we notice a certain time delay of the bed shear stress, in comparison with the observed cohesive sediment concentration. So we extend the bed shear stress expression with a time delay. Doing this, we assume that longitudinal advection takes place.[1]

In this section, we want to determine the time delay of the bed shear stress $\tau_b(k)$. First, we look at the simple constant time delay. After that, we examine the more reliable variable time delay.

6.2.2 Static shift of the bed shear stress $\tau_b(k)$

The observed cohesive sediment concentrations in a short time period from November 22 until November 28 represent data over a short time period around spring-tide. (The date of spring-tide and neap-tide of this period are shown in table 6.1.) In this short time period there are hardly any changes, so we may assume a constant time delay. This keeps the problem simple. (Of course this simplification is not allowable when we look at longer time periods; in that situation we have to determine a variable time delay, as will be described in the next section.)

[1]The physical explanation of the time delay of the bed shear stress, is that the sediment has been eroded at another location and was brought into the water column at Bath by the water movements.

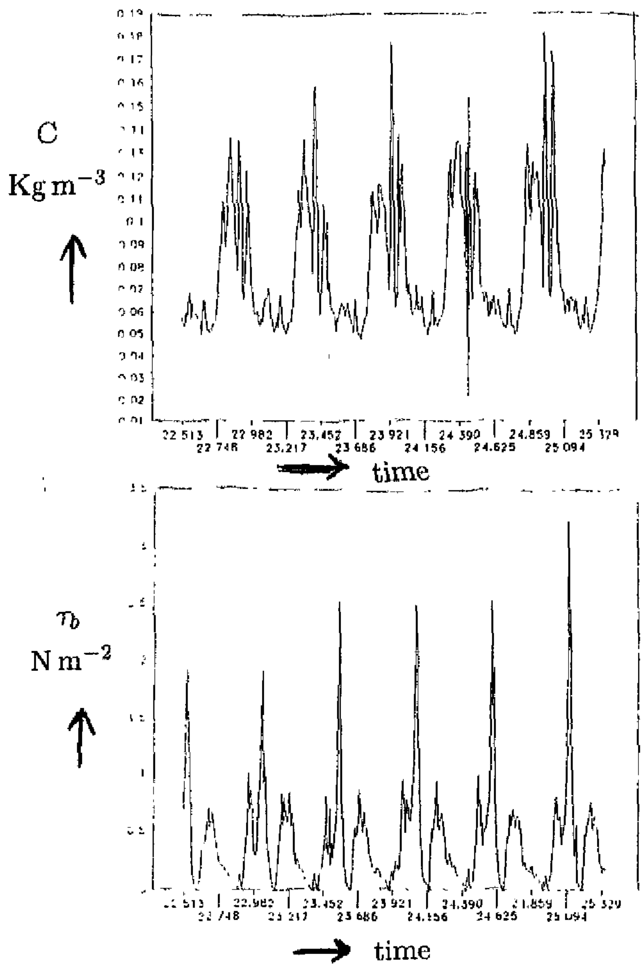


Figure 6.1: The measured cohesive sediment concentration C compared with the bed shear stress τ_b (by Eqn (3.16)) over a period of three days in November 1988 at Bath

I divided the two time series into small parts of for instance, a few days, or a part of a day, and I compared the different series for these parts. I tried to calculate the time delay of the bed shear stress series, with respect to the time series of the cohesive sediment concentration. I found an average time delay of

$$\tau_b(k) := \tau_b(k - 15) \tag{6.1}$$

So the bed shear stress at time step k, $\tau_b(k)$, will be determined from the bed shear stress of 15 time steps earlier, $\tau_b(k - 15)$. This means a time delay of about 3 hours. So the cohesive sediment in the water column near Bath, has been eroded about 3 hours before, upstream or downstream.

Since the velocity of the water is not a constant value, the time delay of the bed shear stress is no constant time delay. For when the distance between Bath and a certain sediment source remains the same and the velocity is changing, then the time delay of the bed shear stress signal has to change also. We will examine this in the next section.

Spring-tide and Neap-tide			
Date	**Time**	**Decimal time number**	**Phase of the moon**
23/11/1988	16:53	23.7035	full moon
01/12/1988	7:49	31.3257	last quarter
09/12/1988	6:36	39.2750	new moon

Table 6.1: Date and time of the spring-tide (i.e. full moon, or new moon) and the neap-tide (i.e. first quarter or last quarter), occurred in November and December 1988

6.2.3 Dynamic shift of the bed shear stress $\tau_b(k)$

In fact the time delay of the bed shear stress is no constant. The time delay of the bed shear stress, with respect to the cohesive sediment concentration, is not the same on each day. In a period around spring-tide, the differences between the time shifts are not very big, but when we consider another time period of a few days, we have to assume a time delay, depending on k. Actually this time delay is not explicitly depending on k, but on the velocity V_k, because the sediment is supplied from locations a certain distance l away. So we have to know the velocity V_k, to determine the time delay on $\tau_b(k)$.

Because the velocity may be distinguished in two directions, i.e. the upstream and the downstream direction, we have to split up the cohesive sediment concentration model into two cases: the case of ebb-tide and the case of flood-tide. The direction of the velocity determines from what location the sediment is transported to Bath. It appears that the distance between Bath and the mud source upstream is longer than the distance between Bath and the mud source downstream (when examining the chart of the situation around Bath). So we have to distinguish between the ebb-stream and the flood-stream.

After examining the figures of the cohesive sediment concentration C and the calculated bed shear stress τ_b, keeping in mind the whole situation around Bath, (i.e. the locations of the sources, the locations of the ebb-channels and the flood-channels, the direction of the flow during ebb-tide, and the direction during flood-tide, etc.) I finally found the following equations for τ_b

$$\tau_{b,ebb}(t) = \tau_b(t - \frac{0.13}{V_t}) \tag{6.2}$$

$$\tau_{b,flood}(t) = \tau_b(t - \frac{0.03}{V_t}) \tag{6.3}$$

where V_t is the absolute value of the velocity, t is the time in seconds, and 0.13 resp. 0.03 is the time in decimal time numbers, so

$$\begin{aligned} 0.13 = 0.13 \times 24 \times 3600\,\text{s} &= 11232\,\text{s} \approx 3\text{ hours} \\ 0.03 = 0.03 \times 24 \times 3600\,\text{s} &= 2592\,\text{s} \approx 1\text{ hour} \end{aligned}$$

In time steps[2] k:

$$\tau_{b,ebb}(k) = \tau_b(k - \frac{1}{0.008380}\frac{0.13}{V_k}) \tag{6.4}$$

$$\tau_{b,flood}(k) = \tau_b(k - \frac{1}{0.008380}\frac{0.03}{V_k}) \tag{6.5}$$

So the time delay of the bed shear stress is depending on the velocity, and on the daily tide (i.e. ebb-tide or flood-tide).

6.3 Distinction between the ebb-stream and the flood-stream

In the previous section we discussed the fact that we have to take into account a certain time delay on the calculated signal of the bed shear stress. For this we need two different bed shear stress expressions, one for the situation of ebb-tide, and one for the situation of flood-tide. However, this is not the only reason, why we want to split up the cohesive sediment concentration model into two different models; one for the case of ebb-tide, and one for the case of flood-tide.

Until now, we considered the simplified situation, assuming one bottom, both for the ebb-stream and the flood-stream, but according to Manni [15]

[2] One decimal time number is $\frac{1}{0.008380}$ time steps.

we have to distinguish between ebb-channels and flood-channels. Since the transport of the (polluted) river sediment is using the ebb-channels, and the transport of the (fresh) sediment from the sea is using the flood-channels. So the amount of cohesive sediment at the bottom during ebb-tide is not the same as the amount of cohesive sediment at the bottom during flood-tide. Sediment which has settled at the bottom during flood-tide, does not have to come into the water column during ebb-tide. The reason why we split up the cohesive sediment concentration model into a case of ebb-tide and a case of flood-tide is the fact that we have to deal with two different situations, with different streams and different channels.

After we split up the model we examined until now, into two different models, one for the case of ebb-tide, and one for the case of flood-tide, we have to estimate eight unknown parameters instead of four. We assume all parameters to be constants, but the parameters of the case of ebb-tide will in general be different from the parameters in the case of flood-tide. We shall estimate the eight parameters:

$$\begin{array}{rcl} M_{ebb} & \text{and} & M_{flood} \\ W_{s,ebb} & \text{and} & W_{s,flood} \\ \tau_{ce,ebb} & \text{and} & \tau_{ce,flood} \\ \tau_{cd,ebb} & \text{and} & \tau_{cd,flood} \end{array}$$

To estimate the eight parameters, we need the knowledge of the water depth H_k at time step k. For instance, suppose we want to estimate the erosion constant M and the settling velocity W_s, not varying the critical bed shear stresses for erosion and sedimentation, τ_{ce} and τ_{cd}. We measure the water level H_k at a time step k. When $H_k \geq 17.4$ m. (i.e. the mean water depth at Bath during November, December 1988), we are going to estimate the erosion constant M and the settling velocity W_s at flood-tide. (We estimate in the same way as in chapter 5.) When $H_k < 17.4$ m. we are going to estimate M and W_s at ebb-tide, i.e. $M(\text{ebb})$ and $W_s(\text{ebb})$.

6.4 Estimation using the extended model

6.4.1 Estimation of M and W_s in the cases of ebb and flood

Method

Because we examine the period of November 22 until November 28, 1988, which is around spring-tide, we will use Eqn (6.1) to determine the time delay of τ_{b_k},

$$\tau_b(k) \quad := \quad \tau_b(k - 15)$$

The estimations are carried out in the same way as in chapter 5, using the same models, except for the time delay on the bed shear stress $\tau_b(k)$.
We assume

$$\begin{aligned}
\tau_{ce,ebb} = 0.08 \quad &\text{and} \quad \tau_{ce,flood} = 0.18 \\
\tau_{cd,ebb} = 0.06 \quad &\text{and} \quad \tau_{cd,flood} = 0.08 \\
Q(k)_e \quad &= \quad Q(k)_f = \begin{bmatrix} 5.0 \cdot 10^{-4} & 0 \\ 0 & 0 \end{bmatrix} \\
R(k)_e \quad &= \quad R(k)_f = 5.0 \cdot 10^{-4}
\end{aligned}$$

with initial values

$$\begin{aligned}
M_{ebb}(0) &= 2.0 \cdot 10^{-3} \quad \text{and} \quad M_{flood}(0) &= 1.0 \cdot 10^{-3} \\
W_{s,ebb}(0) &= 3.0 \cdot 10^{-5} \quad \text{and} \quad W_{s,flood}(0) &= 3.2 \cdot 10^{-5}
\end{aligned}$$

Results

To obtain estimates for the erosion constant M and the settling velocity W_s, both in the cases of ebb-tide and flood-tide, we use the FORTRAN program 'kfhmw'. We obtain the estimates as shown in figure 6.2 and conclude

$$\hat{M}_{ebb} = 2.0 \cdot 10^{-3} \quad \text{and} \quad \hat{M}_{flood} = 1.0 \cdot 10^{-3} \tag{6.6}$$

$$\hat{W}_{s,ebb} = 3.0 \cdot 10^{-5} \quad \text{and} \quad \hat{W}_{s,flood} = 3.5 \cdot 10^{-5} \tag{6.7}$$

Notice that indeed the estimated values for both the erosion constant M and the settling velocity W_s are different in the different cases of ebb-tide and flood-tide.

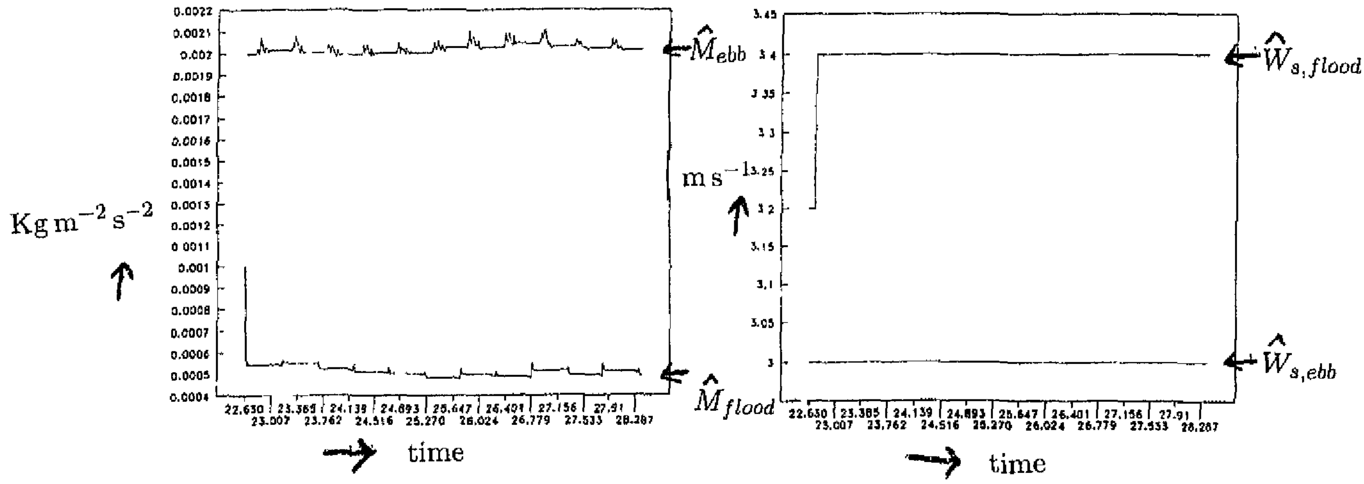


Figure 6.2: Estimates of the erosion constant M and the settling velocity W_s, in the cases of ebb-tide and flood-tide, using the extended model, over a period of six days

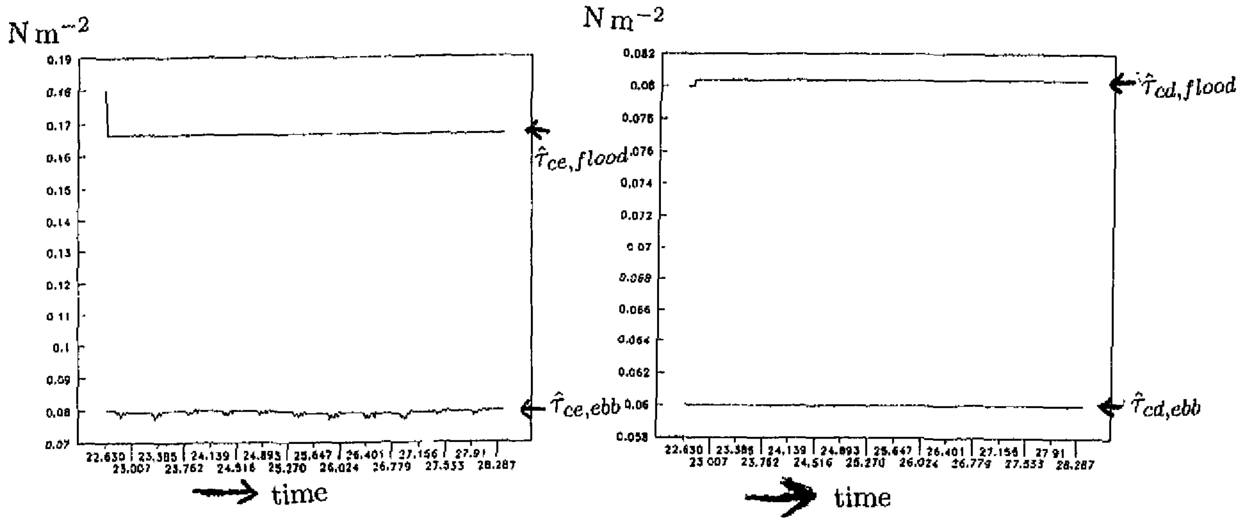


Figure 6.3: Estimates of the critical bed shear stress for erosion τ_{ce} and the critical bed shear stress for sedimentation τ_{cd}, in the cases of ebb-tide and flood-tide, using the extended model, over a period of six days

6.4.2 Estimation of τ_{ce} and τ_{cd} in the cases of ebb and flood

Method

We will use again the constant time delay of the bed shear stress $\tau_b(k)$, i.e. Eqn (6.1). Assume

$$\begin{aligned}
M_{ebb} &= 2.0 \cdot 10^{-3} \quad \text{and} \quad M_{flood} = 1.0 \cdot 10^{-3} \\
W_{s,ebb} &= 3.0 \cdot 10^{-5} \quad \text{and} \quad W_{s,flood} = 3.5 \cdot 10^{-5} \\
Q(k)_e &= Q(k)_f = \begin{bmatrix} 5.0 \cdot 10^{-4} & 0 \\ 0 & 0 \end{bmatrix} \\
R(k)_e &= R(k)_f = 5.0 \cdot 10^{-4}
\end{aligned}$$

with initial values

$$\begin{aligned}
\tau_{ce,e}(0) = 0.08 \quad &\text{and} \quad \tau_{ce,f}(0) = 0.18 \\
\tau_{cd,e}(0) = 0.06 \quad &\text{and} \quad \tau_{cd,f}(0) = 0.08
\end{aligned}$$

where M and W_s for the cases of ebb and flood are as found in the previous section.

Results

To obtain estimates of the critical bed shear stress for erosion τ_{ce} and the critical bed shear stress for sedimentation τ_{cd}, both in the cases of ebb-tide and flood-tide, we use the FORTRAN program 'kfhtt'. We obtain the estimates as shown in figure 6.3 and conclude

$$\hat{\tau}_{ce,ebb} = 0.08 \quad \text{and} \quad \hat{\tau}_{ce,flood} = 0.17 \tag{6.8}$$

$$\hat{\tau}_{cd,ebb} = 0.06 \quad \text{and} \quad \hat{\tau}_{cd,flood} = 0.08 \tag{6.9}$$

Notice that the figure of $\hat{\tau}_{ce}$ shows a typical oscillation. The oscillation lasts about four and a half hours. It is repeated about every twelve hours. That is exactly the time of one period of ebb-tide and flood-tide. Comparing this figure with the tidal data, we may notice that the oscillation takes place during ebb-tide. It starts a little before the LW-turn.

Further we notice that the estimated values for the critical be shear stresses of erosion and sedimentation, $\hat{\tau}_{ce}$ and $\hat{\tau}_{cd}$, are indeed depending on the daily tide. We also notice that $\tau_{ce,ebb} \ll \tau_{ce,flood}$. we may explain this by the fact that there are big mud sources upstream from Bath, while in the downstream direction, the mud sources are much less.

Remark: We obtain

$$\left.\begin{array}{lcl} M_{ebb} & > & M_{flood} \\ \tau_{ce,ebb} & < & \tau_{ce,flood} \end{array}\right\} \Rightarrow \frac{\partial E}{\partial t}(\text{ebb}) > \frac{\partial E}{\partial t}(\text{flood})$$

$$\left.\begin{array}{lcl} W_{s,ebb} & < & W_{s,flood} \\ \tau_{cd,ebb} & < & \tau_{cd,flood} \end{array}\right\} \Rightarrow \frac{\partial d}{\partial t}(\text{ebb}) < \frac{\partial d}{\partial t}(\text{flood})$$

by using Eqns (3.3) and (3.5) of chapter 3:

$$\begin{aligned} \frac{\partial E}{\partial t} &= M\,(\frac{\tau_b}{\tau_{ce}} - 1), & \tau_b \geq \tau_{ce} \\ \frac{\partial d}{\partial t} &= C\,W_s(1 - \frac{\tau_b}{\tau_{cd}}), & \tau_b \leq \tau_{cd} \end{aligned}$$

Knowing that (Eqn (3.2)),

$$\frac{\partial C}{\partial t} = \frac{S}{H} = \frac{\partial E}{\partial t} - \frac{\partial d}{\partial t}$$

we have

$$\frac{\partial C}{\partial t}(\text{ebb}) > \frac{\partial C}{\partial t}(\text{flood})$$

This corresponds with figure 2.2 in chapter 2, where we found that the cohesive sediment concentration during ebb-tide is higher than the concentration during flood-tide.

6.4.3 Obtained model

We substituted the estimated parameters into the extended model. In figure 6.4 we compare the cohesive sediment concentrations obtained by the model, with the observed cohesive sediment concentrations. Notice that the calculated concentrations are too high, in comparison with the observed cohesive sediment concentrations. This result does not satisfy us; something is wrong. Maybe, we made the wrong assumptions, or the model is not right at all, or the Kalman filtering method we used is not working well.

The Kalman filtering method will be adapted in the next chapters. But first, we shall check the model we used.

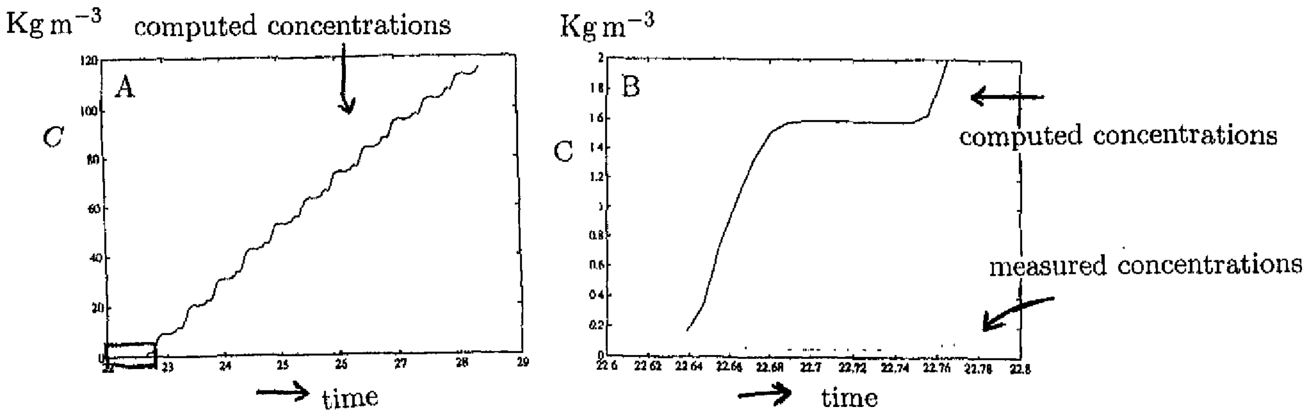


Figure 6.4: The model compared with the observations using the extended model, over a period of six days

A. for November 22 until November 29, 1988 at Bath.
B. for November 22, 1988 at Bath (an enlarged part of A)

where

$$\begin{array}{llll} \hat{M}_{ebb} &= 2.0 \cdot 10^{-3} & \text{and} & \hat{M}_{flood} = 1.0 \cdot 10^{-3} \\ \hat{W}_{s,ebb} &= 3.0 \cdot 10^{-5} & \text{and} & \hat{W}_{s,flood} = 3.5 \cdot 10^{-5} \\ \hat{\tau}_{ce,ebb} &= 0.08 & \text{and} & \hat{\tau}_{ce,flood} = 0.17 \\ \hat{\tau}_{cd,ebb} &= 0.06 & \text{and} & \hat{\tau}_{cd,flood} = 0.08 \end{array}$$

In order to check the model, let us substitute some arbitrary values for the unknown parameters into the model:

$$\begin{array}{lllllll} \tau_{ce,ebb} & = & 0.14 & ; & M_{ebb} & = & 0.5 \cdot 10^{-4} \\ \tau_{ce,flood} & = & 0.20 & ; & M_{flood} & = & 1.0 \cdot 10^{-4} \\ \tau_{cd,ebb} & = & 0.06 & ; & W_{s_{ebb}} & = & 1.0 \cdot 10^{-2} \\ \tau_{cd,flood} & = & 0.08 & ; & W_{s_{flood}} & = & 1.0 \cdot 10^{-2} \end{array}$$

Substituting these values for the eight unknown parameters, we find an almost perfect concentration figure. We use the MATLAB-programs 'rgeg' and 'model' (see appendix C). The concentrations obtained by the model are compared with the observed cohesive sediment concentrations in figure 6.5.

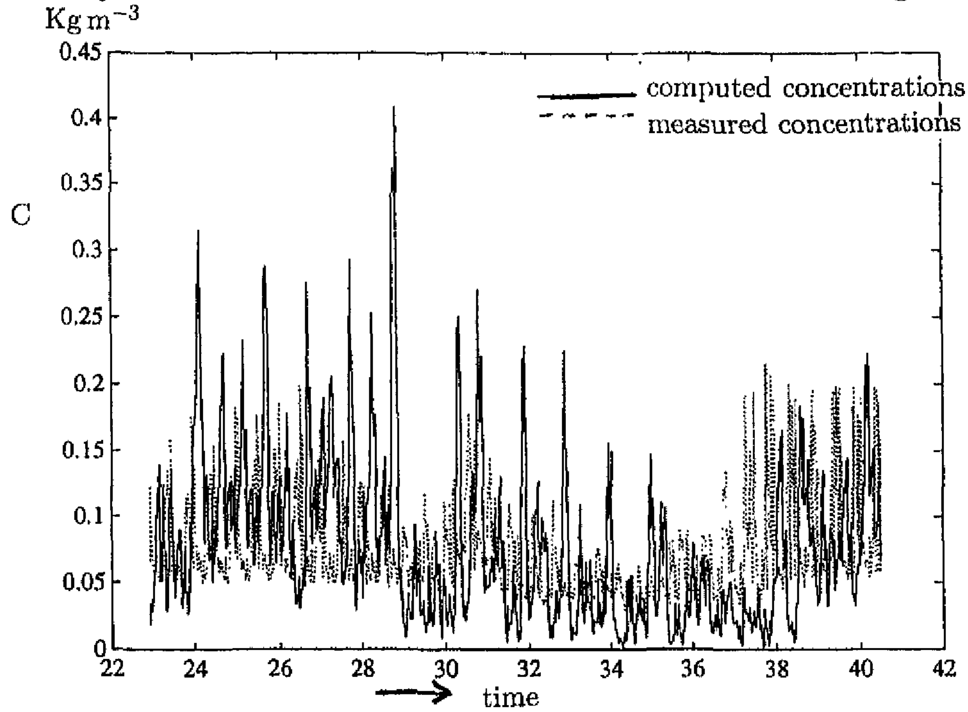


Figure 6.5: The model compared with the observations, using arbitrary values for the eight unknown parameters, over a period of eighteen days

where	*22 means*	*November*	*22,*	*1988*
	29 means	*November*	*29,*	*1988*
	30 means	*November*	*30,*	*1988*
	31 means	*December*	*1,*	*1988*
	32 means	*December*	*2,*	*1988*

and so on.

Figure 6.5 looks perfect. The model has about the same peaks as the real figure, only the peaks have to be a little smaller and have to shift a little. Notice, we are looking at a time period of eighteen days, i.e. from November 22 until December 10, 1988, so we are looking at a time period including a spring-tide/neap-tide period, according to table 6.1. In this kind of period, the total cohesive sediment concentration is fluctuating in time. From the figure we note that the model is following the real figure in its fluctuations. From this we may conclude that our simplified model, is not that bad at all. But, we have to keep in mind, that the estimated values we used in this figure, are not estimated at all, but chosen in an arbitrary way. So the used values of the parameters are not tuned to each other.

We have to try to improve our filtering procedures, to obtain more reliable results. What we may learn from this figure, is that maybe our estimation procedure is not correct. In the next chapters, we are going to examine the noise statistics of the used extended Kalman filter.

6.5 Conclusions

In this chapter we examined the behaviour of the bed shear stress $\tau_b(k)$. We considered two kinds of time shifts: a static time shift and a dynamic time shift. We found the following time shifts,

Static time shift:

$$\tau_b(k) \;:=\; \tau_b(k - 15)$$

Dynamic time shift:

$$\begin{aligned}\tau_{b,ebb}(k) &= \tau_b(k - \frac{1}{0.008380}\frac{0.13}{V_k})\\ \tau_{b,flood}(k) &= \tau_b(k - \frac{1}{0.008380}\frac{0.03}{V_k})\end{aligned}$$

The averaged time delay of the bed shear stress $\tau_b(k)$ is about 3 hours. This is acceptable, considering the average velocity of the water, and the distance between Bath and the mud sources.

We decided to distinguish between the ebb-stream and the flood-stream, because these are total different situations. So we split up our model into two parts: one part for the case of erosion, and one part for the case of sedimentation. Now we have eight unknown parameters, instead of four, i.e. two different values for each of them.

We split the model and used the bed shear stress $\tau_b(k)$ with a static time shift, we estimated the eight unknown parameters. We obtained

$$\begin{array}{lll}\hat{M}_{ebb} = 2.0 \cdot 10^{-3} & \text{and} & \hat{M}_{flood} = 1.0 \cdot 10^{-3}\\ \hat{W}_{s,ebb} = 3.0 \cdot 10^{-5} & \text{and} & \hat{W}_{s,flood} = 3.5 \cdot 10^{-5}\\ \hat{\tau}_{ce,ebb} = 0.08 & \text{and} & \hat{\tau}_{ce,flood} = 0.17\\ \hat{\tau}_{cd,ebb} = 0.06 & \text{and} & \hat{\tau}_{cd,flood} = 0.08\end{array}$$

Indeed, the values are different in the different cases of ebb and flood. At first sight the estimated values look right, but when we examined the cohesive sediment concentrations obtained by the model, we noticed that these concentrations were too high.

Because we had a notion of the values of the eight unknown parameters, we substituted some arbitrary values into the model, and compared these cohesive sediment concentrations with the observed cohesive sediment concentrations. These two figures had the same structure. We found the same peaks, just a little shifted in time, and some of the peaks were a little to high, but the shape of the peaks looked the same. This is a result of great value. Since the model looks right using some arbitrary values, we know that the model is not wrong at all. So the wrong estimates may be caused by the way we applied the Kalman filtering method. In the next chapters, we shall examine the Kalman filtering method.

For a well operating extended Kalman filter, i.e. an extended Kalman filter which generates reliable estimates, it is necessary to have an accurate knowledge of the noise statistics (Mous, 1994). Until this far, we made an assumption about these values, but in the next chapter, we are going to examine a way to identify these noise statistics.

Remark: The working of the extended Kalman filtering method in this problem could also be examined by testing the used Kalman filter with simulated observations. With this test we could determine whether or not the Kalman filter was working well. (This test would be better than the test we used, by examining the model with some arbitrary chosen values.) The test with the simulated observations is desirable, because the theory about the extended Kalman filter is not sufficient to predict a reliable working of the extended Kalman filter.

Chapter 7

Identification of the noise statistics

7.1 Introduction

In the previous chapters we determined the unknown parameters of a model describing the tidal effects on the cohesive sediment concentration in the Western Scheldt. For the estimation we used the (extended) Kalman filtering method. Because we are not satisfied about the obtained values by these estimations, we are going to examine the Kalman filtering method in this chapter. We will concentrate on the noise statistics, especially on the covariance matrix of the system noise.

We explain why the system noise may be important for the estimation of the unknown parameters and we give two methods to identify this system noise covariance matrix. The first method is the Maximum Likelihood method, the second is a method suggested by Mehra in 1970. After the explanation of these two methods, we will show how we may apply the identification of the system noise covariance matrix Q to our estimation problem.

In the next chapter we will identify Q. Using the identified Q we will obtain new estimates for the unknown parameters.

7.2 The system noise covariance matrix Q

We know that when $(A \sim FQ^{\frac{1}{2}})$ stabilizable and $(C \sim A)$ detectable, then P_k converges to a limiting value $\bar{P}$ as $k \to \infty$, independent of $P(0)$. This is applicable to our problem, because $(A_k \sim FQ^{\frac{1}{2}})$ is controllable for all k, and $(C \sim A_k)$ is observable for all k. The limiting value $\bar{P}$ is independent of the initial value $P(0)$, but it is influenced by the system noise covariance matrix Q as may be understood from the Kalman filter equations,

$$\begin{aligned} P_{k+1|k} &= A_k P_k A_k^T + F_k Q(k) F_k^T \\ K_{k+1} &\triangleq P_{k+1|k} C_{k+1}^T [C_{k+1} P_{k+1|k} C_{k+1}^T + R(k+1)]^{-1} \\ \hat{X}_{k+1} &= \hat{X}_{k+1|k} + K_{k+1}(Y_{k+1} - C_{k+1}\hat{X}_{k+1}) \end{aligned}$$

When Q is not correct, it may cause an error in the limiting value $\bar{P}$ of P_k. This will cause an error in the estimated value of the state $\hat{X}_k$.

We need a reliable approximation of the system noise covariance matrix Q, to obtain useful estimates. We shall identify it from the data in the next chapter, using both the method of Maximum Likelihood and the method suggested by Mehra. These methods will be described in the next sections.

7.3 Two methods to identify Q

7.3.1 Maximum Likelihood method

Let us denote the unknown parameter vector by θ (in this case $\theta = Q$). Consider the following linear time invariant discrete-time stochastic dynamical system:

$$X_{k+1} = AX_k + Bu(k) + FW_k \tag{7.1}$$
$$Y_k = CX_k + V_k \qquad , k \geq 0 \tag{7.2}$$

where the state X_k is an n-dimensional vector, the system disturbance W_k is an r-dimensional vector, the observation Y_k is an m-dimensional vector, V_k is the measurement disturbance, and $\{u(k)\}$ is a known p-dimensional vector input sequence. $\{W_k\}$ and $\{V_k\}$ are independent sequences of independent, zero mean Gaussian random vectors with covariances Q and R, respectively. Suppose that the matrices A, B, F, C, Q and R are only partially known and θ is the vector of unknown parameters in those matrices. We assume that θ belongs to a compact (closed and bounded) parameter space $\Theta \subset \mathbb{R}^d$. The parameter estimation problem is then, to find an estimate of θ, based on the observations $Y_0 = y_0, \cdots, Y_N = y_N$, for some fixed N. We denote such an estimate by $\hat{\theta}_N(y_0, \cdots, y_N)$.

Let us introduce

$$Z_{k+1}(\theta) = Y_{k+1} - CBu(k) - CA\hat{X}_k(\theta) \tag{7.3}$$

Using the Kalman filter equations it is easily to see that $\hat{X}_k$ is a function of $\mathbf{Y}_k$ and $\mathbf{u}(k-1)$ and we may write

$$Y_{k+1} = g(k, \mathbf{Y}_k, \mathbf{u}(k); \theta) + Z_{k+1}(\theta) \tag{7.4}$$

where $\mathbf{Y}_k$ denotes the vector $(Y_k^T, Y_{k-1}^T, \cdots)^T$ and $\mathbf{u}(k)$ denotes the vector $(u(k)^T, u(k-1)^T, \cdots)^T$ and the innovation process Z_k is white noise.

The likelihood function of our problem is obtained by calculating the probability density function of $Y_0, \cdots, Y_N$. Using Bayes' rule we can always write

$$f_{\mathbf{Y}_N|\mathbf{u}(N-1),\theta}(\mathbf{y}_N) = f_{Y_0}(y_0) \prod_{k=1}^{N} f_{Y_k|\mathbf{Y}_{k-1},\mathbf{u}(k-1),\theta}(\mathbf{y}_k \mid y_{k-1}) \tag{7.5}$$

where $f_{Y_k|\mathbf{Y}_{k-1},\mathbf{u}(k-1),\theta}(\mathbf{y}_k \mid y_{k-1})$ is the conditional probability density of Y_k given $\mathbf{Y}_{k-1}$, $\mathbf{u}(k-1)$ and θ. Using Eqn (7.4), we may relate the conditional probability density of Y_k to that of Z_k as follows

$$f_{Y_k|\mathbf{Y}_{k-1},\mathbf{u}}(k-1), \theta(y_k \mid \mathbf{y}_{k-1}) = f_{z_k}(z_k(\theta) \mid \theta) \mid det \left[\frac{\partial z_k}{\partial y_k}\right] \mid \tag{7.6}$$

From Eqn (7.4), the Jacobian $\mid det \left[\frac{\partial Z_k}{\partial Y_k}\right] \mid = 1$, and $z_k(\theta)$ is a realization of $Z_k(\theta)$ given by

$$z_k(\theta) = y_k - g(k-1, \mathbf{y}_{k-1}, \mathbf{u}(k-1); \theta) \tag{7.7}$$

From results in Kalman filtering, we know that $Z_k(\theta)$, for true parameter θ, is Gaussian white noise with zero mean and covariance

$$\mathcal{H}(k;\theta) = C(AP(k-1;\theta)A^T + FQF^T)C^T + R \tag{7.8}$$

Therefore,

$$\begin{aligned} f_{\mathbf{Y}_N|\mathbf{u}(N-1),\theta}(\mathbf{y}_N) &= f_{Y_0}(y_0) \prod_{k=1}^{N} [(2\pi)^m det[\mathcal{H}(k:\theta)]^{-\frac{1}{2}} \cdot \\ &\quad exp(-\frac{1}{2} z_k(\theta)^T \mathcal{H}(k;\theta)^{-1} z_k(\theta)) \end{aligned}$$

$$
\begin{aligned}
&= f_{Y_0}(y_0) \prod_{k=1}^{N} [(2\pi)^m det[\mathcal{H}(k;\theta)]^{-\frac{1}{2}} \cdot \\
&\quad exp(-\frac{1}{2}\sum_{k=1}^{N} z_k(\theta)^T \mathcal{H}(k;\theta)^{-1} z_k(\theta)) \qquad (7.9)
\end{aligned}
$$

The likelihood function is obtained from Eqn (7.9) by substituting $\mathbf{Y}_N$ in place of the actual observation $\mathbf{y}_N$. The likelihood function for our problem is then

$$
\begin{aligned}
L(\mathbf{Y}_N;\theta) &\triangleq f_{\mathbf{Y}_N|\mathbf{u}(N-1)}(\mathbf{Y}_N) \\
&= f_{Y_0}(Y_0) \prod_{k=1}^{N} [(2\pi)^m det[\mathcal{H}(k;\theta)]^{-\frac{1}{2}} \cdot \\
&\quad exp\left(-\frac{1}{2}\sum_{k=1}^{N} Z_k(\theta^T)\mathcal{H}(k;\theta)^{-1} Z_k(\theta)\right) \qquad (7.10)
\end{aligned}
$$

where

$$
\begin{aligned}
Z_k(\theta) &= Y_k - g(k-1, \mathbf{Y}_{k-1}, \mathbf{u}(k-1);\theta) \\
&= Y_k - CA\hat{X}_{k-1} - CBu(k-1) \qquad (7.11)
\end{aligned}
$$

A maximum likelihood estimator of θ, denoted $\hat{\theta}_N$, is that value of θ for which $L(\mathbf{Y}_N;\theta)$ is a maximum; equivalently, the value of θ for which

$$
\tilde{L} = \sum_{k=1}^{N} [log(det[\mathcal{H}(k;\theta)])] + \frac{1}{2}\sum_{k=1}^{N} \left(Z_k^T(\theta)\mathcal{H}(k;\theta)^{-1} Z_k(\theta)\right) \qquad (7.12)
$$

has a minimum.

We will use this $\tilde{L}(\mathbf{Y}_N, \theta)$ in the estimation described in the next section. The Maximum Likelihood method may also be used for the estimation of other unknown parameters of an arbitrary model.

Remark: We use here the Kalman filtering method for the estimation of the unknown parameters of the cohesive sediment concentration model, instead of the Maximum Likelihood method, because the Kalman filtering method is recursive, and the Maximum Likelihood method is not. Since we have a lot of observations, (i.e. in about 20 days, we have more than 2000 observations), we prefer the recursive method.

First we examine the other identification method, the method suggested by Mehra (1970).

7.3.2 Method of Mehra

Consider a multivariable linear discrete system

$$\begin{aligned} X_{k+1} &= AX_k + FW_k && (7.13) \\ Y_k &= CX_k + V_k && (7.14) \end{aligned}$$

with X_k an $n \times 1$ state vector, A an $n \times n$ non singular transition matrix, F an $n \times q$ constant input matrix, Y_k an $r \times 1$ measurement vector and C an $r \times n$ constant output matrix. The sequences W_k $(q \times 1)$ and V_k $(r \times 1)$ are uncorrelated Gaussian white noise sequences with means and covariances

$$\begin{aligned} E[W_k] &= 0; && E[W_k W_j^T] = Q\delta_{ij} && (7.15) \\ E[V_k] &= 0; && E[V_k V_j^T] = R\delta_{ij} && (7.16) \\ E[W_k V_j^T] &= 0, && \text{for all } i, j. && (7.17) \end{aligned}$$

where δ_{ij} denotes the Kronecker delta function.
Q is a bounded positive definite matrix ($Q > 0$) and the initial state X_0 is normally distributed with zero mean and covariance P_0.
It is assumed that the system is time invariant, completely controllable and observable. Both the system and the filter (optimal or suboptimal) are assumed to have reached steady-state conditions.

Let Q_0 represent the initial estimate of Q ($Q_0 > 0$), K_0 the initial estimate of the steady state Kalman filter gain ($n \times r$ matrix).

$$\begin{aligned} K_0 &= P_0 C^T (C P_0 C^T + R_0)^{-1} && (7.18) \\ P_0 &= A[P_0 - P_0 C^T (C P_0 C^T + R_0)^{-1} C P_0) A^T \\ &\quad + F Q_0 F^T && (7.19) \end{aligned}$$

P_0 may be recognized as the steady ations of the Kalman filtering method.

$$\begin{aligned} \hat{X}_{k+1|k} &= A\hat{X}_k && (7.20) \\ \hat{X}_k &= \hat{X}_{k|k-1} + K_0(Y_k - C\hat{X}_{k|k-1}) && (7.21) \end{aligned}$$

The filtering equations are

$$\begin{aligned} \hat{X}_{k+1|k} &= A\hat{X}_k && (7.20) \\ \hat{X}_k &= \hat{X}_{k|k-1} + K_0(Y_k - C\hat{X}_{k|k-1}) && (7.21) \end{aligned}$$

where $\hat{X}_{k+1|k}$ is the estimate of X_{k+1} based on all measurements up to k

where $\hat{X}_{k+1|k}$ is the estimate of X_{k+1} based on all measurements up to k, i.e. $Y_0, \cdots, Y_k$.

In an optimal Kalman filter (i.e. when $Q_0 = Q$), we have P_0 the covariance of the error in estimating the state. But in a suboptimal case, the covariance of the error (P) is given by the following equation (Mehra, 1970):

$$\begin{aligned} P &= Q[P - K_0CP - PC^TK_0^T + K_0(CPC^T + R)K_0^T]A^T \\ &\quad + FQF^T \end{aligned} \tag{7.22}$$

where $P = E[(X_k - \hat{X}_{k|k-1})(X_k - \hat{X}_{k|k-1})$
Rewriting Eqn (7.22)

$$\begin{aligned} P &= A(I - K_0C)P(I - K_0C)^TA^T \\ &\quad + AK_0RK_0^TA^T + FQF^T \end{aligned} \tag{7.23}$$

We need the true value of Q. This value is unknown. To obtain an estimate of Q, we have to check whether the Kalman filter constructed using an initial estimate of Q is close to optimal or not (hypothesis testing). In case it is suboptimal, we obtain a new estimate of Q using the autocorrelation function of the innovation process.

To check whether the Kalman filter is doing well we use

Statement 1 For an optimal filter, the sequence $Z_k = (Y_k - C\hat{X}_{k|k-1})$, known as the innovation sequence, is a Gaussian white noise sequence.

proof:
See Mehra, 1970 [16].

To test the optimality of the Kalman filter, we need the following statement

Statement 2 Let K denote the steady-state filter gain. Under steady state, the innovation sequence Z_k is a stationary Gaussian sequence:

$$Z_k = Y_k - C\hat{X}_{k|k-1}$$

proof:
Mehra [16].

Necessary and sufficient condition for the optimality of a Kalman filter is that the innovation sequence Z_k is white.

A simple test for the whiteness of the innovation sequence is based on the autocorrelation functions. Define

$$r_i \equiv E[Z_k Z_{k-i}^T] \tag{7.24}$$

Then

$$\begin{aligned} r_i &= CPC^T + R & ,i = 0 \quad (7.25)\\ &= C[A(I - KC)]^{i-1}A[PC^T - Kr_0] & ,i > 0 \quad (7.26) \end{aligned}$$

Furthermore

$$r_{-i} = r_i^T \tag{7.27}$$

We obtain an estimate of r_i, denoted as $\hat{r}_i$, by using the ergodic property of a stationary random sequence

$$\hat{r}_i = \frac{1}{N}\sum_{k=i}^{N} Z_k Z_{k-i}^T \tag{7.28}$$

where N is the number of sample points.

Estimates of the normalized autocorrelation coefficients ρ_i are obtained by dividing the elements of $\hat{r}_i$ by the appropriate elements of $\hat{r}_0$, e.g.,

$$[\hat{\rho}_i]_{kl} = \frac{[\hat{r}_i]_{kl}}{([\hat{r}_0]_{kk}[\hat{r}_0]_{ll})^{\frac{1}{2}}} \tag{7.29}$$

Using Kendall and Stuart (1976, [9]) and writing $\hat{\rho}_i = \frac{\hat{r}_i}{\hat{r}_0}$

$$\begin{aligned} E\hat{\rho}_i &= \frac{E\hat{r}_i}{\hat{r}_0} - \frac{\hat{r}_i E\hat{r}_0}{\hat{r}_0^2} & (7.30)\\ var\,\hat{\rho}_i &= \frac{var\,\hat{r}_i}{\hat{r}_0^2} - \frac{2\hat{r}_i cov(\hat{r}_i,\hat{r}_0)}{\hat{r}_0^3 + \frac{\hat{r}_i^2 var(\hat{r}_0)}{\hat{r}_0^4}} & (7.31) \end{aligned}$$

For the white noise case $r_i = 0$ for all $i \neq 0$ (Mehra [16])

$$\begin{aligned} \text{and } cov(\hat{r}_i,\hat{r}_j) &= 0 & ,i \neq j\\ &= \frac{1}{N}\hat{r}_0^2 & ,i = j > 0\\ &= \frac{1}{N}\hat{r}_0^2 + \hat{r}_0^2 & ,i = j = 0 \quad (7.32) \end{aligned}$$

w.l.o.g. $r_0 = \hat{r}_0 = 1$ and it is easy to see that

$$E\hat{\rho}_i = \rho_i - O(\frac{1}{N}) = 0 \text{ in white noise case} \tag{7.33}$$

$$var\,\hat{\rho}_i = \frac{1}{N} + O(\frac{1}{N^2}) \tag{7.34}$$

$\hat{\rho}_i$ is asymptotically normal

$$\frac{\hat{\rho}_i - E[\hat{\rho}_i]}{\sqrt{var\,\hat{\rho}_i}} \cong N(0,1)$$

$$\Rightarrow \frac{\hat{\rho}_i}{\sqrt{\frac{1}{N}}} \cong N(0,1) \tag{7.35}$$

Therefore the 95 percent confidence limits for $\hat{\rho}_i, i > 0$ are $\pm 1.96/N^{\frac{1}{2}}$ (for large N).

Test
Look at a set of values for $\hat{\rho}_i, i > 0$ and check the number of times they lie outside the band $(\pm 1.96/N^{\frac{1}{2}})$. If this number is less than 5 percent of the total, the sequence Z_k is white.

Estimation of Q
If the test just described reveals that the filter is suboptimal, the next step will be to obtain a better estimate of Q. This may be done by using C_k, defined earlier. We will show it for the situation $n = 1$. For the general case $(n \neq 1)$ see Mehra [16].
We need[1] (Anderson and Moore, 1979)

Definition 1 Let Φ be an $n \times n$ matrix. Its *Pseudo-inverse* $\Phi^{\#}$ is uniquely defined by the following equations:

$$\Phi^{\#}\Phi x = x \qquad \forall\, x\, \epsilon \mathcal{R}[\Phi^T] = \mathcal{N}[\Phi]^{\perp}$$
$$\Phi^{\#} x = 0 \qquad \forall\, x\, \epsilon\, \mathcal{R}[\Phi]^{\perp} = \mathcal{N}[\Phi^T]$$

Observe that $\Phi^{\#}\Phi$ is the identity on $\mathcal{R}[\Phi^T] = \mathcal{N}[\Phi]^{\perp}$

Where $R[\Phi]$, the range space of Φ is the set of all vectors Φx, where x ranges over the set of all n-vectors. Its dimension is equal to the rank of Φ.

[1] There are actually a number of different pseudo-inverses. Here we describe the Moore-Penrose pseudo-inverse

$\mathcal{N}[\Phi]$, the null space of Φ, is the set of vectors y for which $\Phi y = 0$.

First we want to obtain an estimate of PC^T. We use Eqn (7.26)

$$r_1 = CAPC^T - CAKr_0 \tag{7.36}$$

Denoting by $\hat{P}\hat{C}^T$ the estimate of PC^T we may write

$$\hat{P}\hat{C}^T = K\hat{r}_0 + \Phi^{\#}\hat{r}_1 \tag{7.37}$$

where $\Phi^{\#}$ is the pseudo-inverse of matrix Φ which is defined as

$$\Phi = CA \tag{7.38}$$

To obtain an estimate of Q, we use Eqn (7.23).
write

$$P = APA^T + \Omega + FQF^T \tag{7.39}$$

$$\text{where } \Omega = A(-K_0CP - PC^TK_0^T + K_0C_0K_0^T)A^T \tag{7.40}$$

Premultiplying both sides of Eqn (7.39) by C and post multiplying by $(A^{-k})^TC^T$, and substituting the estimated values, we obtain

$$CF\hat{Q}F(A^{-1})^TC^T = \hat{C}\hat{P}(A^{-1})^TC^T - CA\hat{P}\hat{C}^T - C\hat{\Omega}(A^{-1})C^T \tag{7.41}$$

where

$$\hat{\Omega} = A[K_0\hat{C}\hat{P} - \hat{P}\hat{C}^TK_0^T + K_0\hat{r}_0K_0^T]A^T \tag{7.42}$$

To apply the method of Mehra, we divide the observations in, for instance, 5 iteration steps i (batches) of N points. Of every batch we determine the estimate of Q in the following way

$$\hat{Q}_{i+1} = \hat{Q}_i + \left(\frac{i}{i+1}\right)(\hat{Q}_{i+1|i} - \hat{Q}_i) \tag{7.43}$$

where

$\hat{Q}_i$	:	the estimate of Q after i batches
$\hat{Q}_{i+1\|i}$	:	the estimate of Q based on the $(i+1)^{th}$ batch
$\hat{Q}_{i+1}$	:	the estimate of Q after $(i+1)$ batches

After each iteration step we verify if the calculated mean square error $tr(P_c)$ (where P_c is calculated from the variance equation, using $\hat{Q}$) does not differ much from the estimate of the actual mean square error

$$\frac{1}{N}\sum_{k=1}^{N}(X_k - \hat{X}_{k|k-1})^T(X_k - \hat{X}_{k|k-1})$$

7.4 Application of the noise identification

We want to estimate the eight unknown parameters of the mathematical model, describing the tidal effects on the cohesive sediment concentration in the Western Scheldt. We use the (extended) Kalman filtering method. Because we know nothing about the system noise covariance matrix, we are going to identify this matrix, for all eight different models, one for each unknown parameter. We assume

$$Q_i = \begin{bmatrix} Q_{11}(i) & 0 \\ 0 & Q_{22}(i) \end{bmatrix} \qquad i = 1 \cdots 8 \tag{7.44}$$

where $Q_{11}(i)$ and $Q_{22}(i)$ are not depending on time k.

We shall use the Maximum Likelihood method to obtain rough estimates of $Q_{11}(i)$ and $Q_{22}(i)$, $i = 1 \cdots 8$. These rough estimates will be used as initial values to estimate $Q_{11}(i)$, $i = 1 \cdots 8$ by the method of Mehra, and $Q_{22}(i)$, $i = 1 \cdots 8$ by the Maximum Likelihood method.

Using the obtained system noise covariance matrices Q_i, we are going to estimate the eight unknown parameters θ_i with the (extended) Kalman filter in the next chapter.

7.5 Conclusions

In the previous chapters, we estimated the unknown parameters of the mathematical model, which describes the tidal influences on the cohesive sediment transport in the Western Scheldt. We used the extended Kalman filtering method. Since the obtained estimates are not reliable, we decided to examine the Kalman filtering method we used, especially the noise statistics of this method.

In this chapter we examined two methods to identify the system noise covariance matrix Q. The first method is the Maximum Likelihood method, the second method, we examined, is a method suggested by Mehra in 1970. Both methods may be applied to our problem, provided that we make some assumptions.

The Maximum Likelihood method determines a likelihood function $L(\mathbf{Y}_N;\theta)$, for the unknown parameter θ and observations $Y_0, Y_1, \cdots Y_N$. With this likelihood function it is possible to determine the maximum likelihood estimator of θ, denoted by $\hat{\theta}_N$. This is the value of θ for which $L(\mathbf{Y}_N; \theta)$ has a maximum.

The method of Mehra checks whether the Kalman filter constructed using an initial estimate of the system noise covariance matrix Q is close to optimal or not. If it is suboptimal the method obtains a new estimate of Q using the autocorrelation function of the innovation process.

We will use both methods in the next chapter to identify the eight unknown system noise covariance matrices Q_i, $i = 1 \cdots 8$. Each Q_i belongs to a system which determines the estimate of an unknown parameter of the cohesive sediment concentration model. We will assume

$$Q_i = \begin{bmatrix} Q_{11}(i) & 0 \\ 0 & Q_{22}(i) \end{bmatrix} \qquad 1 = 1 \cdots 8$$

where $Q_{11}(i)$ and $Q_{22}(i)$ are not depending on time k, to keep the problem simple. We will use the method of Mehra to estimate $Q_{11}(i)$, $i = 1 \cdots 8$, and the Maximum Likelihood method to estimate $Q_{22}(i)$, $i = 1 \cdots 8$.

Chapter 8

Parameter and system noise identification

8.1 Introduction

We have obtained estimates for the eight unknown parameters of the cohesive sediment concentration model. Since we are not satisfied about the estimated values, we decided to examine the Kalman filtering method, we used for the estimation. We examined the system noise covariance matrix Q in the previous chapter. We gave two methods to identify this matrix; the Maximum Likelihood method and a method suggested by Mehra. In this section we will use both methods to identify the system noise covariance matrix $Q(i),\ \ i = 1 \cdots 8$. Knowing these covariance matrices $Q(i)$, we estimate the unknown parameters $\theta_i,\ \ i = 1 \cdots 8$, again.

We start to identify the process noise covariance matrices Q needed for the estimation of the erosion constant M and the settling velocity W_s, in the cases of ebb and flood. Using these system noise covariance matrices we obtain estimates for M and W_s, using the (extended) Kalman filtering method. After this we identify the system noise covariance matrices Q needed for the estimation of the critical bed shear stresses for erosion and sedimentation, τ_{ce} and τ_{cd}, in the cases of ebb and flood. We obtain estimates for τ_{ce} and τ_{cd} using the extended Kalman filter with the identified Q. Finally, we will substitute the obtained estimates in the cohesive sediment concentration model and compare the computed concentrations with the observed concentrations.

8.2 Identification of the system noise covariance matrix Q

We will first identify the system noise covariance matrices $Q(i)$, $i = 1 \cdot\cdot 4$, i.e. the Q-matrices needed for the estimation of the erosion constants M_{ebb} and M_{flood}, and for the settling velocities $W_{s,ebb}$ and $W_{s,flood}$. We first obtain rough estimations of $Q(i)$, using the Maximum Likelihood method. Using these rough estimates as initial values, we will identify $Q_{11}(i)$ with the method of Mehra, and $Q_{22}(i)$ with the Maximum Likelihood method, in later sections.

We use as initial error covariance matrix $P(0)$

$$P_{0|-1} = \bar{P}(0) = \begin{bmatrix} R & 0 \\ 0 & 1.0 \end{bmatrix} \tag{8.1}$$

where

$$\begin{aligned} P_{0|-1} &= E(\tilde{X}_{0|-1}\tilde{X}_{0|-1}) \\ &= E(X_0 - \hat{X}_{0|-1})(X_0 - \hat{X}_{0|-1}) \\ &= E(X_0 - \bar{x}(0))(X_0 - \bar{x}(0)) \\ &= \begin{bmatrix} E(C_0 - \bar{C}(0))(C_0 - \bar{C}(0)) & E(C_0 - \bar{C}(0))(\theta_0 - \bar{\theta}(0)) \\ E(\theta_0 - \bar{\theta}(0))(C_0 - \bar{C}(0)) & E(\theta_0 - \bar{\theta}(0))(\theta_0 - \bar{\theta}(0)) \end{bmatrix} \end{aligned}$$

Thus we assume

$$E(C_0 - \bar{C}(0))(C_0 - \bar{C}(0)) = R \tag{8.2}$$

since we have $Y_k = C_k$. Furthermore

$$\begin{aligned} & E(C_0 - \bar{C}(0))(\theta_0 - \bar{\theta}(0)) \\ = \; & E(\theta_0 - \bar{\theta}(0))(C_0 - \bar{C}(0)) \\ = \; & 0 \end{aligned} \tag{8.3}$$

because we assume the initial values of C and θ are independent. Finally

$$E(\theta_0 - \bar{\theta}(0))(\theta_0 - \bar{\theta}(0)) = 1.0 \tag{8.4}$$

Where this large variance stands for the initial uncertainty in the parameter θ (Mous, 1994).

But, of course, as already mentioned in section 7.2, this error covariance matrix $P(0)$ does not need to be very important, because, when $(A \sim FQ^{\frac{1}{2}})$

stabilizable and ($C \sim A$) detectable, then $P(k)$ converges to a limiting value $\bar{P}$ as $k \to \infty$, independent of $P(0)$. This is applicable to our problem, because ($A \sim FQ^{\frac{1}{2}}$) is controllable, and ($C \sim A$) is observable. So the estimates are independent on $P(0)$. But since the model we used is a simplified model, and we make a lot of assumptions, we decide to choose the initial error covariance matrix $P(0)$ in a way in which it will give no troubles.

Rough estimation of Q

Method

We assume the shape of Q

$$Q(i) = \begin{bmatrix} Q_{11}(i) & 0 \\ 0 & Q_{22}(i) \end{bmatrix} \tag{8.5}$$

where $i = 1 \cdot\cdot 4$, i.e. one Q_i for every unknown parameter: M_{ebb}, M_{flood}, $W_{s,ebb}$, $W_{s,flood}$.

We start identifying Q_{11}, using the Maximum Likelihood method. We use the MATLAB-programs 'ri11' and 'qu11' (appendix C). We use the (initial) values

$$M_{ebb}(0) = 5 \cdot 10^{-5} \quad ; \quad M_{flood}(0) = 1 \cdot 10^{-4} \tag{8.6}$$

$$W_{s,ebb}(0) = 1 \cdot 10^{-2} \quad ; \quad W_{s,flood}(0) = 1 \cdot 10^{-2} \tag{8.7}$$

$$\tau_{ce,ebb} = 0.14 \quad ; \quad \tau_{ce,flood} = 0.20 \tag{8.8}$$

$$\tau_{cd,ebb} = 0.06 \quad ; \quad \tau_{cd,flood} = 0.08 \tag{8.9}$$

$$F_k = I_2 \tag{8.10}$$

$$R(k) = 0.7 \cdot 10^{-7} \tag{8.11}$$

$$Q_{22}(i) = 1 \cdot 10^{-5} \qquad i = 1,2,..,4 \tag{8.12}$$

After this we identify Q_{22}, using the Maximum Likelihood method, where we assume $Q_{11}(i)$ as obtained in the previous identification. We use the MATLAB-programs 'ri22' and 'qu22'.

Results
We obtain

$$\hat{Q}(1) = \begin{bmatrix} 1.0 \cdot 10^{-3} & 0 \\ 0 & 1.0 \cdot 10^{-7} \end{bmatrix} \tag{8.13}$$

$$\hat{Q}(2) = \begin{bmatrix} 1.0 \cdot 10^{-4} & 0 \\ 0 & 1.0 \cdot \cdot 10^{-7} \end{bmatrix} \tag{8.14}$$

$$\hat{Q}(3) = \begin{bmatrix} 1.0 \cdot 10^{-3} & 0 \\ 0 & 1.0 \cdot 10^{-5} \end{bmatrix} \tag{8.15}$$

$$\hat{Q}(4) = \begin{bmatrix} 1.0 \cdot 10^{-5} & 0 \\ 0 & 1.0 \cdot 10^{-12} \end{bmatrix} \tag{8.16}$$

Remark: Since the likelihood function has a maximum value, we know that the Kalman filter estimations are indeed influenced by the process noise covariance matrix Q.

Estimation of Q_{11}

Method
We use the method of Mehra to estimate $Q_{11}(i)$, $i = 1 \cdot\cdot 4$. This method uses a linear system

$$X_{k+1} = AX_k + Gu(k) + FW_k \tag{8.17}$$
$$Y_k = CX_k + V_k \tag{8.18}$$

To estimate the unknown erosion constants M_{ebb} and M_{flood}, we have used Eqn (5.8) in chapter 5. Rewriting this equation we obtain

$$C_{k+1} = C_k + \frac{\alpha M_k}{H_k}\left(\frac{\tau_b(k)}{\tau_{ce}} - 1\right) \qquad \tau_b(k) \geq \tau_{ce} \tag{8.19}$$

This may be written as the system

$$C_{k+1} = C_k + \frac{\alpha M}{H_k}\left(\frac{\tau_b(k)}{\tau_{ce}} - 1\right) + FW_k \tag{8.20}$$
$$Y_k = C_k + V_k \tag{8.21}$$

for a certain M. So we assume in Eqns (8.17)–(8.18)

$$u(k) = \frac{\alpha M}{H_k}\left(\frac{\tau_b(k)}{\tau_{ce}} - 1\right)$$

$$\begin{aligned} A &= 1 \\ G &= 1 \\ F &= 1 \\ C &= 1 \end{aligned}$$

Remark: Notice that we use the method of Mehra only to determine Q_{11}, and not to determine Q_{22}, since this method uses linear systems!

To estimate the unknown settling velocities $W_{s,ebb}$ and $W_{s,flood}$, we have used Eqn (5.12) in chapter 5. Rewriting this equation we obtain

$$C_{k+1} = C_k + C_k \frac{\alpha W_s(k)}{H_k} \left(\frac{\tau_b(k)}{\tau_{cd}} - 1 \right) \qquad \tau_b(k) \leq \tau_{cd} \tag{8.22}$$

This may be written as the system

$$\begin{aligned} C_{k+1} &= \left[1 + \frac{\alpha W_s}{H_k} \left(\frac{\tau_b(k)}{\tau_{cd}} - 1 \right) \right] C_k + F W_k \qquad (8.23) \\ Y_k &= C_k + V_k \qquad (8.24) \end{aligned}$$

for a certain W_s. So we assume in Eqns (8.17)–(8.18)

$$\begin{aligned} A &= \frac{1}{N} \sum_{k=1}^{N} [1 + \frac{\alpha W_s}{H_k} (\frac{\tau_b(k)}{\tau_{cd}} - 1)] \\ F &= 1 \\ C &= 1 \\ G &= 0 \end{aligned}$$

To obtain estimates of $Q_{11}(i), \quad i = 1 \cdot\cdot 4$, using the method of Mehra and Eqns (8.20)–(8.21) and (8.23)–(8.24), we need values for the unknown parameters M_{ebb}, M_{flood}, $W_{s,ebb}$ and $W_{s,flood}$. We determine these values by using the (extended) Kalman filter and Eqns (8.13)–(8.16). We assume the initial values of Eqns (8.6)–(8.11). Using the MATLAB programs 'rkalf', 'kalfil' and 'kfmean' we obtain

$$\hat{M}_{ebb} = 5.6 \cdot 10^{-5} \quad \text{and} \quad \hat{M}_{flood} = 2.3 \cdot 10^{-5} \tag{8.25}$$

$$\hat{W}_{s,ebb} = 1.4 \cdot 10^{-3} \quad \text{and} \quad \hat{W}_{s,flood} = 9.5 \cdot 10^{-4} \tag{8.26}$$

We use these values for the erosion constant M and the settling velocity W_s to determine $Q_{11}(i), \quad i = 1 \cdot\cdot 4$ with the method of Mehra. Further, we use the MATLAB-programs 'rmeh1' and 'mehra3' (appendix C). We start with initial values of Q_{11_i}, $i = 1 \cdot\cdot 4$, according to Eqns (8.13)–(8.16):

$$\begin{aligned}
\hat{Q}_{11}(1) &= 1.0 \cdot 10^{-3} \\
\hat{Q}_{11}(2) &= 1.0 \cdot 10^{-4} \\
\hat{Q}_{11}(3) &= 1.0 \cdot 10^{-3} \\
\hat{Q}_{11}(4) &= 1.0 \cdot 10^{-5}
\end{aligned}$$

For τ_{ce} and τ_{cd} we use Eqns (8.8)–(8.9).

Results

We obtain table 8.1, which is shown at the end of this chapter. We take that value for $Q_{11}(i)$ where the percentage of points lying outside the 95 percent confidence limits is less than 5 percent. Further, from these values $Q_{11}(i)$, we take the value with the maximum likelihood, i.e. maximum $L(\mathbf{Y}_N; \hat{Q})$. We obtain

$$\begin{aligned}
\hat{Q}_{11}(1) &= 6.8 \cdot 10^{-4} && (8.27) \\
\hat{Q}_{11}(2) &= 8.2 \cdot 10^{-5} && (8.28) \\
\hat{Q}_{11}(3) &= 1.0 \cdot 10^{-3} && (8.29) \\
\hat{Q}_{11}(4) &= 1.7 \cdot 10^{-5} && (8.30)
\end{aligned}$$

The estimates obtained by this iteration are shown in figure 8.1.

Now we have obtained estimates for $Q_{11}(i)$, $i = 1 \cdots 4$, we shall identify $Q_{22}(i)$, $i = 1 \cdots 4$ in the next section. For this we will use the Maximum Likelihood method. Since the system is not linear, we may not use the method of Mehra.

Estimation of Q_{22}

Method

We want to obtain better estimates for $Q_{22}(i)$, $i = 1 \cdot\cdot 4$ by the Maximum Likelihood method. We use the (initial) values of Eqns (8.6)–(8.11):

$$\begin{aligned}
M_{ebb}(0) &= 5 \cdot 10^{-5} \quad ; \quad M_{flood}(0) = 1 \cdot 10^{-4} \\
W_{s,ebb}(0) &= 1 \cdot 10^{-2} \quad ; \quad W_{s,flood}(0) = 1 \cdot 10^{-2}
\end{aligned}$$

$$\tau_{ce,ebb} = 0.14 \quad ; \quad \tau_{ce,flood} = 0.20$$
$$\tau_{cd,ebb} = 0.06 \quad ; \quad \tau_{cd,flood} = 0.08$$
$$F_k = I_2$$
$$R(k) = 0.7 \cdot 10^{-7}$$

For $Q_{11}(i)$, $i = 1 \cdot\cdot 4$ we use Eqns (8.27)–(8.30):

$$\hat{Q}_{11}(1) = 6.8 \cdot 10^{-4}$$
$$\hat{Q}_{11}(2) = 8.2 \cdot 10^{-5}$$
$$\hat{Q}_{11}(3) = 1.0 \cdot 10^{-3}$$
$$\hat{Q}_{11}(4) = 1.7 \cdot 10^{-5}$$

Results

We assume $Q_{22} \subset \Theta$, $\Theta = [1.0 \cdot 10^{-9}, 1.0]$, and we find the approximation of Q_{22}:

$$\hat{Q}_{22}(1) = 1.0 \cdot 10^{-9}$$
$$\hat{Q}_{22}(2) = 1.0 \cdot 10^{-9}$$
$$\hat{Q}_{22}(3) = 3.0 \cdot 10^{-5}$$
$$\hat{Q}_{22}(4) = 1.0 \cdot 10^{-9}$$

Now, we obtained for the system noise covariance matrices $Q(i)$, $i = 1 \cdot\cdot 4$ (needed for the estimation of the erosion constant M and the settling velocity W_s, both in the cases of ebb and flood),

$$\hat{Q}(1) = \begin{bmatrix} 6.8 \cdot 10^{-4} & 0 \\ 0 & 1.0 \cdot 10^{-9} \end{bmatrix} \tag{8.31}$$

$$\hat{Q}(2) = \begin{bmatrix} 8.2 \cdot 10^{-5} & 0 \\ 0 & 1.0 \cdot 10^{-9} \end{bmatrix} \tag{8.32}$$

$$\hat{Q}(3) = \begin{bmatrix} 1.0 \cdot 10^{-3} & 0 \\ 0 & 3.0.10.^{-5} \end{bmatrix} \tag{8.33}$$

$$\hat{Q}(4) = \begin{bmatrix} 1.7 \cdot 10^{-5} & 0 \\ 0 & 1.0 \cdot 10^{-9} \end{bmatrix} \tag{8.34}$$

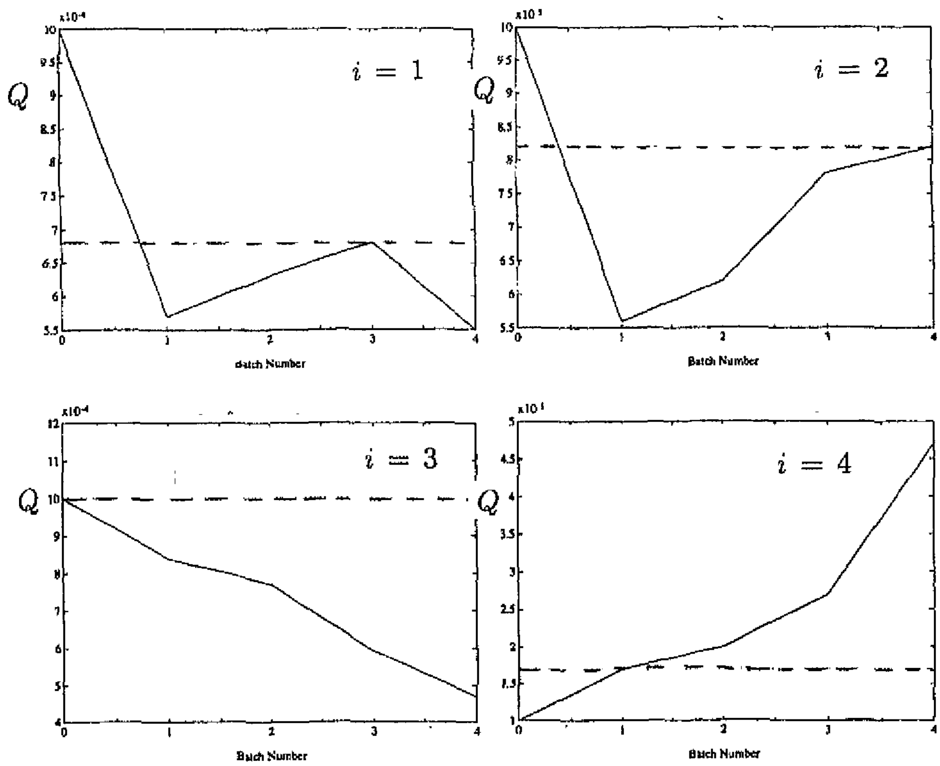


Figure 8.1: Results of the identification of $Q_{11}(i)$ by the method of Mehra, for $i = 1 \cdot\cdot 4$

This is for the estimation of M_{ebb}, M_{flood}, $W_{s,ebb}$, and $W_{s,flood}$. Each batch consists of 430 points (except batch number 5; this one consists of 427 points).

Now we have identified the system noise covariance matrices Q, we are able to determine M and W_s using the (extended) Kalman filter, for the cases of ebb-tide and flood-tide. To estimate the critical values for the bed shear stresses for erosion and for sedimentation, τ_{ce} and τ_{cd}, we need an identification of $Q(i)$, $i = 5 \cdot\cdot 8$. This identification is carried out in the same way as the identification of $Q(i)$, for $i = 1 \cdot\cdot 4$. It is worked out in appendix A.

8.3 Estimation of the unknown parameters using identified Q

8.3.1 Estimation of M and W_S, using identified Q

Method

To estimate the unknown erosion constants M_{ebb} and M_{flood} and the unknown settling velocities $W_{s,ebb}$ and $W_{s,flood}$, we use the MATLAB programs 'rkalf', 'kalfil' and 'kfmean' and the (initial) values (Eqns (8.6)–(8.11)):

$$
\begin{aligned}
M_{ebb}(0) &= 5 \cdot 10^{-5} \quad ; \quad M_{flood}(0) = 1 \cdot 10^{-4} \\
W_{s,ebb}(0) &= 1 \cdot 10^{-2} \quad ; \quad W_{s,flood}(0) = 1 \cdot 10^{-2} \\
\tau_{ce,ebb} &= 0.14 \quad ; \quad \tau_{ce,flood} = 0.20 \\
\tau_{cd,ebb} &= 0.06 \quad ; \quad \tau_{cd,flood} = 0.08 \\
F_k &= I_2 \\
R(k) &= 0.7 \cdot 10^{-7}
\end{aligned}
$$

The system noise covariance matrices $Q(i)$ are described by Eqns (8.31)–(8.34).

Results

The estimates are shown in figure 8.2. Notice that these figures are fluctuating more than the figures of the estimates of chapter 6, where we did not identify Q, but where we have chosen an arbitrary value for Q.

Since the figures are not converging, we take the average value of the estimates, started at the 1000$^{\text{th}}$ estimate, to obtain a value for the estimated parameters. We conclude

$$\hat{M}_{ebb} = 1.5 \cdot 10^{-5} \tag{8.35}$$

$$\hat{M}_{flood} = -6.7 \cdot 10^{-7} \text{ with final value } 4.2 \cdot 10^{-5} \tag{8.36}$$

$$\hat{W}_{s_{ebb}} = 1.4 \cdot 10^{-3} \tag{8.37}$$

$$\hat{W}_{s_{flood}} = 9.3 \cdot 10^{-4} \tag{8.38}$$

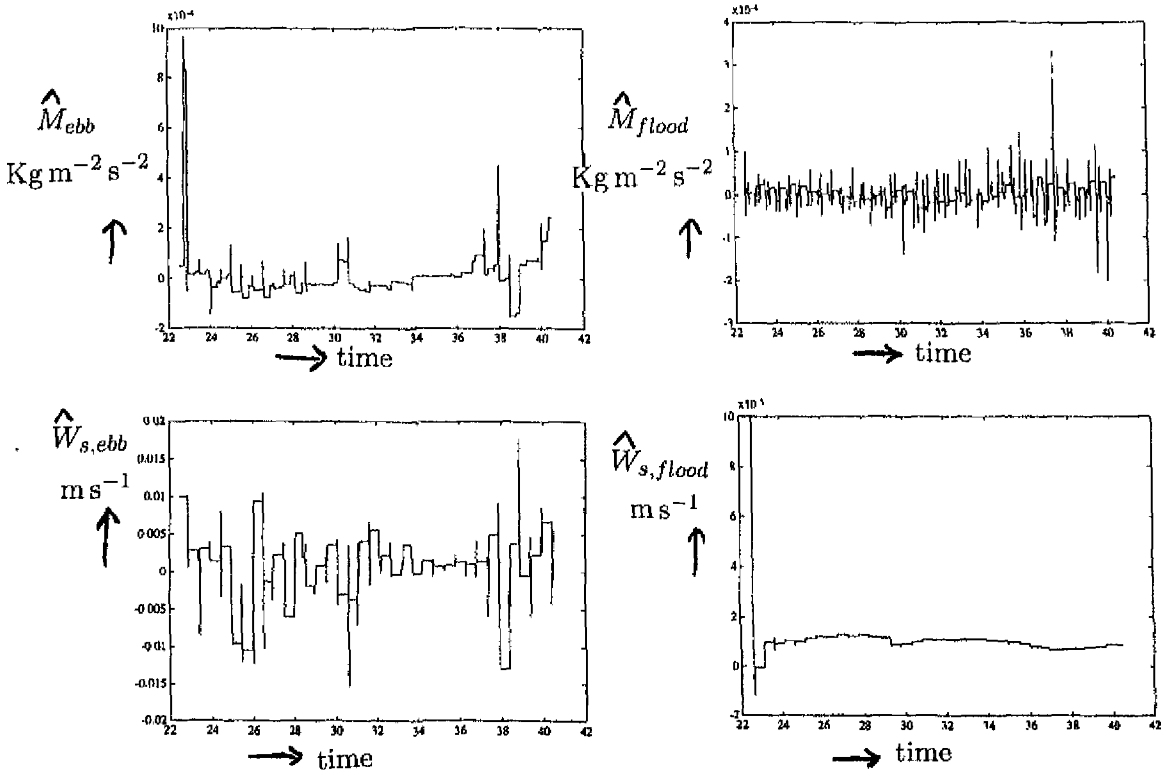


Figure 8.2: Estimates of the erosion constant M and the settling velocity W_s, in the cases of ebb-tide and flood-tide, using identified Q, over a period of eighteen days

8.3.2 Estimation of $\tau_{\rm ce}$ and $\tau_{\rm cd}$, using identified Q

The estimations to determine the system noise covariance matrices $Q(i), \ i = 5 \cdots 8$ as well as the estimations to determine the critical bed shear stresses τ_{ce} and τ_{cd}, using these Q-matrices are given in appendix A. In this chapter we will only give the results.

Results

We find the system noise covariance matrices Q:

$$\hat{Q}(5) \quad = \quad \begin{bmatrix} 5.7 \cdot 10^{-4} & 0 \\ 0 & 1.0 \cdot 10^{-9} \end{bmatrix} \tag{8.39}$$

$$\hat{Q}(6) = \begin{bmatrix} 1.4 \cdot 10^{-4} & 0 \\ 0 & 1.0 \cdot 10^{-8} \end{bmatrix} \quad (8.40)$$

$$\hat{Q}(7) = \begin{bmatrix} 4.3 \cdot 10^{-4} & 0 \\ 0 & 1.0 \cdot 10^{-8} \end{bmatrix} \quad (8.41)$$

$$\hat{Q}(8) = \begin{bmatrix} 1.4 \cdot 10^{-5} & 0 \\ 0 & 1.0 \cdot 10^{-8} \end{bmatrix} \quad (8.42)$$

where $Q(i)$, $i = 5 \cdot\cdot 8$ belongs to the system to determine $\tau_{ce,ebb}$, $\tau_{ce,flood}$, $\tau_{cd,ebb}$, or $\tau_{cd,flood}$.

We use these system noise covariance matrices Q to determine the critical bed shear stresses for erosion and sedimentation, τ_{ce} and τ_{cd}, using the extended Kalman filter. The estimates are shown in figure 8.3. Since the figures of the estimates do not converge, we determine a value for the estimated parameters by taking the average value of the estimates, starting at the 1000^{tn} value. We conclude

$$\hat{\tau}_{ce,ebb} = -0.33 \quad \text{and} \quad \hat{\tau}_{ce,flood} = 0.89 \quad (8.43)$$

$$\hat{\tau}_{cd,ebb} = 0.14 \quad \text{and} \quad \hat{\tau}_{cd,flood} = 0.26 \quad (8.44)$$

Notice that $\tau_{ce,ebb} < 0$, and $\tau_{ce} < \tau_{cd}$. So the critical bed shear stress for erosion τ_{ce} is not a reliable value. Something is wrong. This may be examined in further investigation. On the other hand ,the situation of flood looks well.

Remark: At this time the Kalman filter was not checked by using simulated observations. In later investigation, I did this, and it appeared that the Kalman filter I used was not correct. Maybe I made a calculation error, when I wrote the computer programs. When I estimated the unknown parameters, using simulated observations, the figures of the estimates were converging well.

8.3.3 Obtained model using identified Q

Method

We substitute the obtained estimates for the erosion constant M, the settling velocity W_s, the critical bed shear stress for erosion τ_{ce} and the critical bed shear stress for sedimentation τ_{cd}, in the cases of ebb-tide and flood-tide in the model (Eqns (8.45)–(8.48)).

$$M_{ebb} = 2 \cdot 10^{-5} \quad ; \quad M_{flood} = 4 \cdot 10^{-5} \tag{8.45}$$

$$W_{s,ebb} = 1 \cdot 10^{-3} \quad ; \quad W_{s,flood} = 9 \cdot 10^{-4} \tag{8.46}$$

$$\tau_{ce,ebb} = -0.33 \quad ; \quad \tau_{ce,flood} = 0.89 \tag{8.47}$$

$$\tau_{cd,ebb} = 0.14 \quad ; \quad \tau_{cd,flood} = 0.26 \tag{8.48}$$

We use the MATLAB programs 'rgeg' and 'model'.

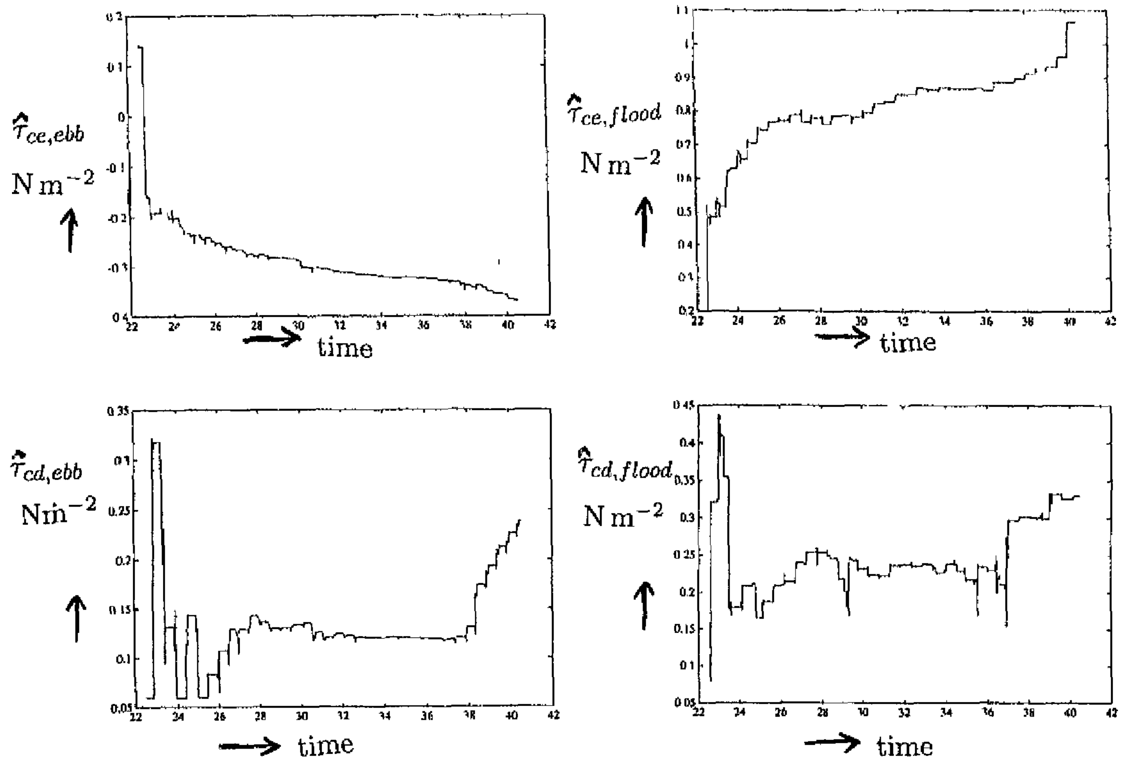


Figure 8.3: Estimates of the critical bed shear stresses for erosion and sedimentation, τ_{ce} and τ_{cd}, in the cases of ebb-tide and flood-tide, using identified Q, over a period of eighteen days

Results

In figure 8.4 we compare the computed cohesive sediment concentrations with the observed concentrations. Looking at this figure, we notice that the computed cohesive sediment concentrations are negative. This is not what we want to have. On the other hand, the shape of the peaks of the computed concentrations looks like the shape of the peaks of the observed concentrations. Maybe little adaptations are enough to improve our model.

Remark: See the remark made in the previous section about checking the Kalman filter with simulated observations.

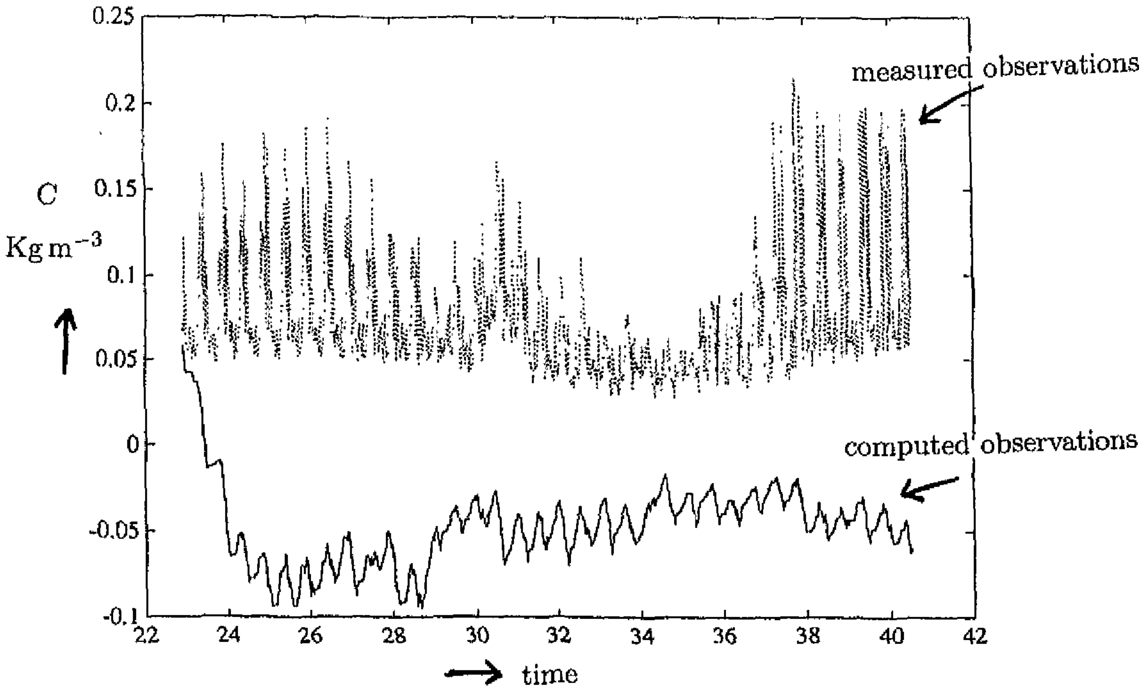


Figure 8.4: The model compared with the observations, using identified Q, over a period of eighteen days

We used Eqns (8.45)–(8.48) for the unknown parameters of the mathematical model.

8.4 Conclusions

In this chapter we used the Maximum Likelihood method to determine whether or not the estimations, needed to determine the unknown parameters of the cohesive sediment concentration model, have been influenced by the system noise covariance matrix Q. We made a rough estimation of the system noise covariance matrices of the eight systems, i.e. one system for each of the unknown parameters of the concentration model. It appeared that the extended Kalman filter is depending on this matrix.

We assumed the shape of the system noise covariance matrices to be:

$$Q_i = \begin{bmatrix} Q_{11}(i) & 0 \\ 0 & Q_{22}(i) \end{bmatrix} \qquad 1 = 1 \cdots 8$$

where $Q_{11}(i)$ and $Q_{22}(i)$ not depending on time k, to keep the problem simple. We estimated Q_{11} with the method of Mehra using as initial value for Q a rough estimation of Q_{11}. After that we estimated Q_{22} by the Maximum Likelihood method.

Using the obtained system noise covariance matrices Q, we estimated the erosion constant M, the settling velocity W_s, the critical bed shear stresses for erosion and sedimentation, τ_{ce} and τ_{cd}, for both the cases of ebb-tide and flood-tide. We obtained figures which are strongly fluctuating in time, and the average of the obtained values is

$$\begin{aligned} \hat{M}_{ebb} &= 2 \cdot 10^{-5} \quad ; \quad \hat{M}_{flood} &= 4 \cdot 10^{-5} \\ \hat{W}_{s,ebb} &= 1 \cdot 10^{-3} \quad ; \quad \hat{W}_{s,flood} &= 9 \cdot 10^{-4} \\ \hat{\tau}_{ce,ebb} &= -0.33 \quad ; \quad \hat{\tau}_{ce,flood} &= 0.89 \\ \hat{\tau}_{cd,ebb} &= 0.14 \quad ; \quad \hat{\tau}_{cd,flood} &= 0.26 \end{aligned}$$

The estimated value of $\tau_{ce,ebb}$ is negative. This can not be real. But, due to a lack of time, I did not examine this value any further.

The obtained values were substituted in the model, describing the tidal effects on the cohesive sediment concentration in the Western Scheldt. The computed concentrations were compared with the observed concentrations. We found that the computed concentrations were negative, caused by the negative value of $\tau_{ce,ebb}$. But the figure of the computed concentrations has the same characteristics as the figure of the observed concentrations. Although the peaks in the computed figure are not as high as the peaks in the observed figure, we observe in the computed figure some of the peaks we want to describe.

Although we made a lot of assumptions to use both the Maximum Likelihood method and the method of Mehra, we may conclude that the extended Kalman filter with identified process noise covariance matrix Q, may result into better values for the unknown parameters, than the extended Kalman filter with arbitrary Q.

Remark: Maybe the estimates will be even better, when we use a correct Kalman filter program. Since in later investigation, (by estimating using simulated observations,) it appeared that maybe the used program contains some calculation errors.

<u>Note 1</u>: An estimation of Q_{11} with the Maximum Likelihood method, gives the same result as an estimation of Q_{11} with the method of Mehra. Provided that the initial value we use in both methods is the same. (In this case the initial value is the rough estimate obtained by the Maximum Likelihood method.) This conclusion is very useful, because the method of Mehra takes less computer time than the Maximum Likelihood method.

<u>Note 2</u>: While using the method of Mehra, the data were divided into 5 batches. In doing this we assume the circumstances of all of them are the same. Of course this is not true. But this assumption does not cause an error, as can be understood by the fact that the obtained estimates of each batch look right, and they do not differ much from each other.

Model Number i	Number of Iterations	$\hat{Q}$	Likelihood Function $L(\mathbf{Y}_N;\hat{Q})$ *1	Percentage *2
1	0	$1.0 \cdot 10^{-3}$	6.13	0.07
	1	$5.7 \cdot 10^{-4}$	6.15	0.13
	2	$6.3 \cdot 10^{-4}$	6.36	0.03
	3	$6.8 \cdot 10^{-4}$	6.93	0.00
	4	$5.5 \cdot 10^{-4}$	2.82	0.00
		$7.9 \cdot 10^{-4}$		
2	0	$1.0 \cdot 10^{-4}$	8.65	0.20
	1	$5.6 \cdot 10^{-5}$	8.73	0.30
	2	$6.2 \cdot 10^{-5}$	7.95	0.10
	3	$7.8 \cdot 10^{-5}$	8.32	0.07
	4	$8.2 \cdot 10^{-5}$	4.94	0.03
		$1.4 \cdot 10^{-4}$		
3	0	$1.0 \cdot 10^{-3}$	6.02	0.03
	1	$8.4 \cdot 10^{-4}$	6.45	0.17
	2	$7.7 \cdot 10^{-4}$	6.96	0.13
	3	$5.9 \cdot 10^{-4}$	7.25	0.07
	4	$4.7 \cdot 10^{-4}$	1.76	0.13
		$7.3 \cdot 10^{-4}$		
4	0	$1.0 \cdot 10^{-5}$	9.38	0.03
	1	$1.7 \cdot 10^{-5}$	9.63	0.03
	2	$2.0 \cdot 10^{-5}$	7.42	0.00
	3	$2.7 \cdot 10^{-5}$	5.25	0.07
	4	$4.7 \cdot 10^{-5}$	9.13	0.00
		$4.5 \cdot 10^{-5}$		

Table 8.1: Results of the method of Mehra, to estimate the system noise covariance matrices $Q(i)$, $i = 1 \cdot\cdot 4$

$*_1$: *The likelihood function we use here is*

$$L(\mathbf{Y}_N;\hat{Q}) \;=\; -\frac{1}{N}\sum_{k=1}^{N} Z_k^T(CPC^T + R)^{-1}Z_k - ln\,|\,CPC^T + R\,|$$

$*_2$: *Percentage of points lying outside the 95 percent confidence limits.*

<u>*Note:*</u>
The estimate of the actual mean square error

$$(\frac{1}{N})\sum_{k=1}^{N}(X_k - \hat{X}_{k|k-1})^T(X_k - \hat{X}_{k|k-1})$$

where X_k is obtained by actual estimation, and the calculated mean square error $tr(P_0)$, as computed in the program, are not written in this table, because those values were equal.

Chapter 9

Summary and Conclusions

We wanted to construct a mathematical model, which describes the tidal effects on the cohesive sediment transport in the Western Scheldt. We used data obtained near Bath, and we wanted to describe the influence of the water level and the (inland) horizontal velocity on the cohesive sediment concentration in a fixed column near Bath. The mathematical model we used is a simplification of the conservation of mass equation for a column of water,

$$\frac{\partial C}{\partial t} = \frac{S}{H} = \frac{\partial E}{\partial t} - \frac{\partial d}{\partial t}$$

This means that the change of the cohesive sediment concentration C in time, equals a certain source term S, divided by the water depth H. This is equal to the amount of erosion E minus the amount of sedimentation d at the bottom of the water column.

Erosion takes place when the bed shear stress τ_b is higher than the critical bed shear stress for erosion τ_{ce}. Further, sedimentation takes place when the bed shear stress τ_b is less than the critical bed shear stress for sedimentation τ_{cd}. Substituting the expressions for the situations of erosion and sedimentation into the conservation of mass equation, we obtained the discrete equations,

$$\begin{aligned} C_{k+1} &= C_k + \frac{\alpha M}{H_k}\left(\frac{\tau_b(k)}{\tau_{ce}} - 1\right) && \tau_b(k) \geq \tau_{ce} \\ C_{k+1} &= C_k && \tau_{cd} < \tau_b(k) < \tau_{ce} \\ C_{k+1} &= C_k + \frac{\alpha C_k W_s}{H_k}\left(\frac{\tau_b(k)}{\tau_{cd}} - 1\right) && \tau_b(k) \leq \tau_{cd} \quad . \end{aligned}$$

where $\alpha = \Delta t$ is the discrete time step, and the bed shear stress $\tau_b(k)$ is given by,

$$\tau_b(k) = \frac{\rho_w\, g\, V_k^2(\text{surface})}{(Ch)^2}\left(\frac{2}{H_k - 1}\right)^{0.30}$$

We estimated the erosion constant M, the settling velocity W_s, the critical bed shear stress for erosion τ_{ce} and the critical bed shear stress for sedimentation τ_{cd}. To estimate these unknown parameters, we used the extended Kalman filtering method. This method is useful for our estimation problem, because it is recursive, i.e. we do not need to store the previous measurements while updating the estimate, and because it takes into account the system and the measurement noise.

First, we estimated the erosion constant M and the settling velocity W_s, assuming certain constant values for the critical bed shear stresses for erosion and sedimentation, τ_{ce} and τ_{cd}. Next, we estimated the unknown τ_{ce} and τ_{cd}, using the obtained estimated values for M and W_s. We substituted the four obtained estimates into the model, and we compared the computed cohesive sediment concentrations with the observed concentrations. We found that the figure of the computed concentrations was not equal to the figure of the observed concentrations. We concluded that the obtained estimates would not be correct.

We improved the mathematical model. We distinguished between the ebb-stream and the flood-stream, since these are two different situations. Further, we adapted the bed shear stress expression $\tau_b(k)$ with a certain time shift. We considered the static time shift and the dynamic time shift,

Static time shift:

$$\tau_b(k) := \tau_b(k - 15)$$

Dynamic time shift:

$$\begin{aligned}\tau_{b,ebb}(k) &= \tau_b(k - \frac{1}{0.008380}\frac{0.13}{V_k}) \\ \tau_{b,flood}(k) &= \tau_b(k - \frac{1}{0.008380}\frac{0.03}{V_k})\end{aligned}$$

Now, we estimated eight unknown parameters: the erosion constant M, the settling velocity W_s, the critical bed shear stress for erosion τ_{ce} and the critical bed shear stress for sedimentation τ_{cd}, all for the cases of ebb-tide and flood-tide. We estimated these eight unknown parameters for a short period during spring tide, and we used the bed shear stress expression with a static shift. Substituting the obtained values into the model, we obtained computed cohesive sediment concentrations. When we compared these concentrations with the observed concentrations, we noticed that they did not agree.

After these estimations I had an idea of the eight unknown parameters. I substituted some arbitrary values for the unknown parameters in the model. The shape of the figure of the obtained cohesive sediment concentrations looked like the shape of the figure of the real observations. We noticed the same peaks, however they were shifted in time. The period we examined, was a period during a spring tide/neap tide cycle. The fluctuations of the cohesive sediment concentration during this cycle were shown in the figure of the computed cohesive sediment concentrations. Since these substituted parameters were not in harmony with each other, because I made them up by myself, these parameters were not the parameters we were looking for. However, this result showed that the model is not very bad.

To improve the working of the extended Kalman filter, we identified the system noise covariance matrix Q. We used two methods to identify this matrix: the Maximum Likelihood method, and a method suggested by Mehra.

The Maximum Likelihood method considered a linear time invariant discrete time stochastic dynamical system

$$\begin{aligned} X_{k+1} &= AX_k + Bu(k) + FW_k \\ Y_k &= CX_k + V_k \qquad\qquad , k \geq 0 \end{aligned}$$

where the state X_k is an n-dimensional vector, the system disturbance W_k is an r-dimensional vector, the observation Y_k is an m-dimensional vector, V_k is the observation error, and $\{u(k)\}$ is a known p-dimensional vector input sequence. $\{W_k\}$ and $\{V_k\}$ are independent sequences of independent, zero mean Gaussian random vectors with covariances Q and R, respectively. We supposed that the matrices A and B were only partially known and θ was the vector of the unknown parameters in those matrices. We assumed that θ belonged to a compact (closed and bounded) parameter space $\Theta \subset \mathbb{R}^d$.
The parameter estimation problem was, to find an estimate of θ, based on the observations $Y_0 = y_0, \cdots, Y_N = y_N$, for some fixed N. We denoted

such an estimate by $\hat{\theta}_N(y_0, \cdots, y_N)$. We determined the maximum likelihood estimator of θ, denoted by $\hat{\theta}_N$, that is that value of θ for which the likelihood function $L(\mathbf{Y}_N; \theta)$ had a maximum.

The method suggested by Mehra in 1970, considered the multivariable linear discrete system

$$\begin{aligned} X_{k+1} &= AX_k + FW_k \\ Y_k &= CX_k + V_k \end{aligned}$$

with X_k an $n \times 1$ state vector, A an $n \times n$ non singular transition matrix, F an $n \times q$ constant input matrix, Y_k an $r \times 1$ measurement vector and C an $r \times n$ constant output matrix. The sequences $W_k\,(q \times 1)$ and $V_k\,(r \times 1)$ were uncorrelated Gaussian white noise sequences. It was assumed that the system was time invariant, completely controllable and observable. Both the system and the filter (optimal or suboptimal) were assumed to have reached steady-state conditions. The method of Mehra checked whether the Kalman filter constructed using an initial estimate of the system noise covariance matrix Q_0 was close to optimal or not. If it was suboptimal the method yielded a new estimate $\hat{Q}$ using the autocorrelation function of the innovation process.

We assumed that the process noise covariance matrix had the shape

$$Q_i = \begin{bmatrix} Q_{11}(i) & 0 \\ 0 & Q_{22}(i) \end{bmatrix} \qquad i = 1 \cdots 8$$

where $Q_{11}(i)$ and $Q_{22}(i)$ were not depending on time to keep the problem simple. We estimated Q_{11} by the method of Mehra, and Q_{22} by the Maximum Likelihood method. Since the method of Mehra is written for a linear system, we used the linear equations

$$\begin{aligned} C_{k+1} &= C_k + \frac{\alpha M}{H_k}\left(\frac{\tau_b(k)}{\tau_{ce}} - 1\right) + FW_k & \tau_b(k) \geq \tau_{ce} \\ C_{k+1} &= C_k + FW_k & \tau_{cd} < \tau_b(k) < \tau_{ce} \\ C_{k+1} &= C_k + \frac{\alpha C_k W_s}{H_k}\left(\frac{\tau_b(k)}{\tau_{cd}} - 1\right) + FW_k & \tau_b(k) \leq \tau_{cd} \end{aligned}$$

to identify Q_{11}. To use the method of Mehra, we should know the eight unknown parameters. Since these were unknown, we estimated them, using the rough estimates for Q. To identify Q_{22} with the Maximum Likelihood method we used the linearized systems, as used in the extended Kalman filtering method.

First, we obtained rough estimations of $Q_{11}(i)$ and $Q_{22}(i)$ with the Maximum Likelihood method. These estimates were used as initial values by determining $Q_{11}(i)$ with the method of Mehra and $Q_{22}(i)$ with the Maximum Likelihood method.

The method suggested by Mehra needed less computer time than the Maximum Likelihood method. I checked whether the method of Mehra gave the same results for $\hat{Q}_{11}$, as the method of Maximum Likelihood, using the same initial value $Q_{11}(0)$. It appeared that both models gave the same estimates. So, we concluded, that the method of Mehra needed less computer time than the Maximum Likelihood method, and they both gave the same results. But we might not forget, that the method of Mehra, did not only assume that the discrete system was linear and time invariant, but it also assumed that the discrete system was completely controllable and observable, and that both the system and the filter had reached steady-state. So when all these conditions are fulfilled, it is likely to use the method suggested by Mehra above the Maximum Likelihood method, in the other case it is desirable to use the Maximum Likelihood method.

Using the identified system noise covariance matrices Q, we estimated the eight unknown parameters. The figures of the estimates varied a lot, and I have chosen the average values of the estimates. These average values were substituted in the mathematical model in order to obtain cohesive sediment concentrations. Comparing the figure of these computed concentrations with the figure of the observed concentrations, we noticed that the shape of the computed figure looked correct, although the computed concentrations were negative. Further, the peaks of the concentration were well shown in the computed concentration figure.

The values we finally obtained were

$$\begin{array}{lcllcl} M_{ebb} & = & 2\cdot 10^{-5} & ; \; M_{flood} & = & 4\cdot 10^{-5} \\ W_{s,ebb} & = & 1\cdot 10^{-3} & ; \; W_{s,flood} & = & 9\cdot 10^{-4} \\ \tau_{ce,ebb} & = & -0.33 & ; \; \tau_{ce,flood} & = & 0.89 \\ \tau_{cd,ebb} & = & 0.14 & ; \; \tau_{cd,flood} & = & 0.26 \end{array}$$

We noticed that $\tau_{ce,ebb} < 0$, which is not likely. This might be caused by the fact that the peaks of the cohesive sediment concentration at ebb-tide are much higher than the peaks of the cohesive sediment concentration at flood-tide. (This is shown in figure 2.2 in section 2.3.) The situation of ebb-tide could not be true at all, because $\tau_{ce,ebb} < \tau_{cd,ebb}$ in stead of $\tau_{ce,ebb} > \tau_{cd,ebb}$. However, in the case of flood-tide, we found $\tau_{ce,flood} > \tau_{cd,flood}$. But the $\tau_{ce,flood}$ is very high, especially when you have in mind that we assumed

$$\tau_{ce}(\text{cohesive sediment}) < \tau_{ce}(\text{sand}) \quad \approx \quad 0.20\mathrm{N\,m^{-2}}$$

according to van Leussen, [10]. We concluded that we had to make some improvements to obtain more reliable values, especially in the case of ebb-tide.

We may conclude that the extended Kalman filter may be applied to solve this parameter problem, but that it does not work well. When we identify the system noise covariance matrix Q, we obtain better estimates for the unknown parameters of the mathematical model, but these estimates do not result in a model, which generates nice cohesive sediment concentrations. The model itself is not so very bad, as is shown, when we substituted some arbitrary values for the unknown parameters into the model; the obtained cohesive sediment concentrations look right.

Remark: The used Kalman filter had to be checked using simulated observations. By lack of time, I did not do this during the investigation. At the end of the investigation, I checked the Kalman filter, using simulated observations. I found that the Kalman filter I used, was not working correct; the estimates obtained, using simulated observations were not the same as the estimates obtained, using the measured observations. However, the estimates obtained using simulated observations were converging well, and this is not the case when we used the measured observations. So we may conclude that the Kalman filter I used in this investigation may contain some calculation errors. Since I concluded this at the end of this investigation, there was no time to find these errors and improve the estimations. (This might be done in further investigation)

<u>Note 1</u>: At the end of this investigation, I noticed that τ_b was not correctly constructed in the used FORTRAN and MATLAB programs; a factor $2^{0.30}$ is missing in the τ_{b_k} expression (Eqn (3.16)). Since the shape of the figure of the used $\tau_b(k)$ is the same as the figure of the real $\tau_b(k)$, and we concentrated on the location of the peaks in the figure, there is no harm in this.

Note 2: It is not easy to identify the process noise covariance matrix exactly, because the choice which of the models to use is depending on the determined estimate. For instance, when we estimate τ_{ce} and τ_{cd}, we make use of two models. At each measurement update $(\hat{X}_{k+1})$, the choice which model to use is determined by the values of $\hat{\tau}_{ce_{k+1|k}}$ and $\hat{\tau}_{cd_{k+1|k}}$ of the previous time update, i.e. when $\tau_b(k+1) > \hat{\tau}_{ce}(k+1 \mid k)$ we are going to estimate $\hat{\tau}_{ce}(k+1)$, when $\tau_b(k+1) < \hat{\tau}_{cd}(k+1 \mid k)$ we are going to estimate $\hat{\tau}_{cd}(k+1)$, and when $\hat{\tau}_{cd}(k+1 \mid k) < \tau_b(k+1) < \hat{\tau}_{ce}(k+1 \mid k)$ we are going to estimate neither $\hat{\tau}_{ce}(k+1)$ nor $\hat{\tau}_{cd}(k+1)$. So the assumption of two separated models we use here, (i.e. one for the case of erosion and one for the case of sedimentation) is not a realistic one; the models interact.

Note 3: In trying to estimate M and W_s after estimating τ_{ce} and τ_{cd}, we obtained values which cannot be realistic. Thus the decision was made to start estimating M and W_s, and to estimate the critical bed shear stresses after that.

We also examined the difference between starting with the situation of erosion, $\tau_b \geq \tau_{ce}$, followed by the situation of sedimentation, $\tau_b \leq \tau_{cd}$, or starting with the situation of sedimentation, followed by the situation of erosion. When using the first approach, i.e. starting with erosion, followed by sedimentation, strange values of the estimates were obtained. Therefore we decided to use the second approach: starting with the situation of sedimentation, followed by the situation of erosion. This decision effects the final obtained estimates.

Note 4: It appeared to be very important to use the right initial values while determining the estimates.

Extension of the model

Adaptation of τ_{ce}

We divided the critical bed shear stresses for erosion and sedimentation, τ_{ce} and τ_{cd}, both into two parts, one for the situation of ebb-tide and one for the situation of flood-tide. We assumed these four critical bed shear stresses to be constant, this assumption is not a realistic one. The $\tau_{ce,ebb}$ for instance, will not behave like a constant during the whole period of ebb-tide. When the erosion process starts, there is a layer of fresh cohesive sediment on the top of the bottom. The τ_{ce} of this layer is very low. Much lower than the τ_{ce} a few time steps later. Then a big part of the layer of fresh sediment, or even the whole layer is eroded and the bottom exists of only cohesive sediment particles mixed with sand particles. Since the τ_{ce} of sand is much higher than that of cohesive sediment, the τ_{ce} in this case will be higher than a few time steps earlier. So we may conclude that, the τ_{ce} is depending on time and on the amount of cohesive sediment settled at the bottom during the period of sedimentation before, and it is not a constant.

Adaptation of W_s

We assumed W_s to be a constant. But W_s is depending on, for instance, the rate of flocculation. Further we have to take into consideration, that W_s is higher near the surface of the water column and lower near the bottom. Thus far we only used $W_s = W_{const}$ to describe the settling velocity. But maybe it is better to use all three equations describing W_s,

$$\begin{aligned} W_s &= W_{const}, & C &< C_1 \\ W_s &= K_1 C^n, & C_1 \le C &\le C_2 \\ W_s &= W_{s0}(1 - K_2 C)^{\beta}, & C &> C_2 \end{aligned}$$

Where W_{const} is a constant value of the settling velocity and K_1, K_2, W_{s0}, n and β are coefficients depending on the sediment type and the salinity, and C is the cohesive sediment concentration.

Adaptation of M

We assumed the erosion constant M to be a constant. But this has not to be correct. The parameter M is applied to correct the erosion term in the mathematical model. So it is acceptable to assume that M is not a constant, but a term which is varying in time.

Vertical diffusion coefficient

In the approach used in this study, we used a simplified mathematical model to describe the concentration of cohesive sediment C, depending on the amount of erosion E minus the amount of sedimentation d at the bottom. We assumed that the total amount of cohesive sediment is equally mixed through the whole column. We may adjust the mathematical model by extending it with a diffusion coefficient and using the data of the concentration of the cohesive sediment of all three levels in the water column (and not only on the mean water level.)

We use (W. van Leussen, [10])

$$\text{Upper layer} \quad : \quad a_1\frac{\partial C_1}{\partial t} - C_i(W_i - W_s) - f_i = S_1 \tag{9.1}$$

$$\begin{aligned}\text{Mean layer} \quad : \quad & a_2\frac{\partial C_2}{\partial t} + C_i(W_i - W_s) + f_i \\ & - C_j(W_j - W_s) - f_j = S_2\end{aligned} \tag{9.2}$$

$$\text{Bottom layer} \quad : \quad a_3\frac{\partial C_3}{\partial t} + C_j(W_j - W_s) + f_j = S + S_3 \tag{9.3}$$

The layers are indicated at figure 9.1 and

S_l = Supply of cohesive sediment in layer l $(l = 1, 2, 3)$

S = Source term $= \frac{\partial E}{\partial t} - \frac{\partial d}{\partial t}$

f_i = Turbulent diffusion at the boundary surface i

C_i = Concentration of cohesive sediment at boundary i

W_i = Entrainment velocity in vertical direction

a_i = Diameter of layer i

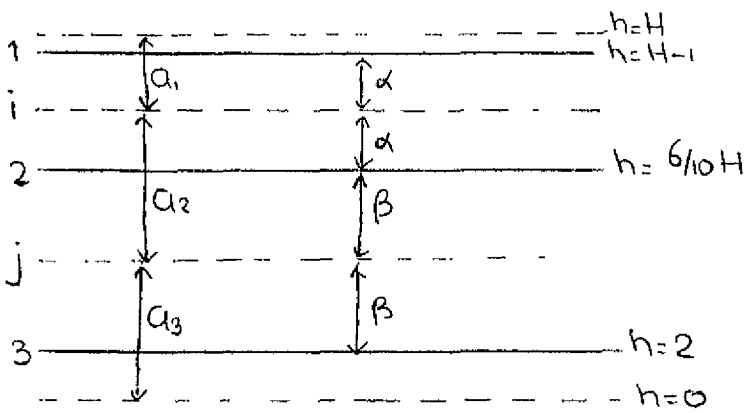


Figure 9.1: Water column divided into three parts

Assume

$$S_l = 0 \quad (l = 1, 2, 3)$$

$$f_i = m_i(\rho_2 - \rho_1)(u_1 - u_2) \tag{9.4}$$

$$
\begin{aligned}
f_j &= m_j(\rho_3 - \rho_2)(u_2 - u_3) && (9.5)\\
&\quad \text{where } m_i \text{ and } m_j \text{ are constants}
\end{aligned}
$$

$$
\begin{aligned}
C_i &= \frac{C_1 + C_2}{2} && (9.6)\\
C_j &= \frac{C_2 + C_3}{2} && (9.7)
\end{aligned}
$$

$$
\begin{aligned}
W_i &= W_0(u_1 - u_2)R_i^{-n}, \quad R_i = \frac{\Delta\rho\, g\, a_1}{\rho(u_1 - u_2)^2} && (9.8)\\
W_j &= W_0(u_2 - u_3)R_j^{-n}, \quad R_j = \frac{\Delta\rho\, g\, a_2}{\rho(u_2 - u_3)^2} && (9.9)\\
&\quad (0.5 < n < 2)
\end{aligned}
$$

where u_i : depth mean velocity component along the x-direction in layer i
ρ : the density of water
g : the gravitation constant

Simplify this by leaving the entrainment velocity out of consideration. We obtain

$$
\begin{aligned}
\text{Upper layer} &: \; a_1\frac{\partial C_1}{\partial t} = f_i && (9.10)\\
\text{Mean layer} &: \; a_2\frac{\partial C_2}{\partial t} = f_j - f_i && (9.11)\\
\text{Bottom layer} &: \; a_3\frac{\partial C_3}{\partial t} = S - f_j\\
&\qquad\qquad = \frac{\partial E}{\partial t} - \frac{\partial d}{\partial t} - f_j && (9.12)
\end{aligned}
$$

With this extension, we may take into account the time a cohesive sediment particle needs to reach the bottom or the surface of a water column. Further, for instance, assuming S_l not equal to zero, we may also consider the dumping of material in the water column.

Equations with V^4

Thus far we only used bed shear stress equations depending on V^2. In the description of the behaviour of sand particles most bed shear stress equations are depending on V^4. Maybe it is a good idea to use the knowledge of these equations of sand particles to describe the behaviour of the cohesive sediment particles. Since, in a mixed bottom it is most likely that the

τ_{ce}(erosion) is acting like the τ_{ce}(sand).

With the examples of extensions made in this section, we may improve the model we used in this investigation, in order to obtain a more reliable model.

Further investigation

In view of the results obtained in this study, I was asked to proceed this investigation, and to write a proposal about the continuation of this project. This proposal is added as a loose supplement. I will describe it in a few words.

The mathematical model for the location Bath may be improved by including the vertical diffusion coefficient. Further, it is possible to state the same kind of models for the locations 'Middelgat' and 'Vlissingen' (figure 2.1). These three models give an indication of the influence of tide on the concentration of cohesive sediment at the three locations in the Western Scheldt. The influence of season may be determined from other measurements, made during twenty years in the whole Western Scheldt. Using the knowledge of the tidal and the seasonal effects on the cohesive sediment concentration, we may eliminate both the tidal effects and the seasonal effects from the the cohesive sediment concentrations of the last twenty years in the Western Scheldt. In this manner, we will obtain a knowledge of a certain trend in the cohesive sediment concentration in the Western Scheldt, over the last twenty years.

Bibliography

[1] Anderson, B.D.O. and J.B. Moore, *Optimal Filtering*, Prentice Hall, Englewood Cliffs, New Jersey, 1979

[2] Bagchi A., *Optimal Control of Stochastic Systems*, Prentice Hall, London, 1993

[3] Gupta N.K. and R.K. Mehra, 'Computational Aspects of Maximum Likelihood Estimation and Reduction in Sensitivity Function Calculations', *IEEE Trans. Automatic Control*, vol. AC-19, pp. 774–783, December 1974

[4] Hannan E.J., *Time Series Analysis*, Methuen & Co, London, 1960

[5] Jenkins G.M. and D.G. Watts, *Spectral Analysis and its applications*, Holden Day, San Francisco, 1968

[6] Jong H. de, 'Keuze van een bruto-reeks uit de beschikbare Zeekat-resultaten t.b.v. een nadere interpretatie', Rijkswaterstaat, dienst getijdewateren, notitie GWAO-89.1305, oktober 1989

[7] Jong H. de, 'Zeekat-registraties in het Nauw van Bath (Westerschelde)', Rijkswaterstaat, dienst getijdewateren, notitie GWAO-89.1357, december 1989

[8] Kendall M. and J.K. Ord, *Time series*, Charles Griffin, London, 1990

[9] Kendall M.G. and A. Stuart, *The advanced theory of statistics*, Volume 3, Charles Griffin, London, 1976

[10] Leussen W. van, 'Onderzoek naar de transportmechanismen van slib in zoet/zout omstandigheden', deel I t/m IV, Rijkswaterstaat Waterloopkundig Laboratorium, mei 1981

[11] Leussen W. van, 'Fine Sediment Transport under Tidal Action', *Geo-Marine-Letters*, vol. 11, pp. 119–126, Springer-Verlag, New York, 1991

[12] Maiwald K.D. en J.H.G. Verhagen, 'Trendanalyse van zwevend stof gegevens van de Waddenzee', opdrachtgever Rijkswaterstaat Dienst Getijdewateren, Waterloopkundig Laboratorium, april 1991

[13] Maldegem D.C. van, 'De slibbalans van het Schelde-estuarium', RWS D.G., Algemeen Onderzoek Fysica, NOTA GWAO-91.081, Middelburg, juni 1993

[14] Maldegem D.C. van and H.P.J. Mulder and A. Langerak, 'A cohesive sediment balance for the Scheldt estuary', Netherlands Journal of Aquatic Ecology, 1993

[15] Manni R.J., 'Slibtransport en slibbalans in de Westerschelde', Studierapport no.4, Rijkswaterstaat Vlissingen, Utrecht, nov 1986

[16] Mehra R.K., 'On the Identification of Variances and Adaptive Kalman Filtering', *IEEE Trans. Automatic Control*, vol. AC-15, pp. 175–184, April 1970

[17] Ministerie van Verkeer en Waterstaat, 'Getijtafels voor Nederland 1988', Staatsuitgeverij, 's-Gravenhage, 1987

[18] Ministerie van Verkeer en Waterstaat, 'Slibonderzoek', Waterloopkundig Laboratorium, januari 1989

[19] Mortensen R.E., *Random signals and systems*, Wiley-Interscience, Singapore, 1987

[20] Mous S.L.J., 'On identification of nonlinear systems', Ph.D.Thesis, Landbouwuniversiteit Wageningen, april 1994

[21] Mulder H.P.J., 'Specificaties sedimenttransport-formulering in SIMONA', werkdocument GWAO-92.620x, Dienst Getijdewateren Rijkswaterstaat, september 1992

[22] Pathirana K.P.P., Z.W. Song, and C.S. Yu, 'Modelling tidal induced currents and transport on parallel computers', *Paper submitted to the 4^{th} International Conference on Development and Application of Computer Techniques to Environmental Studies, Southampton UK September 1-3, 1992*, Laboratory of Hydraulics, Katholieke Universiteit Leuven, September 1992

[23] Swart J.P., 'Onderzoek oppervlakte-bemonstering slib in de Westerschelde periode '69–'80', RWS adviesdienst Vlissingen, NOTA WWKZ-82.V003, Vlissingen 1982

[24] Verlaan P.A.J. en R. Spanhoff, 'Slibconcentratieprofielen', Het bepalen van erosie- en sedimentatiesnelheden met behulp van een 1-D model, Dienst Getijdewateren Rijkswaterstaat, Den Haag, mei 1992

[25] Visser, M., 'On the transport of fine marine sediment in the Netherlands coastal zone', Ph.D.Thesis, Rijksuniversiteit Utrecht, november 1993.

List of Symbols

C	concentration of suspended sediments	$\mathrm{Kg\,m^{-3}}$
Ch	Chézy coefficient	$\mathrm{m^{\frac{1}{2}}\,s^{-1}}$
$\partial d/\partial t$	rate of sedimentation	$\mathrm{m^2\,s^{-1}}$
D_x, D_y	horizontal dispersion coefficients	$\mathrm{m^2\,s^{-1}}$
$\partial E/\partial t$	rate of erosion	$\mathrm{Kg\,m^{-2}\,s^{-1}}$
e_i	forecast error	
g	gravitation constant	$\mathrm{m\,s^{-2}}$
H	water depth, in this case: $17.4 + z$	m
$\mathcal{H}$	covariance matrix of the innovation sequence Z_k	
HW	high water turn	
$L(\mathbf{Y}_N; Q)$	likelihood function of Q	
LW	low water turn	
M	erosion constant	$\mathrm{Kg\,m^{-2}\,s^{-1}}$
$\mathcal{N}[\Phi]$	null space of Φ	
P_k	covariance matrix at time k	
$Q(k)$	process noise covariance matrix, time k	
$R(k)$	measurement noise covariance matrix, time k	
$\mathcal{R}[\Phi]$	range of space Φ	
r_i	autocorrelation function of the innovation sequence	
S	source term	$\mathrm{Kg\,m^{-2}\,s^{-1}}$
t	time	s
T	turbidity	$\mathrm{Kg\,m^{-3}}$
u	depth mean velocity component along x direction	$\mathrm{m\,s^{-1}}$
v	depth mean velocity component along y direction	$\mathrm{m\,s^{-1}}$
V	velocity in the x-direction	$\mathrm{m\,s^{-1}}$
V_k	measurement noise at time k	

W_k	system noise at time k	
W_s	settling velocity	$\mathrm{m\,s^{-1}}$
X_k	state at time k	
Y_k	observation at time k	
z	water level elevation related to mean-sea-level	m
Z_k	innovation function at time k	

α	time step	s
δ_{ij}	Kronecker delta function	
θ	unknown parameter	
ρ_i	normalized autocorrelation coefficient	
ρ_w	density of water	$\mathrm{Kg\,m^{-3}}$
τ_b	bed shear stress	$\mathrm{N\,m^{-2}}$
τ_{ce}	critical bed shear stress for erosion	$\mathrm{N\,m^{-2}}$
τ_{cd}	critical bed shear stress for sedimentation	$\mathrm{N\,m^{-2}}$
$\Phi^{\#}$	pseudo-inverse of Φ	

Appendix A

Estimation with identified Q

A.1 Identification of the system noise covariance matrix Q

We are going to identify the system noise covariance matrices $Q(i)$, $i = 5 \cdot \cdot 8$, needed for the estimation of the critical bed shear stresses for erosion and sedimentation, τ_{ce} and τ_{cd}.

Rough estimation of Q

Method

We assume the shape of Q

$$Q(i) = \begin{bmatrix} Q_{11}(i) & 0 \\ 0 & Q_{22}(i) \end{bmatrix} \tag{A.1}$$

where $i = 5 \cdot \cdot 8$, i.e. one Q_i for every unknown parameter: $\tau_{ce,ebb}$, $\tau_{ce,flood}$, $\tau_{cd,ebb}$ and $\tau_{cd,flood}$.

We start identifying Q_{11}, using the Maximum Likelihood method. We use the MATLAB-programs 'rit11' and 'quta11' (appendix C). We use the (initial) values

$$M_{ebb} = 2 \cdot 10^{-5} \quad ; \quad M_{flood} = 4 \cdot 10^{-5} \tag{A.2}$$

$$W_{s,ebb} = 1 \cdot 10^{-3} \quad ; \quad W_{s,flood} = 9 \cdot 10^{-4} \tag{A.3}$$

$$\tau_{ce,ebb}(0) = 0.14 \quad ; \quad \tau_{ce,flood}(0) = 0.20 \tag{A.4}$$

$$\tau_{cd,ebb}(0) = 0.06 \quad ; \quad \tau_{cd,flood}(0) = 0.08 \tag{A.5}$$

$$
\begin{aligned}
F_k &= I_2 && \text{(A.6)}\\
R(k) &= 0.7 \cdot 10^{-7} && \text{(A.7)}\\
Q_{22}(i) &= 1 \cdot 10^{-5} \qquad i = 1, 2, .., 4 && \text{(A.8)}
\end{aligned}
$$

After this, we identify Q_{22}, using the Maximum Likelihood method, where we assume $Q_{11}(i)$ as obtained in the previous identification. We use the MATLAB-programs 'rit22' and 'quta22'.

Results
We obtain the rough estimates for Q,

$$\hat{Q}(5) = \begin{bmatrix} 1.0 \cdot 10^{-4} & 0 \\ 0 & 1.0 \cdot 10^{-8} \end{bmatrix} \tag{A.9}$$

$$\hat{Q}(6) = \begin{bmatrix} 1.0 \cdot 10^{-4} & 0 \\ 0 & 1.0 \cdot 10^{-9} \end{bmatrix} \tag{A.10}$$

$$\hat{Q}(7) = \begin{bmatrix} 1.0 \cdot 10^{-8} & 0 \\ 0 & 1.0 \cdot 10^{-8} \end{bmatrix} \tag{A.11}$$

$$\hat{Q}(8) = \begin{bmatrix} 1.0 \cdot 10^{-7} & 0 \\ 0 & 1.0 \cdot 10^{-8} \end{bmatrix} \tag{A.12}$$

Estimation of Q_{11}

Method
We use the method suggested by Mehra to estimate $Q_{11}(i)$, $i = 5 \cdot\cdot 8$. This method uses a linear system

$$
\begin{aligned}
X_{k+1} &= AX_k + Gu(k) + FW_k && \text{(A.13)}\\
Y_k &= CX_k + V_k && \text{(A.14)}
\end{aligned}
$$

To estimate the unknown critical bed shear stresses $\tau_{ce,ebb}$ and $\tau_{ce,flood}$, we used Eqn (5.19) in chapter 5. Rewriting this equation we obtain

$$C_{k+1} = C_k + \frac{\alpha M}{H_k}\left(\frac{\tau_b(k)}{\tau_{ce}(k)} - 1\right) \qquad \tau_b(k) \geq \tau_{ce}(k) \tag{A.15}$$

This may be written as the system

$$
\begin{aligned}
C_{k+1} &= C_k + \frac{\alpha M}{H_k}\left(\frac{\tau_b(k)}{\tau_{ce}} - 1\right) + FW_k && \text{(A.16)}\\
Y_k &= C_k + V_k && \text{(A.17)}
\end{aligned}
$$

for a certain τ_{ce}. So we assume in Eqns (A.13)–(A.14)

$$
\begin{aligned}
u(k) &= \frac{\alpha M}{H_k}\left(\frac{\tau_b(k)}{\tau_{ce}} - 1\right) \\
A &= 1 \\
G &= 1 \\
F &= 1 \\
C &= 1
\end{aligned}
$$

To estimate the unknown critical bed shear stresses $\tau_{cd,ebb}$ and $\tau_{cd,flood}$, we used Eqn (5.24) in chapter 5. Rewriting this equation we obtain

$$
C_{k+1} = C_k + C_k\frac{\alpha W_s}{H_k}\left(\frac{\tau_b(k)}{\tau_{cd}(k)} - 1\right) \qquad \tau_b(k) \le \tau_{cd}(k) \tag{A.18}
$$

This may be written as the system

$$
\begin{aligned}
C_{k+1} &= \left[1 + \frac{\alpha W_s}{H_k}\left(\frac{\tau_b(k)}{\tau_{cd}(k)} - 1\right)\right] C_k + FW_k && \text{(A.19)} \\
Y_k &= C_k + V_k && \text{(A.20)}
\end{aligned}
$$

for a certain τ_{cd}. So we assume in Eqns (A.13)–(A.14)

$$
\begin{aligned}
A &= \frac{1}{N}\sum_{k=1}^{N}[1 + \frac{\alpha W_s}{H_k}(\frac{\tau_b(k)}{\tau_{cd}} - 1)] \\
F &= 1 \\
C &= 1 \\
G &= 0
\end{aligned}
$$

To obtain estimates of $Q_{11}(i)$, $i = 5 \cdot\cdot 8$, we need values for the unknown parameters $\tau_{ce,ebb}$, $\tau_{ce,flood}$, $\tau_{cd,ebb}$ and $\tau_{cd,flood}$. We determine these values with the extended Kalman filter and Eqns (A.9)–(A.12). We assume the initial values of Eqns (A.2)–(A.7). Using the MATLAB programs 'rkalf2', 'kalfil2' and 'kfmean2' we obtained values which were not reliable. Thus we decided to use

$$
\begin{aligned}
\tau_{ce,ebb} &= 0.14 \qquad & \tau_{ce,flood} &= 0.20 \\
\tau_{cd,ebb} &= 0.06 \qquad & \tau_{cd,flood} &= 0.08
\end{aligned}
$$

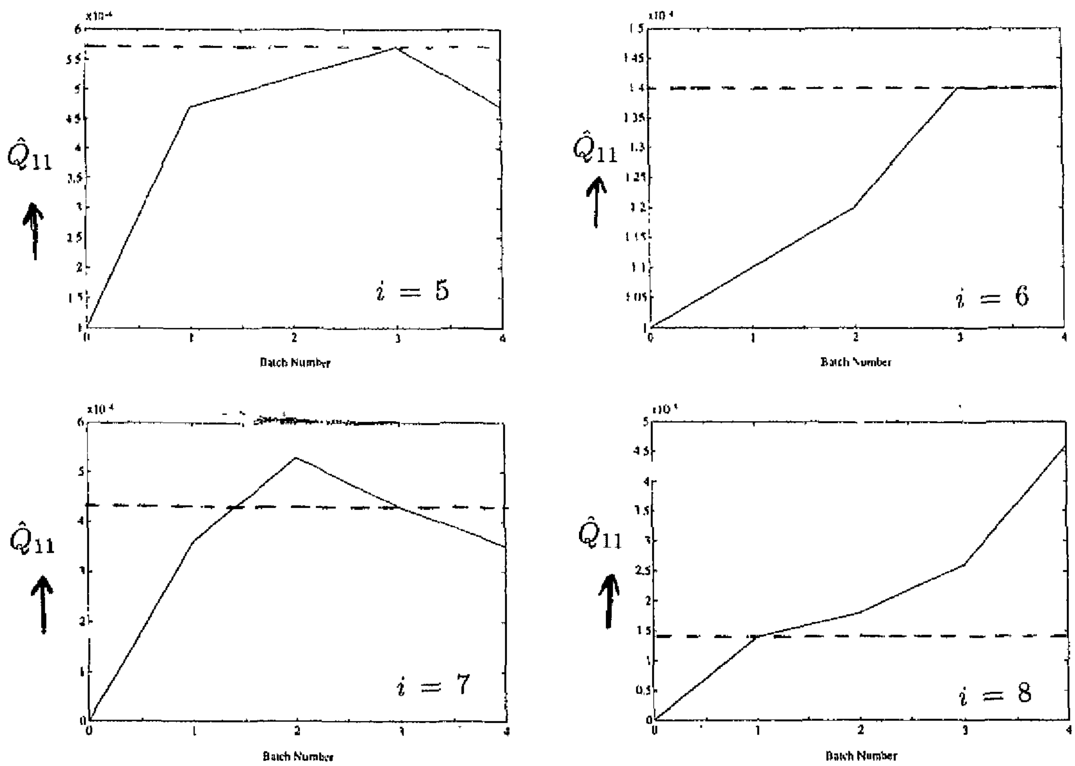


Figure A.1: Results of the identification of $Q_{11}(i)$ by the method of Mehra, for $i = 5 \cdot\cdot 8$

This is for the estimation of $\tau_{ce,ebb}$, $\tau_{ce,flood}$, $\tau_{cd,ebb}$, and $\tau_{cd,flood}$. Each batch consists of 430 points (except batch number 5; this one consists of 427 points).

We find table A.1, which is shown at the end of this chapter, and we conclude that:

$$\hat{Q}_{11}(5) = 5.7 \cdot 10^{-4} \tag{A.21}$$
$$\hat{Q}_{11}(6) = 1.4 \cdot 10^{-4} \tag{A.22}$$
$$\hat{Q}_{11}(7) = 4.3 \cdot 10^{-4} \tag{A.23}$$
$$\hat{Q}_{11}(8) = 1.4 \cdot 10^{-5} \tag{A.24}$$

(the iteration steps are shown in figure A.1)

Estimation of Q_{22}

Method

We want to obtain better estimates for $Q_{22}(i)$, $i = 5 \cdot\cdot 8$ by the Maximum Likelihood method. We use the (initial) values of Eqns (A.2)–(A.7):

$$\begin{aligned}
M_{ebb} = 2 \cdot 10^{-5} \quad &; \quad M_{flood} = 4 \cdot 10^{-5} \\
W_{s,ebb} = 1 \cdot 10^{-3} \quad &; \quad W_{s,flood} = 9 \cdot 10^{-4} \\
\tau_{ce,ebb}(0) = 0.14 \quad &; \quad \tau_{ce,flood}(0) = 0.20 \\
\tau_{cd,ebb}(0) = 0.06 \quad &; \quad \tau_{cd,flood}(0) = 0.08 \\
F_k \quad &= \quad I_2 \\
R(k) \quad &= \quad 0.7 \cdot 10^{-7}
\end{aligned}$$

For $Q_{11}(i)$, $i = 5 \cdot\cdot 8$ we use Eqns (A.21)–(A.24).

Results

We assume $Q_{22} \subset \Theta$, $\Theta = [1.0 \cdot 10^{-9}, 1.0]$, and we find the approximation of Q_{22}:

$$\begin{aligned}
\hat{Q}_{22}(1) &= 1.0 \cdot 10^{-9} \\
\hat{Q}_{22}(2) &= 1.0 \cdot 10^{-8} \\
\hat{Q}_{22}(3) &= 1.0 \cdot 10^{-8} \\
\hat{Q}_{22}(4) &= 1.0 \cdot 10^{-8}
\end{aligned}$$

So we obtained for the system noise covariance matrices $Q(i)$, $i = 5 \cdot\cdot 8$ (needed for the estimation of the critical bed shear stress for erosion τ_{ce} and the critical bed shear stress for sedimentation τ_{cd}, both in the cases of ebb and flood),

$$\hat{Q}(5) = \begin{bmatrix} 5.7 \cdot 10^{-4} & 0 \\ 0 & 1.0 \cdot 10^{-9} \end{bmatrix} \tag{A.25}$$

$$\hat{Q}(6) = \begin{bmatrix} 1.4 \cdot 10^{-4} & 0 \\ 0 & 1.0 \cdot 10^{-8} \end{bmatrix} \tag{A.26}$$

$$\hat{Q}(7) = \begin{bmatrix} 4.3 \cdot 10^{-4} & 0 \\ 0 & 1.0 \cdot 10^{-8} \end{bmatrix} \tag{A.27}$$

$$\hat{Q}(8) = \begin{bmatrix} 1.4 \cdot 10^{-5} & 0 \\ 0 & 1.0 \cdot 10^{-8} \end{bmatrix} \tag{A.28}$$

A.2 Estimation of τ_{ce} and τ_{cd}, using the identified Q

Method

To estimate the unknown critical bed shear stresses for erosion $\tau_{ce,ebb}$ and $\tau_{ce,flood}$ and the unknown critical bed shear stresses for sedimentation $\tau_{cd,ebb}$ and $\tau_{cd,flood}$, we use the MATLAB programs 'rkalf2', 'kalfil2', and 'kfmean2' and the (initial) values (Eqns (A.2)–(A.7)):

$$\begin{aligned} M_{ebb} = 2 \cdot 10^{-5} \quad &; \quad M_{flood} = 4 \cdot 10^{-5} \\ W_{s,ebb} = 1 \cdot 10^{-3} \quad &; \quad W_{s,flood} = 9 \cdot 10^{-4} \\ \tau_{ce,ebb}(0) = 0.14 \quad &; \quad \tau_{ce,flood}(0) = 0.20 \\ \tau_{cd,ebb}(0) = 0.06 \quad &; \quad \tau_{cd,flood}(0) = 0.08 \\ F_k &= I_2 \\ R(k) &= 0.7 \cdot 10^{-7} \end{aligned}$$

The system noise covariance matrices $Q(i)$ are described by Eqns (A.25)-(A.28).

Results

The estimates are shown in figure 8.3, in chapter 8. To obtain a value for the estimated parameters, we take the average value of the estimates, started at the 1000^{th} estimate. We conclude

$$\hat{\tau}_{ce,ebb} = -0.33 \tag{A.29}$$

$$\hat{\tau}_{ce,flood} = 0.89 \tag{A.30}$$

$$\hat{\tau}_{cd,ebb} = 0.14 \tag{A.31}$$

$$\hat{\tau}_{cd,flood} = 0.26 \tag{A.32}$$

Model Number i	Number of Iterations	$\hat{Q}^{*_1}$	Likelihood Function $L(\mathbf{Y}_N;\hat{Q})$	Percentage *3
1	0	$1.0 \cdot 10^{-4}$	1.92	0.03
	1	$4.7 \cdot 10^{-4}$	6.16	0.10
	2	$5.2 \cdot 10^{-4}$	6.44	0.07
	3	$5.7 \cdot 10^{-4}$	7.04	0.00
	4	$4.7 \cdot 10^{-4}$	2.23	0.00
		$7.1 \cdot 10^{-4}$		
2	0	$1.0 \cdot 10^{-4}$	8.44	0.73
	1	$1.1 \cdot 10^{-4}$	8.33	0.83
	2	$1.2 \cdot 10^{-4}$	7.93	0.23
	3	$1.4 \cdot 10^{-4}$	8.16	0.07
	4	$1.4 \cdot 10^{-4}$	6.16	0.03
		$1.9 \cdot 10^{-4}$		
3	0	$1.0 \cdot 10^{-8}$	$-9.96 \cdot 10^{-3}$	0.17
	1	$3.6 \cdot 10^{-4}$	6.45	0.13
	2	$5.3 \cdot 10^{-4}$	7.25	0.10
	3	$4.3 \cdot 10^{-4}$	7.50	0.07
	4	$3.5 \cdot 10^{-4}$	- 0.01	0.17
		$6.4 \cdot 10^{-4}$		
4	0	$1.0 \cdot 10^{-7}$	-82.50	0.03
	1	$1.4 \cdot 10^{-5}$	9.55	0.03
	2	$1.8 \cdot 10^{-5}$	7.15	0.00
	3	$2.6 \cdot 10^{-5}$	5.09	0.07
	4	$4.6 \cdot 10^{-5}$	9.14	0.00
		$4.4 \cdot 10^{-4}$		

Table A.1: Results of the method of Mehra, to estimate the system noise covariance matrices $Q(i)$, $i = 5 \cdot\cdot 8$

$*_1$ *The A.mse and the C.mse are equal to the values of $\hat{Q}$, everywhere.*
$*_2$: *Percentage of points lying outside the 95 percent confidence limits.*

Appendix B

National Institute for Coastal and Marine Management

B.1 Organization

This study is carried out at the Dutch 'National Institute for Coastal and Marine Management' (RIKZ). The institute has offices at The Hague, at Haren (Gr.), and at Middelburg. This study is done at Middelburg. As a part of the Ministry of Transport, Public Works and Water Management, this institute provides advice and information on:

- the sustainable use of estuaries, coasts and seas;
- coastal flood protection.

For this purpose, the institute develops and maintains a knowledge and information infrastructure. As a knowledge bank, this institute is also at the service of other parts of the central government and it cooperates with various agencies and organizations at international level. As a part of the Ministry, the institute's main client is the Directorate-General of Public Works and Water Management, while the institute receives commissions in relation to policy implementation from regional departments such as those for Zeeland, South Holland, North Holland, the North Sea and the Northern Netherlands (figure B.1).

The institute is subdivided into several departments, according to table B.1. This investigation is done at the department of 'Research and Strategy; Physics'.

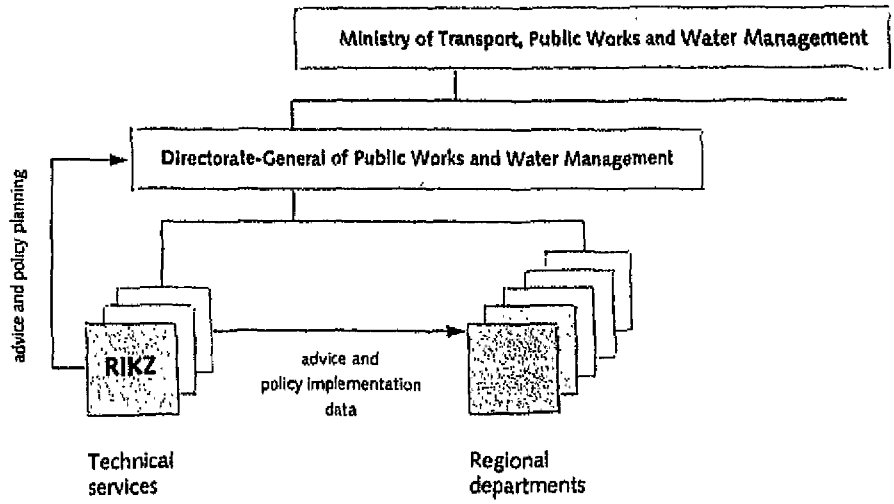


Figure B.1: Survey of the division parts of RIKZ

National Institute for Coastal and Marine Management *RIKZ*		
Division Secretariat and General Services Controller Section Personnel Section		
Research and Strategy	**Consultance and Policy Analysis**	**Information and Technology**
BEON Program Bureau	National Affairs	Information Systems
Physics	North Sea	Hydro-Instrumentation
Biology	Wadden Sea	Laboratories
Chemistry	Delta area	Information technology

Table B.1: RIKZ organization chart

Most of the research at the institute is done within the framework of projects. This study is about the transport of cohesive sediment and therefore it may be interesting for the projects SCHOON (in English "clean") and OOST-WEST (in English "east-west").

B.2 Projects

The project SCHOON is a joint operation by the Directorate Zeeland and the Tidal Waters Division of the Netherlands Directorate-General of Public Works and Water Management. It was started at the beginning of 1991. The aim of the project is to carry out research on the consequences of the cleaning-up activities for concentrations of substances, the waste load and the processes taking place in the estuary.

The project OOSTWEST has also been started in 1991. The aim of the project is to develop a sustainable and ecologically sound estuary. This development has been hampered by extensive reclamations in the past, resulting in a shortage of natural flood plains, and exacerbated by intensive dredging in the present. A sustainable management of the physical system includes

- a development of an integrated dredging-extraction-dumping strategy
- the restoration and creation of flood plain areas and
- selected measures to enlarge rare habitat types such as saline marshlands and freshwater tidal areas.

This study was also interesting for the project TROEBEL (in English 'turbidity'). This project is examining the question to what extent human people are able, wanted or not, to influence the turbidity and the sediment concentration of a salt water system. The project TROEBEL is subdivided into two phases. The first phase has just finished. The second phase will be finished in the Autumn of '94 and will pronounce upon formulating reference values and a measure and check strategy for the turbidity in salt waters.

B.3 Field of activity

Computer facilities

Since I was the only mathematician at the institute at Middelburg, not much mathematical software was available. So I had to use the software of the mathematicians of the department at The Hague. After programming in FORTRAN, I started to work with a simple version of MATLAB. I also had the disposal of the LATEX program of The Hague. This program was totally unknown at Middelburg, so was MATLAB. At the start of this case, I used the personal computer of my supervisor, which I extended with 2 MB.

Supply of data

I had to gather the data I used from several departments. Almost all data were delivered in ASCII form. I converted them, using LOTUS programs, in a way I could use them in FORTRAN or MATLAB programs. Most of the data were delivered soon.

Sidelines

Except doing my research, I was asked to give some advice about time-series cases. I also attended a meeting within the framework of my investigation and I spoke to some people about my investigation, at Middelburg as well as at The Hague.

I had the possibility to sail with a ship of the geometrical service from Hansweert to Rupelmonde, where measurements were done. I visited the mud flats of Vianen and the South salting, and I enjoyed an excursion to the eastern part of the Flooded Land of Saeftinge. One morning I visited the pontoon with which the turbidity measurements, I used, were carried out.

Finally, I wrote a proposal to continue this study, because of the promising results so far. At this time this proposal is being considered at the Ministry. The proposal is added as a loose supplement to this report.

Appendix C

Listings

FORTRAN programs				
	$N = 343$ $H = 1$ $\Delta t = 1$	$N = 701$ $H = 1$ $\Delta t = 1$	$N = 701$ H is variable $\Delta t = 1$	$N = 701$ H is variable $\Delta t = \alpha = 724$
M and W_s	kfmwc1		kfhmw	
τ_{ce} and τ_{cd}	kftau1			
M and τ_{ce}		kftem	kfhtem	
Mathematical model	model1			
M	kfmc1			
τ_{ce}	kftaue			
M and W_s, ebb and flood				kfhmw'
τ_{ce} and τ_{cd}, ebb and flood			kfhtt	

Table C.1: Summary of the used FORTRAN programs

where M: *erosion constant*
W_s: *settling velocity*
τ_{ce}: *critical bed shear stress for erosion*
τ_{cd}: *critical bed shear stress for sedimentation*
N: *number of observations*
H: *water level*
Δt: *time step*

MATLAB programs				
	ML Q_{11}	ML Q_{22}	Mehra Q_{11}	Kalman filter
M and W_s	ri11 qu11	ri22 qu22	rmeh1 mehra3	rkalf kalfil kfmean
τ_{ce} and τ_{cd}	rit11 quta11	ri22 quta22	rmeh1 mehra3	rkalf2 kalfil2 kfmean2
mathematical model	rgeg, model			

Table C.2: Summary of the used MATLAB programs

where M: *erosion constant*
W_s: *settling velocity*
τ_{ce}: *critical bed shear stress for erosion*
τ_{cd}: *critical bed shear stress for sedimentation*
ML: *using the Maximum Likelihood method*
Mehra: *using the method of Mehra*

```
      PROGRAM KFMWC1
      PARAMETER (NOBS=343)
C
      INTEGER I
      REAL R(1,1), QM(1,1), QW(1,1), AM(2,2), AW(2,2), XM(2), XW(2)
      REAL X(2), PM(2,2), PW(2,2), Y(1), K(2), M(2,2)
      REAL YDATA(NOBS), HDATA(NOBS), V2DATA(NOBS)
      REAL XNEW(2), KNEW(2), ENEW(2,2), PNEW(2,2), GNEW(2,2)
      REAL HNEW(2,2), TAUB, TAUE, TAUD, TERM
C
      OPEN(UNIT=100,FILE='CONC',STATUS='OLD')
      OPEN(UNIT=200,FILE='HOOGTE',STATUS='OLD')
      OPEN(UNIT=300,FILE='VOPP2',STATUS='OLD')
      OPEN(UNIT=400,FILE='UITMW1',STATUS='NEW')
      OPEN(UNIT=500,FILE='EXTRA1',STATUS='NEW')
      READ(100,*,END=tt) YDATA
      READ(200,*,END=tt) HDATA
      READ(300,*,END=tt) V2DATA
C                                          values
      TAUE    = 0.14
      TAUD    = 0.08
      R(1,1)  = 2.5E-5
      QM(1,1) = 5E-4
      QW(1,1) = 5E-4
      AM(1,1) = 1.0
      AM(2,1) = 0.0
      AM(2,2) = 1.0
      AW(2,1) = 0.0
      AW(2,2) = 1.0
      M(1,1)  = 1.0
      M(1,2)  = 0.0
      M(2,1)  = 0.0
      M(2,2)  = 0.0
C                                          initial values
      XM(1)   = YDATA(1)
      XM(2)   = 2.0E-3
      XW(1)   = YDATA(1)
      XW(2)   = 3.0E-4
      X(1)    = YDATA(1)
      PM(1,1) = 2.5E-5
      PM(1,2) = 1E-5
      PM(2,1) = 1E-5
      PM(2,2) = 4E-6
      PW(1,1) = 2.5E-5
      PW(1,2) = 5E-7
      PW(2,1) = 5E-7
      PW(2,2) = 1E-8
C
      WRITE(400,99998)
      WRITE(500,99998)
      WRITE(400,99999) 1, 0, ' E', XM(1), XM(2), XW(2), PM(1,1),
     &                 PM(1,2), PM(2,2)
      WRITE(400,99999) 1, 0, ' S', XW(1), XM(2), XW(2), PW(1,1),
     &                 PW(1,2), PW(2,2)
C
      DO 10 I=2, NOBS
C
         Y(1)    = YDATA(I)
         TAUB    = (4.831*V2DATA(I))*((HDATA(I)-1)**(-0.30))
         IF (TAUB.GE.TAUE) THEN
C                                          update
            XM(1)    = X(1)
            CALL KALM1(PM,R,XM,Y,M,KNEW,XNEW,ENEW,PNEW)
            XM(1)    = XNEW(1)
            XM(2)    = XNEW(2)
            PM(1,1) = PNEW(1,1)
            PM(1,2) = PNEW(1,2)
            PM(2,1) = PNEW(2,1)
            PM(2,2) = PNEW(2,2)
            WRITE(400,99999) I-1, I-1, ' E', XM(1), XM(2), XW(2),
     &                       PM(1,1), PM(1,2), PM(2,2),
     &                       KNEW(1), KNEW(2)
C                                          prediction
            AM(1,2) = (TAUB)/(TAUE) - 1
            CALL KALM2(XM,AM,PM,QM,XNEW,HNEW,GNEW,PNEW)
            XM(1)    = XNEW(1)
            XM(2)    = XNEW(2)
            PM(1,1) = PNEW(1,1)
            PM(1,2) = PNEW(1,2)
            PM(2,1) = PNEW(2,1)
            PM(2,2) = PNEW(2,2)
            WRITE(500,99999) I, I-1, ' E', XM(1), XM(2), XW(2), PM(1,1),
     &                       PM(1,2), PM(2,2)
            X(1)     = XM(1)
         ELSE
            IF (TAUB.LE.TAUD) THEN
C                                          update
               XW(1)    = X(1)
               CALL KALM1(PW,R,XW,Y,M,KNEW,XNEW,ENEW,PNEW)
               XW(1)    = XNEW(1)
               XW(2)    = XNEW(2)
               PW(1,1) = PNEW(1,1)
               PW(1,2) = PNEW(1,2)
               PW(2,1) = PNEW(2,1)
               PW(2,2) = PNEW(2,2)
               WRITE(400,99999) I-1, I-1, ' S', XW(1), XM(2), XW(2),
     &                          PW(1,1), PW(1,2), PW(2,2),
     &                          KNEW(1), KNEW(2)
C                                          prediction
               TERM     = (TAUB)/(TAUD) - 1
               AW(1,1) = 1 + XW(2)*TERM
               AW(1,2) = XW(1)*TERM
               CALL KALM3(XW,TERM,AW,PW,QW,XNEW,HNEW,GNEW,PNEW)
               XW(1)    = XNEW(1)
               XW(2)    = XNEW(2)
               PW(1,1) = PNEW(1,1)
               PW(1,2) = PNEW(1,2)
               PW(2,1) = PNEW(2,1)
               PW(2,2) = PNEW(2,2)
               WRITE(500,99999) I, I-1, ' S', XW(1), XM(2), XW(2),
     &                       PW(1,1), PW(1,2), PW(2,2)
               X(1)     = XW(1)
            ELSE
               WRITE(400,99999) I-1, I-1, ' N', X(1), XM(2), XW(2)
               WRITE(500,99999) I, I-1, ' N', X(1), XM(2), XW(2)
            ENDIF
         ENDIF
   10 CONTINUE
C
99998 FORMAT (' k/j  ', 'ES',' C(k/j)', '  M(k/j)', ' W(k/j) ',
     &        'P(k/j)11', 'P(k/j)12', 'P(k/j)22',
     &        ' K1(k) ', ' K2(k)')
99999 FORMAT (I4, '/', I3, A2, 8F8.6)
      END
C
C
C
C
C
C
C ******************************************************************
C
      SUBROUTINE KALM1(PIN,RIN,XIN,YIN,MIN,KUIT,XUIT,EUIT,PUIT)
C                                          calculates X(k+1),
C                                          K(k+1) and P(k+1)
      REAL PIN(2,2), RIN(1,1), QIN(1,1), XIN(2), YIN(1), MIN(2,2)
      REAL KUIT(2), XUIT(2), EUIT(2,2), PUIT(2,2)
C
      KUIT(1)   = PIN(1,1)/(PIN(1,1)+RIN(1,1))
      KUIT(2)   = PIN(2,1)/(PIN(1,1)+RIN(1,1))
      XUIT(1)   = XIN(1) + KUIT(1)*(YIN(1)-XIN(1))
      XUIT(2)   = XIN(2) + KUIT(2)*(YIN(1)-XIN(1))
      EUIT(1,1) = VMEE(PIN,MIN)
      EUIT(1,2) = VMET(PIN,MIN)
      EUIT(2,1) = VMTE(PIN,MIN)
      EUIT(2,2) = VMTT(PIN,MIN)
      PUIT(1,1) = PIN(1,1) - VMEE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(1,2) = PIN(1,2) - VMET(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,1) = PIN(2,1) - VMTE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,2) = PIN(2,2) - VMTT(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      RETURN
      END
C
      SUBROUTINE KALM2(XIN,AIN,PIN,QIN,XUIT,HUIT,GUIT,PUIT)
C                                          calculates X(k+1/k) and
C                                          P(k+1/k)
      REAL XIN(2), AIN(2,2), PIN(2,2), QIN(1,1), XUIT(2), HUIT(2,2)
      REAL GUIT(2,2), PUIT(2,2)
C
      XUIT(1)   = VMVECE(AIN,XIN)
      XUIT(2)   = VMVECT(AIN,XIN)
      HUIT(1,1) = AIN(1,1)
      HUIT(1,2) = AIN(2,1)
      HUIT(2,1) = AIN(1,2)
      HUIT(2,2) = AIN(2,2)
      GUIT(1,1) = VMEE(PIN,HUIT)
      GUIT(1,2) = VMET(PIN,HUIT)
      GUIT(2,1) = VMTE(PIN,HUIT)
      GUIT(2,2) = VMTT(PIN,HUIT)
      PUIT(1,1) = VMEE(AIN,GUIT) + QIN(1,1)
      PUIT(1,2) = VMET(AIN,GUIT)
      PUIT(2,1) = VMTE(AIN,GUIT)
      PUIT(2,2) = VMTT(AIN,GUIT)
      RETURN
      END
C
      SUBROUTINE KALM3(XIN,TIN,AIN,PIN,QIN,XUIT,HUIT,GUIT,PUIT)
C                                          calculates X(k+1/k) and
C                                          P(k+1/k)
      REAL XIN(2), AIN(2,2), TIN, PIN(2,2), QIN(1,1), XUIT(2)
      REAL HUIT(2,2), GUIT(2,2), PUIT(2,2)
C
      XUIT(1)   = VMVECE(AIN,XIN) - XIN(1)*XIN(2)*TIN
      XUIT(2)   = VMVECT(AIN,XIN)
      HUIT(1,1) = AIN(1,1)
      HUIT(1,2) = AIN(2,1)
      HUIT(2,1) = AIN(1,2)
      HUIT(2,2) = AIN(2,2)
      GUIT(1,1) = VMEE(PIN,HUIT)
      GUIT(1,2) = VMET(PIN,HUIT)
      GUIT(2,1) = VMTE(PIN,HUIT)
      GUIT(2,2) = VMTT(PIN,HUIT)
      PUIT(1,1) = VMEE(AIN,GUIT) + QIN(1,1)
      PUIT(1,2) = VMET(AIN,GUIT)
      PUIT(2,1) = VMTE(AIN,GUIT)
      PUIT(2,2) = VMTT(AIN,GUIT)
      RETURN
      END
C
C ******************************************************************
C                                          multiply matrices
      FUNCTION VMEE(B,D)
      REAL B(2,2), D(2,2)
      VMEE = B(1,1)*D(1,1) + B(1,2)*D(2,1)
      RETURN
      END
C
      REAL FUNCTION VMET(B,D)
      REAL B(2,2), D(2,2)
      VMET = B(1,1)*D(1,2) + B(1,2)*D(2,2)
      RETURN
      END
C
      REAL FUNCTION VMTE(B,D)
      REAL B(2,2), D(2,2)
      VMTE = B(2,1)*D(1,1) + B(2,2)*D(2,1)
      RETURN
      END
C
      REAL FUNCTION VMTT(B,D)
      REAL B(2,2), D(2,2)
      VMTT = B(2,1)*D(1,2) + B(2,2)*D(2,2)
      RETURN
      END
C
C                                          mat*vector
      REAL FUNCTION VMVECE(B,Z)
      REAL B(2,2), Z(2)
      VMVECE = B(1,1)*Z(1) + B(1,2)*Z(2)
      RETURN
      END
C
      REAL FUNCTION VMVECT(B,Z)
      REAL B(2,2), Z(2)
      VMVECT = B(2,1)*Z(1) + B(2,2)*Z(2)
      RETURN
      END
```

```fortran
      PROGRAM KFTAU1
      PARAMETER (NOBS=343)
C
      INTEGER I
      REAL R(1,1), QM(1,1), QW(1,1), AM(2,2), AW(2,2), XM(2), XW(2)
      REAL X(2), PM(2,2), PW(2,2), Y(1), K(2), M(2,2)
      REAL YDATA(NOBS), HDATA(NOBS), V2DATA(NOBS)
      REAL XNEW(2), KNEW(2), ENEW(2,2), PNEW(2,2), GNEW(2,2)
      REAL HNEW(2,2), TAUB, WDAK, MDAK
C
      OPEN(UNIT=100,FILE='CONC',STATUS='OLD')
      OPEN(UNIT=200,FILE='HOOGTE',STATUS='OLD')
      OPEN(UNIT=300,FILE='VOPP2',STATUS='OLD')
      OPEN(UNIT=400,FILE='UITTAU2',STATUS='NEW')
      OPEN(UNIT=500,FILE='ERBIJ2',STATUS='NEW')
      READ(100,*,END=tt)YDATA
      READ(200,*,END=tt)HDATA
      READ(300,*,END=tt)V2DATA
C                                          values
      MDAK      = 0.2E-4
      WDAK      = 0.3E-3
      R(1,1)    = 2.5E-5
      QM(1,1)   = 5E-4
      QW(1,1)   = 5E-4
      AM(1,1)   = 1.0
      AM(2,1)   = 0.0
      AM(2,2)   = 1.0
      AW(2,1)   = 0.0
      AW(2,2)   = 1.0
      M(1,1)    = 1.0
      M(1,2)    = 0.0
      M(2,1)    = 0.0
      M(2,2)    = 0.0
C                                          initial values
      XM(1)     = YDATA(1)
      XM(2)     = 0.12
      XW(1)     = YDATA(1)
      XW(2)     = 0.08
      X(1)      = YDATA(1)
      PM(1,1)   = 2.5E-5
      PM(1,2)   = 3E-4
      PM(2,1)   = 3E-4
      PM(2,2)   = 3.6E-3
      PW(1,1)   = 2.5E-5
      PW(1,2)   = 1E-4
      PW(2,1)   = 1E-4
      PW(2,2)   = 4E-4
C
       WRITE(400,99998)
       WRITE(500,99998)
       WRITE(400,99999) 1, 0, ' E', XM(1), XM(2), XW(2), PM(1,1),
     &                  PM(1,2), PM(2,2)
       WRITE(400,99999) 1, 0, ' S', XW(1), XM(2), XW(2), PW(1,1),
     &                  PW(1,2), PW(2,2)
C
       DO 10 I=2, NOBS
C
           Y(1)     = YDATA(I)
           TAUB     = (4.831*V2DATA(I))*((HDATA(I)-1)**(-0.30))
           IF (TAUB.GE.XM(2)) THEN
C                                          update
               XM(1)     = X(1)
               CALL KALM1(PM,R,XM,Y,M,KNEW,XNEW,ENEW,PNEW)
               XM(1)     = XNEW(1)
               XM(2)     = XNEW(2)
               PM(1,1) = PNEW(1,1)
               PM(1,2) = PNEW(1,2)
               PM(2,1) = PNEW(2,1)
               PM(2,2) = PNEW(2,2)
               WRITE(400,99999) I-1, I-1, ' E', XM(1), XM(2), XW(2),
     &                          PM(1,1), PM(1,2), PM(2,2),
     &                          KNEW(1), KNEW(2)
C                                          prediction
               AM(1,2) = (-1)*MDAK*TAUB*((XM(2))**(-2))
               CALL KALM4(XM,MDAK,TAUB,AM,PM,QM,XNEW,HNEW,GNEW,PNEW)
               XM(1)     = XNEW(1)
               XM(2)     = XNEW(2)
               PM(1,1) = PNEW(1,1)
               PM(1,2) = PNEW(1,2)
               PM(2,1) = PNEW(2,1)
               PM(2,2) = PNEW(2,2)
               WRITE(500,99999) I, I-1, ' E', XM(1), XM(2), XW(2), PM(1,1),
     &                          PM(1,2), PM(2,2)
               X(1)      = XM(1)
           ELSE
               IF (TAUB.LE.XW(2)) THEN
C                                          update
                   XW(1)     = X(1)
                   CALL KALM1(PW,R,XW,Y,M,KNEW,XNEW,ENEW,PNEW)
                   XW(1)     = XNEW(1)
                   XW(2)     = XNEW(2)
                   PW(1,1) = PNEW(1,1)
                   PW(1,2) = PNEW(1,2)
                   PW(2,1) = PNEW(2,1)
                   PW(2,2) = PNEW(2,2)
                   WRITE(400,99999) I-1, I-1, ' S', XW(1), XM(2), XW(2),
     &                              PW(1,1), PW(1,2), PW(2,2),
     &                              KNEW(1), KNEW(2)
C                                          prediction
                   AW(1,1) = 1 + WDAK*((TAUB)/(XW(2))-1)
                   AW(1,2) = (-1)*XW(1)*WDAK*TAUB*((XW(2))**(-2))
                   CALL KALM5(XW,WDAK,TAUB,AW,PW,QW,XNEW,HNEW,GNEW,PNEW)
                   XW(1)     = XNEW(1)
                   XW(2)     = XNEW(2)
                   PW(1,1) = PNEW(1,1)
                   PW(1,2) = PNEW(1,2)
                   PW(2,1) = PNEW(2,1)
                   PW(2,2) = PNEW(2,2)
                   WRITE(500,99999) I, I-1, ' S', XW(1), XM(2), XW(2),
     &                              PW(1,1), PW(1,2), PW(2,2)
                   X(1)      = XW(1)
               ELSE
                   WRITE(400,99999) I-1, I-1, ' N', X(1), XM(2), XW(2)
                   WRITE(500,99999) I, I-1, ' N', X(1), XM(2), XW(2)
               ENDIF
           ENDIF
   10 CONTINUE
C
99998 FORMAT ('  k/j  ', 'ES','  C(k/j)', '    M(k/j)', '  W(k/j) ',
     &        'P(k/j)11', 'P(k/j)12', 'P(k/j)22',
     &        ' K1(k) ', ' K2(k)')
99999 FORMAT (I4, '/', I3, A2, 8F8.6)
      END
C
C
C
C
C
C
C ************************************************************************
C
      SUBROUTINE KALM1(PIN,RIN,XIN,YIN,MIN,KUIT,XUIT,EUIT,PUIT)
C                                          calculates X(k+1),
C                                          X(k+1) and P(k+1)
      REAL PIN(2,2), RIN(1,1), QIN(1,1), XIN(2), YIN(1), MIN(2,2)
      REAL KUIT(2), XUIT(2), EUIT(2,2), PUIT(2,2)
C
      KUIT(1)   = PIN(1,1)/(PIN(1,1)+RIN(1,1))
      KUIT(2)   = PIN(2,1)/(PIN(1,1)+RIN(1,1))
      XUIT(1)   = XIN(1) + KUIT(1)*(YIN(1)-XIN(1))
      XUIT(2)   = XIN(2) + KUIT(2)*(YIN(1)-XIN(1))
      EUIT(1,1) = VMEE(PIN,MIN)
      EUIT(1,2) = VMET(PIN,MIN)
      EUIT(2,1) = VMTE(PIN,MIN)
      EUIT(2,2) = VMTT(PIN,MIN)
      PUIT(1,1) = PIN(1,1) - VMEE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(1,2) = PIN(1,2) - VMET(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,1) = PIN(2,1) - VMTE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,2) = PIN(2,2) - VMTT(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      RETURN
      END
C
      SUBROUTINE KALM4(XIN,MHOED,BTAU,AIN,PIN,QIN,XUIT,HUIT,GUIT,PUI
C                                          calculates X(k+1/k) and
C                                          P(k+1/k)
      REAL XIN(2), AIN(2,2), MHOED, BTAU, PIN(2,2), QIN(1,1), XUIT(2
      REAL HUIT(2,2), GUIT(2,2), PUIT(2,2)
C
      XUIT(1)   = VMVECE(AIN,XIN) + (2*MHOED*BTAU)/(XIN(2)) - MHOED
      XUIT(2)   = VMVECT(AIN,XIN)
      HUIT(1,1) = AIN(1,1)
      HUIT(1,2) = AIN(2,1)
      HUIT(2,1) = AIN(1,2)
      HUIT(2,2) = AIN(2,2)
      GUIT(1,1) = VMEE(PIN,HUIT)
      GUIT(1,2) = VMET(PIN,HUIT)
      GUIT(2,1) = VMTE(PIN,HUIT)
      GUIT(2,2) = VMTT(PIN,HUIT)
      PUIT(1,1) = VMEE(AIN,GUIT) + QIN(1,1)
      PUIT(1,2) = VMET(AIN,GUIT)
      PUIT(2,1) = VMTE(AIN,GUIT)
      PUIT(2,2) = VMTT(AIN,GUIT)
      RETURN
      END
C
      SUBROUTINE KALM5(XIN,WHOED,BTAU,AIN,PIN,QIN,XUIT,HUIT,GUIT,PUI
C                                          calculates X(k+1/k) and
C                                          P(k+1/k)
      REAL XIN(2), AIN(2,2), WHOED, BTAU, PIN(2,2), QIN(1,1), XUIT(2
      REAL HUIT(2,2), GUIT(2,2), PUIT(2,2)
C
      XUIT(1)   = VMVECE(AIN,XIN) + (XIN(1)*WHOED*BTAU)/(XIN(2))
      XUIT(2)   = VMVECT(AIN,XIN)
      HUIT(1,1) = AIN(1,1)
      HUIT(1,2) = AIN(2,1)
      HUIT(2,1) = AIN(1,2)
      HUIT(2,2) = AIN(2,2)
      GUIT(1,1) = VMEE(PIN,HUIT)
      GUIT(1,2) = VMET(PIN,HUIT)
      GUIT(2,1) = VMTE(PIN,HUIT)
      GUIT(2,2) = VMTT(PIN,HUIT)
      PUIT(1,1) = VMEE(AIN,GUIT) + QIN(1,1)
      PUIT(1,2) = VMET(AIN,GUIT)
      PUIT(2,1) = VMTE(AIN,GUIT)
      PUIT(2,2) = VMTT(AIN,GUIT)
      RETURN
      END
C
C ************************************************************************
C                                          multiply matrices
      FUNCTION VMEE(B,D)
      REAL B(2,2), D(2,2)
      VMEE = B(1,1)*D(1,1) + B(1,2)*D(2,1)
      RETURN
      END
C
      REAL FUNCTION VMET(B,D)
      REAL B(2,2), D(2,2)
      VMET = B(1,1)*D(1,2) + B(1,2)*D(2,2)
      RETURN
      END
C
      REAL FUNCTION VMTE(B,D)
      REAL B(2,2), D(2,2)
      VMTE = B(2,1)*D(1,1) + B(2,2)*D(2,1)
      RETURN
      END
C
      REAL FUNCTION VMTT(B,D)
      REAL B(2,2), D(2,2)
      VMTT = B(2,1)*D(1,2) + B(2,2)*D(2,2)
      RETURN
      END
C
C                                          mat*vector
      REAL FUNCTION VMVECE(B,Z)
      REAL B(2,2), Z(2)
      VMVECE = B(1,1)*Z(1) + B(1,2)*Z(2)
      RETURN
      END
C
      REAL FUNCTION VMVECT(B,Z)
      REAL B(2,2), Z(2)
      VMVECT = B(2,1)*Z(1) + B(2,2)*Z(2)
      RETURN
      END
```

```
[redacted] PROGRAM KFMC1
      PARAMETER (NOBS=343)
C
      INTEGER I
      REAL R(1,1), QM(1,1), QW(1,1), AM(2,2), AW(1,1), XM(2), XW(1)
      REAL X(2), PM(2,2), PW(1,1), Y(1), K(2), M(2,2)
      REAL YDATA(NOBS), HDATA(NOBS), V2DATA(NOBS)
      REAL XNEW(2), KNEW(2), ENEW(2,2), PNEW(2,2), GNEW(2,2)
      REAL HNEW(2,2), TAUB, TAUE, TAUD, VERM, W
C
      OPEN(UNIT=100,FILE='CONC',STATUS='OLD')
      OPEN(UNIT=200,FILE='HOOGTE',STATUS='OLD')
      OPEN(UNIT=300,FILE='VOPP2',STATUS='OLD')
      OPEN(UNIT=400,FILE='UITN1',STATUS='NEW')
      OPEN(UNIT=500,FILE='OOKM1',STATUS='NEW')
      READ(100,*,END=tt)YDATA
      READ(200,*,END=tt)HDATA
      READ(300,*,END=tt)V2DATA
C                                               values
      TAUE     = 0.060
      TAUD     = 0.059
      W        = 0.3E-3
      R(1,1)   = 2.5E-5
      QM(1,1)  = 5E-4
      QW(1,1)  = 5E-4
      AM(1,1)  = 1.0
      AM(2,1)  = 0.0
      AM(2,2)  = 1.0
      M(1,1)   = 1.0
      M(1,2)   = 0.0
      M(2,1)   = 0.0
      M(2,2)   = 0.0
C                                               initial values
      XM(1)    = YDATA(1)
      XM(2)    = 0.22E-4 2.0E-3
      XW(1)    = YDATA(1)
      X(1)     = YDATA(1)
      PM(1,1)  = 2.5E-5
      PM(1,2)  = 1E-5
      PM(2,1)  = 1E-5
      PM(2,2)  = 4E-6
      PW(1,1)  = 2.5E-5
C
      WRITE(400,99998)
      WRITE(500,99998)
      WRITE(400,99999) 1, 0, ' E', XM(1), XM(2), PM(1,1),
     &                    PM(1,2), PM(2,2)
      WRITE(400,99999) 1, 0, ' S', XW(1), XM(2), PW(1,1)
C
      DO 10 I=2, NOBS
C
         Y(1)     = YDATA(I)
         TAUB     = (4.831*V2DATA(I))*((HDATA(I)-1)**(-0.30))
         IF (TAUB.GE.TAUE) THEN
C                                               update
            XM(1)    = X(1)
            CALL KALM1(PM,R,XM,Y,M,KNEW,XNEW,ENEW,PNEW)
            XM(1)    = XNEW(1)
            XM(2)    = XNEW(2)
            PM(1,1)  = PNEW(1,1)
            PM(1,2)  = PNEW(1,2)
            PM(2,1)  = PNEW(2,1)
            PM(2,2)  = PNEW(2,2)
            WRITE(400,99999) I-1, I-1, ' E', XM(1), XM(2), XW,
     &                       PM(1,1), PM(1,2), PM(2,2),
     &                       KNEW(1), KNEW(2)
C                                               prediction
            AM(1,2) = (TAUB)/(TAUE) - 1
            CALL KALM2(XM,AM,PM,QM,XNEW,HNEW,GNEW,PNEW)
            XM(1)    = XNEW(1)
            XM(2)    = XNEW(2)
            PM(1,1)  = PNEW(1,1)
            PM(1,2)  = PNEW(1,2)
            PM(2,1)  = PNEW(2,1)
            PM(2,2)  = PNEW(2,2)
            WRITE(500,99999) I, I-1, ' E', XM(1), XM(2), PM(1,1),
     &                       PM(1,2), PM(2,2)
            X(1)     = XM(1)
         ELSE
            IF (TAUB.LE.TAUD) THEN
C                                               update
               CALL KALM11(PW,R,X,Y,KNEW,XNEW,PNEW)
               XW(1)    = XNEW(1)
               PW(1,1) = PNEW(1,1)
               WRITE(400,99999) I-1, I-1, ' S', XW(1), XM(2) PW(1,1), KNEW(1)
C                                               prediction
               VERM     = W*((TAUB)/(TAUD)-1)
               AW(1,1) = 1 + VERM
               CALL KALM12(AW,XW,PW,QW,XNEW,PNEW)
               XW(1)    = XNEW(1)
               PW(1,1) = PNEW(1,1)
               WRITE (500,99999) I, I-1,  ' S', XW(1), XM(2) PW(1,1)
               X(1)     = XW(1)
            ELSE
               WRITE(400,99999) I-1, I-1, ' N', X(1)
               WRITE(500,99999) I, I-1, ' N', X(1)
            ENDIF
         ENDIF
   10 CONTINUE
C
99998 FORMAT ('  k/j  ', 'ES',' C(k/j)', '  M(k/j)',
     &        'P(k/j)11', 'P(k/j)12', 'P(k/j)22',
     &        ' K1(k) ', ' K2(k)')
99999 FORMAT (I4, '/', I3, A2, 7F8.6)
      END
C
C
C
C
C
C
C ******************************************************************
C
      SUBROUTINE KALM1(PIN,RIN,XIN,YIN,MIN,KUIT,XUIT,EUIT,PUIT)
C                                        calculates X(k+1),
C                                        K(k+1) and P(k+1)
      REAL PIN(2,2), RIN(1,1), QIN(1,1), XIN(2), YIN(1), MIN(2,2)
      REAL KUIT(2), XUIT(2), EUIT(2,2), PUIT(2,2)
C
      KUIT(1)    = PIN(1,1)/(PIN(1,1)+RIN(1,1))
      KUIT(2)    = PIN(2,1)/(PIN(1,1)+RIN(1,1))
      XUIT(1)    = XIN(1) + KUIT(1)*(YIN(1)-XIN(1))
      XUIT(2)    = XIN(2) + KUIT(2)*(YIN(1)-XIN(1))
      EUIT(1,1) = VMEE(PIN,MIN)
      EUIT(1,2) = VMET(PIN,MIN)
      EUIT(2,1) = VMTE(PIN,MIN)
      EUIT(2,2) = VMTT(PIN,MIN)
      PUIT(1,1) = PIN(1,1) - VMEE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(1,2) = PIN(1,2) - VMET(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,1) = PIN(2,1) - VMTE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,2) = PIN(2,2) - VMTT(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      RETURN
      END
C
      SUBROUTINE KALM2(XIN,AIN,PIN,QIN,XUIT,HUIT,GUIT,PUIT)
C                                        calculates X(k+1/k) and
C                                        P(k+1/k)
      REAL XIN(2), AIN(2,2), PIN(2,2), QIN(1,1), XUIT(2), HUIT(2,2)
      REAL GUIT(2,2), PUIT(2,2)
C
      XUIT(1)    = VMVECE(AIN,XIN)
      XUIT(2)    = VMVECT(AIN,XIN)
      HUIT(1,1) = AIN(1,1)
      HUIT(1,2) = AIN(2,1)
      HUIT(2,1) = AIN(1,2)
      HUIT(2,2) = AIN(2,2)
      GUIT(1,1) = VMEE(PIN,HUIT)
      GUIT(1,2) = VMET(PIN,HUIT)
      GUIT(2,1) = VMTE(PIN,HUIT)
      GUIT(2,2) = VMTT(PIN,HUIT)
      PUIT(1,1) = VMEE(AIN,GUIT) + QIN(1,1)
      PUIT(1,2) = VMET(AIN,GUIT)
      PUIT(2,1) = VMTE(AIN,GUIT)
      PUIT(2,2) = VMTT(AIN,GUIT)
      RETURN
      END
C
      SUBROUTINE KALM11(PIN,RIN,XIN,YIN,KUIT,XUIT,PUIT)
C                                        calculates X(k+1),
C                                        K(k+1) and P(k+1)
      REAL PIN(1,1), RIN(1,1), XIN(1), YIN(1), KUIT(1), XUIT(1)
      REAL PUIT(1,1)
C
      KUIT(1)    = PIN(1,1)/(PIN(1,1)+RIN(1,1))
      XUIT(1)    = XIN(1) + KUIT(1)*(YIN(1)-XIN(1))
      PUIT(1,1) = PIN(1,1) - (PIN(1,1)*PIN(1,1))/(PIN(1,1)+RIN(1,1))
      RETURN
      END
C
      SUBROUTINE KALM12(AIN,XIN,PIN,QIN,XUIT,PUIT)
C                                        calculates X(k+1/k) and
C                                        P(k+1/k)
      REAL XIN(1), AIN(1,1), PIN(1,1), QIN(1,1), XUIT(1), PUIT(1,1)
C
      XUIT(1)    = AIN(1,1)*XIN(1)
      PUIT(1,1) = AIN(1,1)*PIN(1,1)*AIN(1,1) + QIN(1,1)
      RETURN
      END
C
C ******************************************************************
C                                        multiply matrices
      FUNCTION VMEE(B,D)
      REAL B(2,2), D(2,2)
      VMEE = B(1,1)*D(1,1) + B(1,2)*D(2,1)
      RETURN
      END
C
      REAL FUNCTION VMET(B,D)
      REAL B(2,2), D(2,2)
      VMET = B(1,1)*D(1,2) + B(1,2)*D(2,2)
      RETURN
      END
C
      REAL FUNCTION VMTE(B,D)
      REAL B(2,2), D(2,2)
      VMTE = B(2,1)*D(1,1) + B(2,2)*D(2,1)
      RETURN
      END
C
      REAL FUNCTION VMTT(B,D)
      REAL B(2,2), D(2,2)
      VMTT = B(2,1)*D(1,2) + B(2,2)*D(2,2)
      RETURN
      END
C
C                                        mat*vector
      REAL FUNCTION VMVECE(B,Z)
      REAL B(2,2), Z(2)
      VMVECE = B(1,1)*Z(1) + B(1,2)*Z(2)
      RETURN
      END
C
      REAL FUNCTION VMVECT(B,Z)
      REAL B(2,2), Z(2)
      VMVECT = B(2,1)*Z(1) + B(2,2)*Z(2)
      RETURN
      END
```

```
      PROGRAM KFTAUE
      PARAMETER (NOBS=343)
C
      INTEGER I
      REAL R(1,1), QM(1,1), QW(1,1), AM(2,2), AW(1,1), XM(2), XW(1)
      REAL X(2), PM(2,2), PW(1,1), Y(1), K(2), MAT(2,2)
      REAL YDATA(NOBS), HDATA(NOBS), V2DATA(NOBS)
      REAL XNEW(2), KNEW(2), ENEW(2,2), PNEW(2,2), GNEW(2,2)
      REAL HNEW(2,2), TAUB, TAUD, VERM, W, M
C
      OPEN(UNIT=100,FILE='CONC',STATUS='OLD')
      OPEN(UNIT=200,FILE='HOOGTE',STATUS='OLD')
      OPEN(UNIT=300,FILE='VOPP2',STATUS='OLD')
      OPEN(UNIT=400,FILE='UITTE7',STATUS='NEW')
      OPEN(UNIT=500,FILE='OOKTE7',STATUS='NEW')
      READ(100,*,END=tt)YDATA
      READ(200,*,END=tt)HDATA
      READ(300,*,END=tt)V2DATA
C                                               values
      M        = 0.1E-3
      TAUD     = 0.06
      W        = 0.3E-3
      R(1,1)   = 2.5E-5
      QM(1,1) = 5E-4
      QW(1,1) = 5E-4
      AM(1,1) = 1.0
      AM(2,1) = 0.0
      AM(2,2) = 1.0
      MAT(1,1)  = 1.0
      MAT(1,2)  = 0.0
      MAT(2,1)  = 0.0
      MAT(2,2)  = 0.0
C                                               initial values
      XM(1)    = YDATA(1)
      XM(2)    = 0.07
      XW(1)    = YDATA(1)
      X(1)     = YDATA(1)
      PM(1,1) = 2.5E-5
      PM(1,2) = 3E-4
      PM(2,1) = 3E-4
      PM(2,2) = 3.6E-3
      PW(1,1) = 2.5E-5
C
      WRITE(400,99998)
      WRITE(500,99998)
      WRITE(400,99999) 1, 0, ' E', XM(1), XM(2), PM(1,1),
     &                    PM(1,2), PM(2,2)
      WRITE(400,99999) 1, 0, ' S', XW(1), XM(2), PW(1,1)
C
      DO 10 I=2, NOBS
C
         Y(1)    = YDATA(I)
         TAUB    = (4.831*V2DATA(I))*((HDATA(I)-1)**(-0.30))
         IF (TAUB.GE.XM(2)) THEN
C                                               update
            XM(1)     = X(1)
            CALL KALM1(PM,R,XM,Y,MAT,KNEW,XNEW,ENEW,PNEW)
            XM(1)     = XNEW(1)
            XM(2)     = XNEW(2)
            PM(1,1) = PNEW(1,1)
            PM(1,2) = PNEW(1,2)
            PM(2,1) = PNEW(2,1)
            PM(2,2) = PNEW(2,2)
            WRITE(400,99999) I-1, I-1, ' E', XM(1), XM(2),
     &                       PM(1,1), PM(1,2), PM(2,2),
     &                       KNEW(1), KNEW(2)
C                                               prediction
            AM(1,2) = (-1)*M*TAUB*((XM(2))**(-2))
            CALL KALM4(XM,M,TAUB,AM,PM,QM,XNEW,HNEW,GNEW,PNEW)
            XM(1)     = XNEW(1)
            XM(2)     = XNEW(2)
            PM(1,1) = PNEW(1,1)
            PM(1,2) = PNEW(1,2)
            PM(2,1) = PNEW(2,1)
            PM(2,2) = PNEW(2,2)
            WRITE(500,99999) I, I-1, ' E', XM(1), XM(2), PM(1,1),
     &                       PM(1,2), PM(2,2)
            X(1)      = XM(1)
         ELSE
            IF (TAUB.LE.TAUD) THEN
C                                               update
               CALL KALM11(PW,R,X,Y,KNEW,XNEW,PNEW)
               XW(1)     = XNEW(1)
               PW(1,1) = PNEW(1,1)
               WRITE(400,99999) I-1, I-1, ' S', XW(1), XM(2), PW(1,1), KNEW(
C                                               prediction
               VERM      = W*((TAUB)/(TAUD)-1)
               AW(1,1) = 1 + VERM
               CALL KALM12(AW,XW,PW,QW,XNEW,PNEW)
               XW(1)     = XNEW(1)
               PW(1,1) = PNEW(1,1)
               WRITE (500,99999) i, I-1,  ' S', XW(1), XM(2), PW(1,1)
               X(1)      = XW(1)
            ELSE
               WRITE(400,99999) I-1, I-1, ' N', X(1)
               WRITE(500,99999) I, I-1, ' N', X(1)
            ENDIF
         ENDIF
   10 CONTINUE
C
99998 FORMAT ('  k/j  ', 'ES','  C(k/j)', '   M(k/j)',
     &        'P(k/j)11', 'P(k/j)12', 'P(k/j)22',
     &        ' K1(k) ', ' K2(k)')
99999 FORMAT (I4, '/', I3, A2, 7F8.6)
      END
C
C
C
C
C
C
C *****************************************************************
C
      SUBROUTINE KALM1(PIN,RIN,XIN,YIN,MIN,KUIT,XUIT,EUIT,PUIT)
C                                               calculates X(k+1),
C                                               K(k+1) and P(k+1)
      REAL PIN(2,2), RIN(1,1), QIN(1,1), XIN(2), YIN(1), MIN(2,2)
      REAL KUIT(2), XUIT(2), EUIT(2,2), PUIT(2,2)
C
      KUIT(1)    = PIN(1,1)/(PIN(1,1)+RIN(1,1))
      KUIT(2)    = PIN(2,1)/(PIN(1,1)+RIN(1,1))
      XUIT(1)    = XIN(1) + KUIT(1)*(YIN(1)-XIN(1))
      XUIT(2)    = XIN(2) + KUIT(2)*(YIN(1)-XIN(1))
      EUIT(1,1) = VMEE(PIN,MIN)
      EUIT(1,2) = VMET(PIN,MIN)
      EUIT(2,1) = VMTE(PIN,MIN)
      EUIT(2,2) = VMTT(PIN,MIN)
      PUIT(1,1) = PIN(1,1) - VMEE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(1,2) = PIN(1,2) - VMET(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,1) = PIN(2,1) - VMTE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,2) = PIN(2,2) - VMTT(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      RETURN
      END
C
      SUBROUTINE KALM4(XIN,MIN,BTAU,AIN,PIN,QIN,XUIT,HUIT,GUIT,PUIT)
C                                               calculates X(k+1/k) and
C                                               P(k+1/k)
      REAL XIN(2), AIN(2,2), MIN, BTAU, PIN(2,2), QIN(1,1), XUIT(2)
      REAL HUIT(2,2), GUIT(2,2), PUIT(2,2)
C
      XUIT(1)    = VMVECE(AIN,XIN) + (2*MIN*BTAU)/(XIN(2)) - MIN
      XUIT(2)    = VMVECT(AIN,XIN)
      HUIT(1,1) = AIN(1,1)
      HUIT(1,2) = AIN(2,1)
      HUIT(2,1) = AIN(1,2)
      HUIT(2,2) = AIN(2,2)
      GUIT(1,1) = VMEE(PIN,HUIT)
      GUIT(1,2) = VMET(PIN,HUIT)
      GUIT(2,1) = VMTE(PIN,HUIT)
      GUIT(2,2) = VMTT(PIN,HUIT)
      PUIT(1,1) = VMEE(AIN,GUIT)  + QIN(1,1)
      PUIT(1,2) = VMET(AIN,GUIT)
      PUIT(2,1) = VMTE(AIN,GUIT)
      PUIT(2,2) = VMTT(AIN,GUIT)
      RETURN
      END
C
      SUBROUTINE KALM11(PIN,RIN,XIN,YIN,KUIT,XUIT,PUIT)
C                                               calculates X(k+1),
C                                               K(k+1) and P(k+1)
      REAL PIN(1,1), RIN(1,1), XIN(1), YIN(1), KUIT(1), XUIT(1)
      REAL PUIT(1,1)
C
      KUIT(1)    = PIN(1,1)/(PIN(1,1)+RIN(1,1))
      XUIT(1)    = XIN(1) + KUIT(1)*(YIN(1)-XIN(1))
      PUIT(1,1) = PIN(1,1) - (PIN(1,1)*PIN(1,1))/(PIN(1,1)+RIN(1,1))
      RETURN
      END
C
      SUBROUTINE KALM12(AIN,XIN,PIN,QIN,XUIT,PUIT)
C                                               calculates X(k+1/k) and
C                                               P(k+1/k)
      REAL XIN(1), AIN(1,1), PIN(1,1), QIN(1,1), XUIT(1), PUIT(1,1)
C
      XUIT(1)    = AIN(1,1)*XIN(1)
      PUIT(1,1) = AIN(1,1)*PIN(1,1)*AIN(1,1) + QIN(1,1)
      RETURN
      END
C
C *****************************************************************
C                                               multiply matrices
      FUNCTION VMEE(B,D)
      REAL B(2,2), D(2,2)
      VMEE = B(1,1)*D(1,1) + B(1,2)*D(2,1)
      RETURN
      END
C
      REAL FUNCTION VMET(B,D)
      REAL B(2,2), D(2,2)
      VMET = B(1,1)*D(1,2) + B(1,2)*D(2,2)
      RETURN
      END
C
      REAL FUNCTION VMTE(B,D)
      REAL B(2,2), D(2,2)
      VMTE = B(2,1)*D(1,1) + B(2,2)*D(2,1)
      RETURN
      END
C
      REAL FUNCTION VMTT(B,D)
      REAL B(2,2), D(2,2)
      VMTT = B(2,1)*D(1,2) + B(2,2)*D(2,2)
      RETURN
      END
C
C                                               mat*vector
      REAL FUNCTION VMVECE(B,Z)
      REAL B(2,2), Z(2)
      VMVECE = B(1,1)*Z(1) + B(1,2)*Z(2)
      RETURN
      END
C
      REAL FUNCTION VMVECT(B,Z)
      REAL B(2,2), Z(2)
      VMVECT = B(2,1)*Z(1) + B(2,2)*Z(2)
      RETURN
      END
```

```
      PROGRAM KFTEM
      PARAMETER (NOBS=701)
C
      INTEGER I
      REAL R(1,1), QM(1,1), QW(1,1), AM(3,3), AW(1,1), XM(3), XW(1)
      REAL X(1), PM(3,3), PW(1,1), Y(1), K(3), MAT(3,3)
      REAL YDATA(NOBS), HDATA(NOBS), V2DATA(NOBS)
      REAL XNEW(3), KNEW(3), ENEW(3,3), PNEW(3,3), GNEW(3,3)
      REAL XNOV(1), KNOV(1), PNOV(1,1)
      REAL HNEW(3,3), TAUB, TAUD, W, TAUB, TERM, VERM
C
      OPEN(UNIT=100,FILE='CONC',STATUS='OLD')
      OPEN(UNIT=200,FILE='HOOGTE',STATUS='OLD')
      OPEN(UNIT=300,FILE='VOPP2',STATUS='OLD')
      OPEN(UNIT=400,FILE='TEM1',STATUS='NEW')
      OPEN(UNIT=500,FILE='REST1',STATUS='NEW')
      READ(100,*,END=tt)YDATA
      READ(200,*,END=tt)HDATA
      READ(300,*,END=tt)V2DATA
C                                              values
      W         = 0.3E-3
      TAUD      = 0.06
      R(1,1)    = 2.5E-5
      QM(1,1)   = 5E-4
      QW(1,1)   = 5E-4
      AM(1,1)   = 1.0
      AM(2,1)   = 0.0
      AM(2,2)   = 1.0
      AM(2,3)   = 0.0
      AM(3,1)   = 0.0
      AM(3,2)   = 0.0
      AM(3,3)   = 1.0
      MAT(1,1) = 1.0
      MAT(1,2) = 0.0
      MAT(1,3) = 0.0
      MAT(2,1) = 0.0
      MAT(2,2) = 0.0
      MAT(2,3) = 0.0
      MAT(3,1) = 0.0
      MAT(3,2) = 0.0
      MAT(3,3) = 0.0
C                                              initial values
      XM(1)     = YDATA(1)
      XM(2)     = 0.08
      XM(3)     = 0.1E-3
      XW(1)     = YDATA(1)
      X(1)      = YDATA(1)
      PM(1,1)   = 2.5E-5
      PM(1,2)   = 3E-4
      PM(1,3)   = 1E-5
      PM(2,1)   = 3E-4
      PM(2,2)   = 3.6E-3
      PM(2,3)   = 1.2E-4
      PM(3,1)   = 1E-5
      PM(3,2)   = 1.2E-4
      PM(3,3)   = 4E-6
      PW(1,1) = 2.5E-5
C
      WRITE(400,99998)
      WRITE(500,99998)
      WRITE(400,99999) 1, 0, ' E', XM(1), XM(2), XM(3), XM(3)/XM(2)
      WRITE(400,99999) 1, 0, ' S', XW(1), XM(2), XM(3), XM(3)/XM(2)
C
      DO 10 I=2, NOBS
C
         Y(1)    = YDATA(I)
         TAUB    = (4.831*V2DATA(I))*((HDATA(I)-1)**(-0.30))
         IF (TAUB.GE.XM(2)) THEN
C                                              update
            XM(1) = X(1)
            CALL KALT11(PM,R,XM,Y,MAT,KNEW,XNEW,ENEW,PNEW)
            XM(1)    = XNEW(1)
            XM(2)    = XNEW(2)
            XM(3)    = XNEW(3)
            PM(1,1) = PNEW(1,1)
            PM(1,2) = PNEW(1,2)
            PM(1,3) = PNEW(1,3)
            PM(2,1) = PNEW(2,1)
            PM(2,2) = PNEW(2,2)
            PM(2,3) = PNEW(2,3)
            PM(3,1) = PNEW(3,1)
            PM(3,2) = PNEW(3,2)
            PM(3,3) = PNEW(3,3)
            WRITE(400,99999) I-1, I-1, ' E', XM(1), XM(2), XM(3),
     &                       XM(3)/XM(2), KNEW(1), KNEW(2), KNEW(3)
C                                              prediction
            TERM     = (XM(3)*TAUB)/(XM(2))
            AM(1,2) = ((-1)*TERM)/(XM(2))
            AM(1,3) = (TAUB)/(XM(2)) - 1
            CALL KALT12(AM,XM,TERM,PM,QM,XNEW,HNEW,GNEW,PNEW)
            XM(1)    = XNEW(1)
            XM(2)    = XNEW(2)
            XM(3)    = XNEW(3)
            PM(1,1) = PNEW(1,1)
            PM(1,2) = PNEW(1,2)
            PM(1,3) = PNEW(1,3)
            PM(2,1) = PNEW(2,1)
            PM(2,2) = PNEW(2,2)
            PM(2,3) = PNEW(2,3)
            PM(3,1) = PNEW(3,1)
            PM(3,2) = PNEW(3,2)
            PM(3,3) = PNEW(3,3)
            WRITE(500,99999) I, I-1, ' E', XM(1), XM(2), XM(3),
     &                       XM(3)/XM(2)
            X(1)     = XM(1)
         ELSE
            IF (TAUB.LE.TAUD) THEN
C                                              update
               CALL KALT21(PW,R,XW,Y,KNOV,XNOV,PNOV)
               XW(1)    = XNOV(1)
               PW(1,1) = PNOV(1,1)
               WRITE(400,99999) I-1, I-1, ' S', XW(1), XM(2), XM(3),
     &                          XM(3)/XM(2), KNOV(1)
C                                              prediction
               VERM     = W*((TAUB)/(TAUD) - 1)
               AW(1,1) = 1 + VERM
               CALL KALT22(AW,XW,PW,QW,XNOV,PNOV)
               XW(1)    = XNOV(1)
               PW(1,1) = PNOV(1,1)
               WRITE(500,99999) I, I-1, ' S', XW(1), XM(2), XM(3),
     &                          XM(3)/XM(2)
               X(1)     = XW(1)
            ELSE
               WRITE(400,99999) I-1, I-1, ' N', X(1), XM(2), XM(3),
     &                          XM(3)/XM(2)
               WRITE(500,99999) I, I-1, ' N', X(1), XM(2), XM(3),
     &                          XM(3)/XM(2)
            ENDIF
         ENDIF
   10 CONTINUE
C
99998 FORMAT ('  k/j  ', 'ES',' C(k/j)', ' Tce(k/j)', '  M(k/j) '
     &          ' M/Tce ', ' K1(k)', ' K2(k)', ' K3(k)')
99999 FORMAT (I4, '/', I3, A2, 7F8.6)
      END
C
C ****************************************************************
C
      SUBROUTINE KALT11(PIN,RIN,XIN,YIN,MIN,KUIT,XUIT,EUIT,PUIT)
C                                              calculates X(k+1),
C                                              K(k+1) and P(k+1)
      REAL PIN(3,3), RIN(1,1), XIN(3), YIN(1), MIN(3,3)
      REAL KUIT(3), XUIT(3), EUIT(3,3), PUIT(3,3)
C
      KUIT(1)     = PIN(1,1)/(PIN(1,1)+RIN(1,1))
      KUIT(2)     = PIN(2,1)/(PIN(1,1)+RIN(1,1))
      KUIT(3)     = PIN(3,1)/(PIN(1,1)+RIN(1,1))
      XUIT(1)     = XIN(1) + KUIT(1)*(YIN(1)-XIN(1))
      XUIT(2)     = XIN(2) + KUIT(2)*(YIN(1)-XIN(1))
      XUIT(3)     = XIN(3) + KUIT(3)*(YIN(1)-XIN(1))
      EUIT(1,1) = V3MEE(PIN,MIN)
      EUIT(1,2) = V3MET(PIN,MIN)
      EUIT(1,3) = V3MED(PIN,MIN)
      EUIT(2,1) = V3MTE(PIN,MIN)
      EUIT(2,2) = V3MTT(PIN,MIN)
      EUIT(2,3) = V3MTD(PIN,MIN)
      EUIT(3,1) = V3MDE(PIN,MIN)
      EUIT(3,2) = V3MDT(PIN,MIN)
      EUIT(3,3) = V3MDD(PIN,MIN)
      PUIT(1,1) = PIN(1,1) - V3MEE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(1,2) = PIN(1,2) - V3MET(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(1,3) = PIN(1,3) - V3MED(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,1) = PIN(2,1) - V3MTE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,2) = PIN(2,2) - V3MTT(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,3) = PIN(2,3) - V3MTD(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(3,1) = PIN(3,1) - V3MDE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(3,2) = PIN(3,2) - V3MDT(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(3,3) = PIN(3,3) - V3MDD(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      RETURN
      END
C
      SUBROUTINE KALT12(AIN,XIN,TIN,PIN,QIN,XUIT,HUIT,GUIT,PUIT)
C                                              calculates X(k+1/k) and
C                                              P(k+1/k)
      REAL AIN(3,3), XIN(3), TIN, PIN(3,3), QIN(1,1)
      REAL XUIT(3), HUIT(3,3), GUIT(3,3), PUIT(3,3)
C
      XUIT(1)     = VMV3E(AIN,XIN) + TIN
      XUIT(2)     = VMV3T(AIN,XIN)
      XUIT(3)     = VMV3D(AIN,XIN)
      HUIT(1,1) = AIN(1,1)
      HUIT(1,2) = AIN(2,1)
      HUIT(1,3) = AIN(3,1)
      HUIT(2,1) = AIN(1,2)
      HUIT(2,2) = AIN(2,2)
      HUIT(2,3) = AIN(3,2)
      HUIT(3,1) = AIN(1,3)
      HUIT(3,2) = AIN(3,2)
      HUIT(3,3) = AIN(3,3)
      GUIT(1,1) = V3MEE(PIN,HUIT)
      GUIT(1,2) = V3MET(PIN,HUIT)
      GUIT(1,3) = V3MED(PIN,HUIT)
      GUIT(2,1) = V3MTE(PIN,HUIT)
      GUIT(2,2) = V3MTT(PIN,HUIT)
      GUIT(2,3) = V3MTD(PIN,HUIT)
      GUIT(3,1) = V3MDE(PIN,HUIT)
      GUIT(3,2) = V3MDT(PIN,HUIT)
      GUIT(3,3) = V3MDD(PIN,HUIT)
      PUIT(1,1) = V3MEE(AIN,GUIT) + QIN(1,1)
      PUIT(1,2) = V3MET(AIN,GUIT)
      PUIT(1,3) = V3MED(AIN,GUIT)
      PUIT(2,1) = V3MTE(AIN,GUIT)
      PUIT(2,2) = V3MTT(AIN,GUIT)
      PUIT(2,3) = V3MTD(AIN,GUIT)
      PUIT(3,1) = V3MDE(AIN,GUIT)
      PUIT(3,2) = V3MDT(AIN,GUIT)
      PUIT(3,3) = V3MDD(AIN,GUIT)
      RETURN
      END
C
      SUBROUTINE KALT21(PIN,RIN,XIN,YIN,KUIT,XUIT,PUIT)
C                                              calculates X(k+1),
C                                              K(k+1) and P(k+1)
      REAL PIN(1,1), RIN(1,1), XIN(1), YIN(1)
      REAL KUIT(1), XUIT(1), PUIT(1,1)
C
      KUIT(1)     = PIN(1,1)/(PIN(1,1)+RIN(1,1))
      XUIT(1)     = XIN(1) + KUIT(1)*(YIN(1) - XIN(1))
      PUIT(1,1) = PIN(1,1) - (PIN(1,1)*PIN(1,1))/(PIN(1,1)+RIN(1,1)
      RETURN
      END
C
      SUBROUTINE KALT22(AIN,XIN,PIN,QIN,XUIT,PUIT)
C                                              calculates X(k+1/k) and
C                                              P(k+1/k)
      REAL AIN(1,1), XIN(1), PIN(1,1), QIN(1,1)
      REAL XUIT(1), PUIT(1,1)
C
      XUIT(1)     = AIN(1,1)*XIN(1)
      PUIT(1,1) = AIN(1,1)*PIN(1,1)*AIN(1,1) + QIN(1,1)
      RETURN
      END
C
C ****************************************************************
C                                              multiply matrices
      FUNCTION V3MEE(B,D)
      REAL B(3,3), D(3,3)
      V3MEE = B(1,1)*D(1,1) + B(1,2)*D(2,1) + B(1,3)*D(3,1)
      RETURN
      END
C
      REAL FUNCTION V3MET(B,D)
      REAL B(3,3), D(3,3)
      V3MET = B(1,1)*D(1,2) + B(1,2)*D(2,2) + B(1,3)*D(3,2)
      RETURN
      END
C
      REAL FUNCTION V3MED(B,D)
      REAL B(3,3), D(3,3)
      V3MED = B(1,1)*D(1,3) + B(1,2)*D(2,3) + B(1,3)*D(3,3)
      RETURN
      END
C
      REAL FUNCTION V3MTE(B,D)
      REAL B(3,3), D(3,3)
      V3MTE = B(2,1)*D(1,1) + B(2,2)*D(2,1) + B(2,3)*D(3,1)
```

```
      RETURN
      END
C
      REAL FUNCTION V3MTT(B,D)
      REAL B(3,3), D(3,3)
      V3MTT = B(2,1)*D(1,2) + B(2,2)*D(2,2) + B(2,3)*D(3,2)
      RETURN
      END
C
      REAL FUNCTION V3MTD(B,D)
      REAL B(3,3), D(3,3)
      V3MTD = B(2,1)*D(1,3) + B(2,2)*D(2,3) + B(2,3)*D(3,3)
      RETURN
      END
C
      REAL FUNCTION V3MDE(B,D)
      REAL B(3,3), D(3,3)
      V3MDE = B(3,1)*D(1,1) + B(3,2)*D(2,1) + B(3,3)*D(3,1)
      RETURN
      END
C
      REAL FUNCTION V3MDT(B,D)
      REAL B(3,3), D(3,3)
      V3MDT = B(3,1)*D(1,2) + B(3,2)*D(2,2) + B(3,3)*D(3,2)
      RETURN
      END
C
      REAL FUNCTION V3MDD(B,D)
      REAL B(3,3), D(3,3)
      V3MDD = B(3,1)*D(1,3) + B(3,2)*D(2,3) + B(3,3)*D(3,3)
      RETURN
      END
C                                              mat*vector
      REAL FUNCTION VMV3E(B,Z)
      REAL B(3,3), Z(3)
      VMV3E = B(1,1)*Z(1) + B(1,2)*Z(2) + B(1,3)*Z(3)
      RETURN
      END
C
      REAL FUNCTION VMV3T(B,Z)
      REAL B(3,3), Z(3)
      VMV3T = B(2,1)*Z(1) + B(2,2)*Z(2) + B(2,3)*Z(3)
      RETURN
      END
C
      REAL FUNCTION VMV3D(B,Z)
      REAL B(3,3), Z(3)
      VMV3D = B(3,1)*Z(1) + B(3,2)*Z(2) + B(3,3)*Z(3)
      RETURN
      END
```

```
      PROGRAM MODEL1
      PARAMETER (NOBS=343)
C
      INTEGER I
      REAL C(1), CNEW(1), TERM, VERM
      REAL CODAT, HDATA(NOBS), V2DATA(NOBS)
      REAL M, W, TAUD, TAUE
C
      OPEN(UNIT=100,FILE='C0',STATUS='OLD')
      OPEN(UNIT=200,FILE='HOOGTE',STATUS='OLD')
      OPEN(UNIT=300,FILE='VOPP2',STATUS='OLD')
      OPEN(UNIT=400,FILE='MOD1',STATUS='NEW')
      READ(100,*,END=tt)CODAT
      READ(200,*,END=tt)HDATA
      READ(300,*,END=tt)V2DATA
C                                              values
      M    = 0.2E-4
      W    = 0.3E-3
      TAUD = 0.058
      TAUE = 0.061
      C(1) = CODAT
C
      WRITE(400,99998)
      WRITE(400,99999) 0, '..', C(1)
C
      DO 10 I=2, NOBS
      TAUB = (4.831*V2DATA(I))*((HDATA(I)-1)**(-0.30))
      IF (TAUB.GE.TAUE) THEN
         TERM = M*((TAUB/TAUE)-1)
         CALL EROSIE(C,TERM,CNEW)
         C(1) = CNEW(1)
         WRITE(400,99999) I-1, ' E', C(1), TERM
      ELSE
         IF (TAUB.LE.TAUD) THEN
            VERM = W*((TAUB/TAUD)-1)
            CALL SEDTIE(C,VERM,CNEW)
            C(1) = CNEW(1)
            WRITE(400,99999) I-1, ' S', C(1), 0.0, VERM
         ELSE
            WRITE(400,99999) I-1, ' N', C(1)
         ENDIF
      ENDIF
   10 CONTINUE
99998 FORMAT (' k  ', 'ES', '  C(k)  ', ' TERM ', ' VERM ')
99999 FORMAT (I4, A2, 3F8.6)
      END
C
C ******************************************************************
C
      SUBROUTINE EROSIE(CIN,TIN,CUIT)
C
      REAL CIN(1), TIN, CUIT(1)
C
      CUIT(1) = CIN(1)+TIN
      RETURN
      END
C
      SUBROUTINE SEDTIE(CIN,VIN,CUIT)
C
      REAL CIN(1), VIN, CUIT(1)
C
      CUIT(1) = CIN(1)*(1+VIN)
      RETURN
      END
```

```
      PROGRAM KFHTEM
      PARAMETER (NOBS=701)
C
      INTEGER I
      REAL R(1,1), QM(1,1), QW(1,1), AM(3,3), AW(1,1), XM(3), XW(1)
      REAL X(1), PM(3,3), PW(1,1), Y(1), K(3), MAT(3,3)
      REAL YDATA(NOBS), HDATA(NOBS), V2DATA(NOBS)
      REAL XNEW(3), KNEW(3), ENEW(3,3), PNEW(3,3), GNEW(3,3)
      REAL XNOV(1), KNOV(1), PNOV(1,1)
      REAL HNEW(3,3), TAUB, TAUD, W, TERM, VERM
C
      OPEN(UNIT=100,FILE='CONC',STATUS='OLD')
      OPEN(UNIT=200,FILE='HOOGTE',STATUS='OLD')
      OPEN(UNIT=300,FILE='VOPP2',STATUS='OLD')
      OPEN(UNIT=400,FILE='HTEM1',STATUS='NEW')
      OPEN(UNIT=500,FILE='HREST1',STATUS='NEW')
      READ(100,*,END=tt)YDATA
      READ(200,*,END=tt)HDATA
      READ(300,*,END=tt)V2DATA
C                                              values
      W        = 0.3E-3
      TAUD     = 0.06
      R(1,1)   = 2.5E-5
      QM(1,1)  = 5E-4
      QW(1,1)  = 5E-4
      AM(1,1)  = 1.0
      AM(2,1)  = 0.0
      AM(2,2)  = 1.0
      AM(2,3)  = 0.0
      AM(3,1)  = 0.0
      AM(3,2)  = 0.0
      AM(3,3)  = 1.0
      MAT(1,1) = 1.0
      MAT(1,2) = 0.0
      MAT(1,3) = 0.0
      MAT(2,1) = 0.0
      MAT(2,2) = 0.0
      MAT(2,3) = 0.0
      MAT(3,1) = 0.0
      MAT(3,2) = 0.0
      MAT(3,3) = 0.0
C                                              initial values
      XM(1)    = YDATA(1)
      XM(2)    = 0.08
      XM(3)    = 0.1E-3
      XW(1)    = YDATA(1)
      X(1)     = YDATA(1)
      PM(1,1)  = 2.5E-5
      PM(1,2)  = 3E-4
      PM(1,3)  = 1E-5
      PM(2,1)  = 3E-4
      PM(2,2)  = 3.6E-3
      PM(2,3)  = 1.2E-4
      PM(3,1)  = 1E-5
      PM(3,2)  = 1.2E-4
      PM(3,3)  = 4E-6
      PW(1,1) = 2.5E-5
C
      WRITE(400,99998)
      WRITE(500,99998)
      WRITE(400,99999) 1, 0, ' E', XM(1), XM(2), XM(3), XM(3)/XM(2)
      WRITE(400,99999) 1, 0, ' S', XW(1), XM(2), XM(3), XM(3)/XM(2)
C
      DO 10 I=2, NOBS
C
         Y(1)   = YDATA(I)
         TAUB   = (4.831*V2DATA(I))*((HDATA(I)-1)**(-0.30))
         IF (TAUB.GE.XM(2)) THEN
C                                              update
            XM(1) = X(1)
            CALL KALT11(PM,R,XM,Y,MAT,KNEW,XNEW,ENEW,PNEW)
            XM(1)   = XNEW(1)
            XM(2)   = XNEW(2)
            XM(3)   = XNEW(3)
            PM(1,1) = PNEW(1,1)
            PM(1,2) = PNEW(1,2)
            PM(1,3) = PNEW(1,3)
            PM(2,1) = PNEW(2,1)
            PM(2,2) = PNEW(2,2)
            PM(2,3) = PNEW(2,3)
            PM(3,1) = PNEW(3,1)
            PM(3,2) = PNEW(3,2)
            PM(3,3) = PNEW(3,3)
            WRITE(400,99999) I-1, I-1, ' E', XM(1), XM(2), XM(3),
     &                    XM(3)/XM(2), KNEW(1), KNEW(2), KNEW(3)
C                                              prediction
            TERM    = (XM(3)*TAUB)/(XM(2)*HDATA(I))
            AM(1,2) = ((-1)*TERM)/(XM(2))/(HDATA(I))
            AM(1,3) = (TAUB)/(XM(2)) - 1
            CALL KALT12(AM,XM,TERM,PM,QM,XNEW,HNEW,GNEW,PNEW)
            XM(1)   = XNEW(1)
            XM(2)   = XNEW(2)
            XM(3)   = XNEW(3)
            PM(1,1) = PNEW(1,1)
            PM(1,2) = PNEW(1,2)
            PM(1,3) = PNEW(1,3)
            PM(2,1) = PNEW(2,1)
            PM(2,2) = PNEW(2,2)
            PM(2,3) = PNEW(2,3)
            PM(3,1) = PNEW(3,1)
            PM(3,2) = PNEW(3,2)
            PM(3,3) = PNEW(3,3)
            WRITE(500,99999) I, I-1, ' E', XM(1), XM(2), XM(3),
     &                    XM(3)/XM(2)
            X(1)    = XM(1)
         ELSE
            IF (TAUB.LE.TAUD) THEN
C                                              update
               CALL KALT21(PW,R,XW,Y,KNOV,XNOV,PNOV)
               XW(1)   = XNOV(1)
               PW(1,1) = PNOV(1,1)
               WRITE(400,99999) I-1, I-1, ' S', XW(1), XM(2), XM(3),
     &                       XM(3)/XM(2), KNOV(1)
C                                              prediction
               VERM    = W*((TAUB)/(TAUD) - 1)
               AW(1,1) = 1 + VERM/HDATA(I)
               CALL KALT22(AW,XW,PW,QW,XNOV,PNOV)
               XW(1)   = XNOV(1)
               PW(1,1) = PNOV(1,1)
               WRITE(500,99999) I, I-1, ' S', XW(1), XM(2), XM(3),
     &                       XM(3)/XM(2)
               X(1)    = XW(1)
            ELSE
               WRITE(400,99999) I-1, I-1, ' N', X(1), XM(2), XM(3),
     &                       XM(3)/XM(2)
               WRITE(500,99999) I, I-1, ' N', X(1), XM(2), XM(3),
     &                       XM(3)/XM(2)
            ENDIF
```

```
         ENDIF
   10 CONTINUE
C
99998 FORMAT ('  k/j  ', 'ES',' C(k/j)', ' Tce(k/j)', '  M(k/j) ',
     &        ' M/Tce ', ' K1(k)', ' K2(k)', ' X3(k)')
99999 FORMAT (I4, '/', I3, A2, 7F8.6)
      END
C
C ******************************************************************
C
      SUBROUTINE KALT11(PIN,RIN,XIN,YIN,MIN,KUIT,XUIT,EUIT,PUIT)
C                                         calculates X(k+1),
C                                         K(k+1) and P(k+1)
      REAL PIN(3,3), RIN(1,1), XIN(3), YIN(1), MIN(3,3)
      REAL KUIT(3), XUIT(3), EUIT(3,3), PUIT(3,3)
C
      KUIT(1)    = PIN(1,1)/(PIN(1,1)+RIN(1,1))
      KUIT(2)    = PIN(2,1)/(PIN(1,1)+RIN(1,1))
      KUIT(3)    = PIN(3,1)/(PIN(1,1)+RIN(1,1))
      XUIT(1)    = XIN(1) + KUIT(1)*(YIN(1)-XIN(1))
      XUIT(2)    = XIN(2) + KUIT(2)*(YIN(1)-XIN(1))
      XUIT(3)    = XIN(3) + KUIT(3)*(YIN(1)-XIN(1))
      EUIT(1,1) = V3MEE(PIN,MIN)
      EUIT(1,2) = V3MET(PIN,MIN)
      EUIT(1,3) = V3MED(PIN,MIN)
      EUIT(2,1) = V3MTE(PIN,MIN)
      EUIT(2,2) = V3MTT(PIN,MIN)
      EUIT(2,3) = V3MTD(PIN,MIN)
      EUIT(3,1) = V3MDE(PIN,MIN)
      EUIT(3,2) = V3MDT(PIN,MIN)
      EUIT(3,3) = V3MDD(PIN,MIN)
      PUIT(1,1) = PIN(1,1) - V3MEE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(1,2) = PIN(1,2) - V3MET(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(1,3) = PIN(1,3) - V3MED(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,1) = PIN(2,1) - V3MTE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,2) = PIN(2,2) - V3MTT(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,3) = PIN(2,3) - V3MTD(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(3,1) = PIN(3,1) - V3MDE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(3,2) = PIN(3,2) - V3MDT(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(3,3) = PIN(3,3) - V3MDD(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      RETURN
      END
C
      SUBROUTINE KALT12(AIN,XIN,TIN,PIN,QIN,XUIT,HUIT,GUIT,PUIT)
C                                         calculates X(k+1/k) and
C                                         P(k+1/k)
      REAL AIN(3,3), XIN(3), TIN, PIN(3,3), QIN(1,1)
      REAL XUIT(3), HUIT(3,3), GUIT(3,3), PUIT(3,3)
C
      XUIT(1)    = VMV3E(AIN,XIN) + TIN
      XUIT(2)    = VMV3T(AIN,XIN)
      XUIT(3)    = VMV3D(AIN,XIN)
      HUIT(1,1) = AIN(1,1)
      HUIT(1,2) = AIN(2,1)
      HUIT(1,3) = AIN(3,1)
      HUIT(2,1) = AIN(1,2)
      HUIT(2,2) = AIN(2,2)
      HUIT(2,3) = AIN(3,2)
      HUIT(3,1) = AIN(1,3)
      HUIT(3,2) = AIN(3,2)
      HUIT(3,3) = AIN(3,3)
      GUIT(1,1) = V3MEE(PIN,HUIT)
      GUIT(1,2) = V3MET(PIN,HUIT)
      GUIT(1,3) = V3MED(PIN,HUIT)
      GUIT(2,1) = V3MTE(PIN,HUIT)
      GUIT(2,2) = V3MTT(PIN,HUIT)
      GUIT(2,3) = V3MTD(PIN,HUIT)
      GUIT(3,1) = V3MDE(PIN,HUIT)
      GUIT(3,2) = V3MDT(PIN,HUIT)
      GUIT(3,3) = V3MDD(PIN,HUIT)
      PUIT(1,1) = V3MEE(AIN,GUIT) + QIN(1,1)
      PUIT(1,2) = V3MET(AIN,GUIT)
      PUIT(1,3) = V3MED(AIN,GUIT)
      PUIT(2,1) = V3MTE(AIN,GUIT)
      PUIT(2,2) = V3MTT(AIN,GUIT)
      PUIT(2,3) = V3MTD(AIN,GUIT)
      PUIT(3,1) = V3MDE(AIN,GUIT)
      PUIT(3,2) = V3MDT(AIN,GUIT)
      PUIT(3,3) = V3MDD(AIN,GUIT)
      RETURN
      END
C
      SUBROUTINE KALT21(PIN,RIN,XIN,YIN,KUIT,XUIT,PUIT)
C                                         calculates X(k+1),
C                                         K(k+1) and P(k+1)
      REAL PIN(1,1), RIN(1,1), XIN(1), YIN(1)
      REAL KUIT(1), XUIT(1), PUIT(1,1)
C
      KUIT(1)    = PIN(1,1)/(PIN(1,1)+RIN(1,1))
      XUIT(1)    = XIN(1) + KUIT(1)*(YIN(1) - XIN(1))
      PUIT(1,1) = PIN(1,1) - (PIN(1,1)*PIN(1,1))/(PIN(1,1)+RIN(1,1))
      RETURN
      END
C
      SUBROUTINE KALT22(AIN,XIN,PIN,QIN,XUIT,PUIT)
C                                         calculates X(k+1/k) and
C                                         P(k+1/k)
      REAL AIN(1,1), XIN(1), PIN(1,1), QIN(1,1)
      REAL XUIT(1), PUIT(1,1)
C
      XUIT(1)    = AIN(1,1)*XIN(1)
      PUIT(1,1) = AIN(1,1)*PIN(1,1)*AIN(1,1) + QIN(1,1)
      RETURN
      END
C
C ******************************************************************
C                                         multiply matrices
      FUNCTION V3MEE(B,D)
      REAL B(3,3), D(3,3)
      V3MEE = B(1,1)*D(1,1) + B(1,2)*D(2,1) + B(1,3)*D(3,1)
      RETURN
      END
C
      REAL FUNCTION V3MET(B,D)
      REAL B(3,3), D(3,3)
      V3MET = B(1,1)*D(1,2) + B(1,2)*D(2,2) + B(1,3)*D(3,2)
      RETURN
      END
C
      REAL FUNCTION V3MED(B,D)
      REAL B(3,3), D(3,3)
      V3MED = B(1,1)*D(1,3) + B(1,2)*D(2,3) + B(1,3)*D(3,3)
      RETURN
      END
C
      REAL FUNCTION V3MTE(B,D)
      REAL B(3,3), D(3,3)
      V3MTE = B(2,1)*D(1,1) + B(2,2)*D(2,1) + B(2,3)*D(3,1)
      RETURN
      END
C
      REAL FUNCTION V3MTT(B,D)
      REAL B(3,3), D(3,3)
      V3MTT = B(2,1)*D(1,2) + B(2,2)*D(2,2) + B(2,3)*D(3,2)
      RETURN
      END
C
      REAL FUNCTION V3MTD(B,D)
      REAL B(3,3), D(3,3)
      V3MTD = B(2,1)*D(1,3) + B(2,2)*D(2,3) + B(2,3)*D(3,3)
      RETURN
      END
C
      REAL FUNCTION V3MDE(B,D)
      REAL B(3,3), D(3,3)
      V3MDE = B(3,1)*D(1,1) + B(3,2)*D(2,1) + B(3,3)*D(3,1)
      RETURN
      END
C
      REAL FUNCTION V3MDT(B,D)
      REAL B(3,3), D(3,3)
      V3MDT = B(3,1)*D(1,2) + B(3,2)*D(2,2) + B(3,3)*D(3,2)
      RETURN
      END
C
      REAL FUNCTION V3MDD(B,D)
      REAL B(3,3), D(3,3)
      V3MDD = B(3,1)*D(1,3) + B(3,2)*D(2,3) + B(3,3)*D(3,3)
      RETURN
      END
C                                         mat*vector
      REAL FUNCTION VMV3E(B,Z)
      REAL B(3,3), Z(3)
      VMV3E = B(1,1)*Z(1) + B(1,2)*Z(2) + B(1,3)*Z(3)
      RETURN
      END
C
      REAL FUNCTION VMV3T(B,Z)
      REAL B(3,3), Z(3)
      VMV3T = B(2,1)*Z(1) + B(2,2)*Z(2) + B(2,3)*Z(3)
      RETURN
      END
C
      REAL FUNCTION VMV3D(B,Z)
      REAL B(3,3), Z(3)
      VMV3D = B(3,1)*Z(1) + B(3,2)*Z(2) + B(3,3)*Z(3)
      RETURN
      END
```

```
      PROGRAM KFHMW
      PARAMETER (NOBS=701)
C
      INTEGER I
      REAL R(1,1), QM(1,1), QW(1,1), AM(2,2), AW(2,2), XM(2), XW(2)
      REAL X(2), PM(2,2), PW(2,2), Y(1), K(2), M(2,2)
      REAL YDATA(NOBS), HDATA(NOBS), V2DATA(NOBS)
      REAL XNEW(2), KNEW(2), ENEW(2,2), PNEW(2,2), GNEW(2,2)
      REAL HNEW(2,2), TAUB, TAUE, TAUD, TERM
C
      OPEN(UNIT=100,FILE='CONC',STATUS='OLD')
      OPEN(UNIT=200,FILE='HOOGTE',STATUS='OLD')
      OPEN(UNIT=300,FILE='VOPP2',STATUS='OLD')
      OPEN(UNIT=400,FILE='HMW1',STATUS='NEW')
      OPEN(UNIT=500,FILE='HEXT1',STATUS='NEW')
      READ(100,*,END=tt)YDATA
      READ(200,*,END=tt)HDATA
      READ(300,*,END=tt)V2DATA
C                                         values
      TAUE    = 0.08
      TAUD    = 0.06
      R(1,1)  = 2.5E-5
      QM(1,1) = 5E-4
      QW(1,1) = 5E-4
      AM(1,1) = 1.0
      AM(2,1) = 0.0
      AM(2,2) = 1.0
      AW(2,1) = 0.0
      AW(2,2) = 1.0
      M(1,1)  = 1.0
      M(1,2)  = 0.0
      M(2,1)  = 0.0
      M(2,2)  = 0.0
C                                         initial values
      XM(1)   = YDATA(1)
      XM(2)   = 0.1E-3
      XW(1)   = YDATA(1)
      XW(2)   = 0.3E-3
      X(1)    = YDATA(1)
      PM(1,1) = 2.5E-5
      PM(1,2) = 1E-5
      PM(2,1) = 1E-5
      PM(2,2) = 4E-6
      PW(1,1) = 2.5E-5
      PW(1,2) = 5E-7
      PW(2,1) = 5E-7
      PW(2,2) = 1E-8
C
      WRITE(400,99998)
      WRITE(500,99998)
      WRITE(400,99999) 1, 0, ' E', XM(1), XM(2), XW(2), PM(1,1),
     &                 PM(1,2), PM(2,2)
      WRITE(400,99999) 1, 0, ' S', XW(1), XM(2), XW(2), PW(1,1),
     &                 PW(1,2), PW(2,2)
C
      DO 10 I=2, NOBS
C
         Y(1)    = YDATA(I)
         TAUB    = (4.831*V2DATA(I))*((HDATA(I)-1)**(-0.30))
         IF (TAUB.GE.TAUE) THEN
C                                         update
            XM(1)   = X(1)
            CALL KALM1(PM,R,XM,Y,M,KNEW,XNEW,ENEW,PNEW)
            XM(1)   = XNEW(1)
            XM(2)   = XNEW(2)
            PM(1,1) = PNEW(1,1)
            PM(1,2) = PNEW(1,2)
            PM(2,1) = PNEW(2,1)
            PM(2,2) = PNEW(2,2)
            WRITE(400,99999) I-1, I-1, ' E', XM(1), XM(2), XW(2),
     &                       PM(1,1), PM(1,2), PM(2,2),
     &                       KNEW(1), KNEW(2)
C                                         prediction
            AM(1,2) = ((TAUB)/(TAUE) - 1)/(HDATA(I))
            CALL KALM2(XM,AM,PM,QM,XNEW,HNEW,GNEW,PNEW)
            XM(1)   = XNEW(1)
            XM(2)   = XNEW(2)
            PM(1,1) = PNEW(1,1)
            PM(1,2) = PNEW(1,2)
            PM(2,1) = PNEW(2,1)
            PM(2,2) = PNEW(2,2)
            WRITE(500,99999) I, I-1, ' E', XM(1), XM(2), XW(2), PM(1,1)
     &                       PM(1,2), PM(2,2)
            X(1)    = XM(1)
         ELSE
            IF (TAUB.LE.TAUD) THEN
C                                         update
               XW(1)   = X(1)
               CALL KALM1(PW,R,XW,Y,M,KNEW,XNEW,ENEW,PNEW)
               XW(1)   = XNEW(1)
               XW(2)   = XNEW(2)
               PW(1,1) = PNEW(1,1)
               PW(1,2) = PNEW(1,2)
               PW(2,1) = PNEW(2,1)
               PW(2,2) = PNEW(2,2)
               WRITE(400,99999) I-1, I-1, ' S', XW(1), XM(2), XW(2),
     &                          PW(1,1), PW(1,2), PW(2,2),
     &                          KNEW(1), KNEW(2)
C                                         prediction
               TERM    = ((TAUB)/(TAUD) - 1)/(HDATA(I))
               AW(1,1) = 1 + XW(2)*TERM
               AW(1,2) = XW(1)*TERM
               CALL KALM3(XW,TERM,AW,PW,QW,XNEW,HNEW,GNEW,PNEW)
               XW(1)   = XNEW(1)
               XW(2)   = XNEW(2)
               PW(1,1) = PNEW(1,1)
               PW(1,2) = PNEW(1,2)
               PW(2,1) = PNEW(2,1)
               PW(2,2) = PNEW(2,2)
               WRITE(500,99999) I, I-1, ' S', XW(1), XM(2), XW(2),
     &                          PW(1,1), PW(1,2), PW(2,2)
               X(1)    = XW(1)
            ELSE
               WRITE(400,99999) I-1, I-1, ' N', X(1), XM(2), XW(2)
               WRITE(500,99999) I, I-1, ' N', X(1), XM(2), XW(2)
            ENDIF
         ENDIF
   10 CONTINUE
C
99998 FORMAT ('  k/j  ', 'ES',' C(k/j)', '   M(k/j)', '  W(k/j) ',
     &        'P(k/j)11', 'P(k/j)12', 'P(k/j)22',
     &        ' K1(k) ', ' K2(k)')
99999 FORMAT (I4, '/', I3, A2, 8F8.6)
      END
C
C
C
C
C
C
C ***********************************************************************
C
      SUBROUTINE KALM1(PIN,RIN,XIN,YIN,MIN,KUIT,XUIT,EUIT,PUIT)
C                                         calculates X(k+1),
C                                         X(k+1) and P(k+1)
      REAL PIN(2,2), RIN(1,1), QIN(1,1), XIN(2), YIN(1), MIN(2,2)
      REAL KUIT(2), XUIT(2), EUIT(2,2), PUIT(2,2)
C
      KUIT(1)   = PIN(1,1)/(PIN(1,1)+RIN(1,1))
      KUIT(2)   = PIN(2,1)/(PIN(1,1)+RIN(1,1))
      XUIT(1)   = XIN(1) + KUIT(1)*(YIN(1)-XIN(1))
      XUIT(2)   = XIN(2) + KUIT(2)*(YIN(1)-XIN(1))
      EUIT(1,1) = VMEE(PIN,MIN)
      EUIT(1,2) = VMET(PIN,MIN)
      EUIT(2,1) = VMTE(PIN,MIN)
      EUIT(2,2) = VMTT(PIN,MIN)
      PUIT(1,1) = PIN(1,1) - VMEE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(1,2) = PIN(1,2) - VMET(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,1) = PIN(2,1) - VMTE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,2) = PIN(2,2) - VMTT(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      RETURN
      END
C
      SUBROUTINE KALM2(XIN,AIN,PIN,QIN,XUIT,HUIT,GUIT,PUIT)
C                                         calculates X(k+1/k) and
C                                         P(k+1/k)
      REAL XIN(2), AIN(2,2), PIN(2,2), QIN(1,1), XUIT(2), HUIT(2,2
      REAL GUIT(2,2), PUIT(2,2)
C
      XUIT(1)   = VMVECE(AIN,XIN)
      XUIT(2)   = VMVECT(AIN,XIN)
      HUIT(1,1) = AIN(1,1)
      HUIT(1,2) = AIN(2,1)
      HUIT(2,1) = AIN(1,2)
      HUIT(2,2) = AIN(2,2)
      GUIT(1,1) = VMEE(PIN,HUIT)
      GUIT(1,2) = VMET(PIN,HUIT)
      GUIT(2,1) = VMTE(PIN,HUIT)
      GUIT(2,2) = VMTT(PIN,HUIT)
      PUIT(1,1) = VMEE(AIN,GUIT) + QIN(1,1)
      PUIT(1,2) = VMET(AIN,GUIT)
      PUIT(2,1) = VMTE(AIN,GUIT)
      PUIT(2,2) = VMTT(AIN,GUIT)
      RETURN
      END
C
      SUBROUTINE KALM3(XIN,TIN,AIN,PIN,QIN,XUIT,HUIT,GUIT,PUIT)
C                                         calculates X(k+1/k) and
C                                         P(k+1/k)
      REAL XIN(2), AIN(2,2), TIN, PIN(2,2), QIN(1,1), XUIT(2)
      REAL HUIT(2,2), GUIT(2,2), PUIT(2,2)
C
      XUIT(1)   = VMVECE(AIN,XIN) - XIN(1)*XIN(2)*TIN
      XUIT(2)   = VMVECT(AIN,XIN)
      HUIT(1,1) = AIN(1,1)
      HUIT(1,2) = AIN(2,1)
      HUIT(2,1) = AIN(1,2)
      HUIT(2,2) = AIN(2,2)
      GUIT(1,1) = VMEE(PIN,HUIT)
      GUIT(1,2) = VMET(PIN,HUIT)
      GUIT(2,1) = VMTE(PIN,HUIT)
      GUIT(2,2) = VMTT(PIN,HUIT)
      PUIT(1,1) = VMEE(AIN,GUIT) + QIN(1,1)
      PUIT(1,2) = VMET(AIN,GUIT)
      PUIT(2,1) = VMTE(AIN,GUIT)
      PUIT(2,2) = VMTT(AIN,GUIT)
      RETURN
      END
C
C ***********************************************************************
C                                         multiply matrices
      FUNCTION VMEE(B,D)
      REAL B(2,2), D(2,2)
      VMEE = B(1,1)*D(1,1) + B(1,2)*D(2,1)
      RETURN
      END
C
      REAL FUNCTION VMET(B,D)
      REAL B(2,2), D(2,2)
      VMET = B(1,1)*D(1,2) + B(1,2)*D(2,2)
      RETURN
      END
C
      REAL FUNCTION VMTE(B,D)
      REAL B(2,2), D(2,2)
      VMTE = B(2,1)*D(1,1) + B(2,2)*D(2,1)
      RETURN
      END
C
      REAL FUNCTION VMTT(B,D)
      REAL B(2,2), D(2,2)
      VMTT = B(2,1)*D(1,2) + B(2,2)*D(2,2)
      RETURN
      END
C
C                                         mat*vector
      REAL FUNCTION VMVECE(B,Z)
      REAL B(2,2), Z(2)
      VMVECE = B(1,1)*Z(1) + B(1,2)*Z(2)
      RETURN
      END
C
      REAL FUNCTION VMVECT(B,Z)
      REAL B(2,2), Z(2)
      VMVECT = B(2,1)*Z(1) + B(2,2)*Z(2)
      RETURN
      END
```

[redacted]

```
      PROGRAM KFHMW'
      PARAMETER (NOBS=701)
C
      INTEGER I
      REAL R(1,1), QM(1,1), QW(1,1), AM(2,2), AW(2,2), XME(2), XMV(2)
      REAL XWE(2), XWV(2)
      REAL X(1), PM(2,2), PW(2,2), Y(1), K(2), M(2,2)
      REAL YDATA(NOBS), HDATA(NOBS), V2DATA(NOBS)
      REAL XNEW(2), KNEW(2), ENEW(2,2), PNEW(2,2), GNEW(2,2)
      REAL HNEW(2,2), TAUB, TEV, TEE, TDV, TDE, TERM, ALFA
C
      OPEN(UNIT=100,FILE='CONC',STATUS='OLD')
      OPEN(UNIT=200,FILE='HOOGTE',STATUS='OLD')
      OPEN(UNIT=300,FILE='VOPP2',STATUS='OLD')
      OPEN(UNIT=400,FILE='MWEV1',STATUS='NEW')
      OPEN(UNIT=500,FILE='EXTEV1',STATUS='NEW')
      READ(100,*,END=tt)YDATA
      READ(200,*,END=tt)HDATA
      READ(300,*,END=tt)V2DATA
C                                        values
      ALFA    = 724
      TEV     =
      TER     =
      TDV     =
      TDE     =
      R(1,1)  =
      QM(1,1) =
      QW(1,1) =
      AM(1,1) = 1.0
      AM(2,1) = 0.0
      AM(2,2) = 1.0
      AW(2,1) = 0.0
      AW(2,2) = 1.0
      M(1,1)  = 1.0
      M(1,2)  = 0.0
      M(2,1)  = 0.0
      M(2,2)  = 0.0
C                                        initial values
      XME(2)  =
      XMV(2)  =
      XWE(2)  =
      XWV(2)  =
      X(1)    = YDATA(15)
      PM(1,1) =
      PM(1,2) =
      PM(2,1) =
      PM(2,2) =
      PW(1,1) =
      PW(1,2) =
      PW(2,1) =
      PW(2,2) =
C
      WRITE(400,99998)
      WRITE(500,99998)
      WRITE(400,99999) 1, 0, ' N', ' N', X(1), XMV(2), XME(2), XWV(2),
     &                 XWE(2)
C
      DO 10 I=16, NOBS
C
         Y(1)    = YDATA(I)
         TAUB    = (4.831*V2DATA(I-15))*((HDATA(I-15)-1)**(-0.30))
         IF (HDATA(I).GE.17.4) THEN

            IF (TAUB.GE.TEV) THEN
C                                        update
               XMV(1)  = X(1)
               CALL KALM1(PM,R,XMV,Y,M,KNEW,XNEW,ENEW,PNEW)
               XMV(1)  = XNEW(1)
               XMV(2)  = XNEW(2)
               PM(1,1) = PNEW(1,1)
               PM(1,2) = PNEW(1,2)
               PM(2,1) = PNEW(2,1)
               PM(2,2) = PNEW(2,2)
               WRITE(400,99999) I-1, I-1, ' E', ' V', XMV(1), XMV(2),
     &                          XME(2), XWV(2), XWE(2)
C                                        prediction
               AM(1,2) = ALFA*((TAUB/TEV) - 1)/HDATA(I)
               CALL KALM2(XMV,AM,PM,QM,XNEW,HNEW,GNEW,PNEW)
               XMV(1)  = XNEW(1)
               XMV(2)  = XNEW(2)
               PM(1,1) = PNEW(1,1)
               PM(1,2) = PNEW(1,2)
               PM(2,1) = PNEW(2,1)
               PM(2,2) = PNEW(2,2)
               WRITE(500,99999) I, I-1, ' E', ' V', XMV(1), XMV(2),
     &                          XME(2), XWV(2), XWE(2)
               X(1)    = XMV(1)
            ELSE
               IF (TAUB.LE.TDV) THEN
C                                        update
                  XWV(1)  = X(1)
                  CALL KALM1(PW,R,XWV,Y,M,KNEW,XNEW,ENEW,PNEW)
                  XWV(1)  = XNEW(1)
                  XWV(2)  = XNEW(2)
                  PW(1,1) = PNEW(1,1)
                  PW(1,2) = PNEW(1,2)
                  PW(2,1) = PNEW(2,1)
                  PW(2,2) = PNEW(2,2)
                  WRITE(400,99999) I-1, I-1, ' S', ' V', XWV(1),
     &                             XMV(2), XME(2), XWV(2), XWE(2)
C                                        prediction
                  TERM    = ALFA*((TAUB/TDV) - 1)/HDATA(I)
                  AW(1,1) = 1 + XWV(2)*TERM
                  AW(1,2) = XWV(1)*TERM
                  CALL KALM3(XWV,TERM,AW,PW,QW,XNEW,HNEW,GNEW,PNEW)
                  XWV(1)  = XNEW(1)
                  XWV(2)  = XNEW(2)
                  PW(1,1) = PNEW(1,1)
                  PW(1,2) = PNEW(1,2)
                  PW(2,1) = PNEW(2,1)
                  PW(2,2) = PNEW(2,2)
                  WRITE(500,99999) I, I-1, ' S', ' V', XWV(1), XMV(2),
     &                             XME(2), XWV(2), XWE(2)
                  X(1)    = XWV(1)
               ELSE
                  WRITE(400,99999) I-1, I-1, ' N', ' V', X(1), XMV(2),
     &                             XME(2), XWV(2), XWE(2)
                  WRITE(500,99999) I, I-1, ' N', ' V', X(1), XMV(2),
     &                             XME(2), XWV(2), XWE(2)
               ENDIF
            ENDIF
         ELSE
            IF (TAUB.GE.TEE) THEN
               XME(1)  = X(1)
               CALL KALM1(PM,R,XME,Y,M,KNEW,XNEW,ENEW,PNEW)
               XME(1)  = XNEW(1)
               XME(2)  = XNEW(2)
               PM(1,1) = PNEW(1,1)
               PM(1,2) = PNEW(1,2)
               PM(2,1) = PNEW(2,1)
               PM(2,2) = PNEW(2,2)
               WRITE(400,99999) I-1, I-1, ' E', ' E', XME(1), XMV(2)
     &                          XME(2), XWV(2), XWE(2)
C                                        prediction
               AM(1,2) = ALFA*((TAUB/TEE) -1)/HDATA(I)
               CALL KALM2(XME,AM,PM,QM,XNEW,HNEW,GNEW,PNEW)
               XME(1)  = XNEW(1)
               XME(2)  = XNEW(2)
               PM(1,1) = PNEW(1,1)
               PM(1,2) = PNEW(1,2)
               PM(2,1) = PNEW(2,1)
               PM(2,2) = PNEW(2,2)
               WRITE (500,99999) I, I-1, ' E', ' E', XME(1), XMV(2),
     &                           XME(2), XWV(2), XWE(2)
               X(1)    = XME(1)
            ELSE
               IF (TAUB.LE.TDE) THEN
                  XWE(1) = X(1)
                  CALL KALM1(PW,R,XWE,Y,M,KNEW,XNEW,ENEW,PNEW)
                  XWE(1)  = XNEW(1)
                  XWE(2)  = XNEW(2)
                  PW(1,1) = PNEW(1,1)
                  PW(1,2) = PNEW(1,2)
                  PW(2,1) = PNEW(2,1)
                  PW(2,2) = PNEW(2,2)
                  WRITE (400,99999) I-1, I-1, ' S', ' E', XWE(1),
     &                              XMV(2), XME(2), XWV(2), XWE(2)
C                                        prediction
                  TERM    = ((TAUB)/(TDE)-1)/(HDATA(I))
                  AW(1,1) = 1 + XWE(2)*TERM
                  AW(1,2) = XWE(1)*TERM
                  CALL KALM3(XWE,TERM,AW,PW,QW,XNEW,HNEW,GNEW,PNEW)
                  XWE(1)  = XNEW(1)
                  XWE(2)  = XNEW(2)
                  PW(1,1) = PNEW(1,1)
                  PW(1,2) = PNEW(1,2)
                  PW(2,1) = PNEW(2,1)
                  PW(2,2) = PNEW(2,2)
                  WRITE (500,99999) I, I-1, ' S', ' E', XWE(1), XMV(
     &                              XME(2), XWV(2), XWE(2)
                  X(1)    = XWE(1)
               ELSE
                  WRITE (400,99999) I-1, I-1, ' N', ' E', X(1), XMV(
     &                              XME(2), XWV(2), XWE(2)
                  WRITE (500,99999) I, I-1, ' N', ' E', X(1), XMV(2)
     &                              XME(2), XWV(2), XWE(2)
               ENDIF
            ENDIF
         ENDIF
   10 CONTINUE
C
99998 FORMAT ('  k/j  ', 'ES', 'EV',' C(kj)', '   MV(kj)', '  ME(kj
     &      '  WV(kj) ', '  WE(kj) ')
99999 FORMAT (I4, '/', I3, 2A2, 5F8.6)
      END
C
C
C
C
C
C
C ******************************************************************
C
      SUBROUTINE KALM1(PIN,RIN,XIN,YIN,MIN,KUIT,XUIT,EUIT,PUIT)
C                                        calculates X(k+1),
C                                        K(k+1) and P(k+1)
      REAL PIN(2,2), RIN(1,1), QIN(1,1), XIN(2), YIN(1), MIN(2,2)
      REAL KUIT(2), XUIT(2), EUIT(2,2), PUIT(2,2)
C
      KUIT(1)   = PIN(1,1)/(PIN(1,1)+RIN(1,1))
      KUIT(2)   = PIN(2,1)/(PIN(1,1)+RIN(1,1))
      XUIT(1)   = XIN(1) + KUIT(1)*(YIN(1)-XIN(1))
      XUIT(2)   = XIN(2) + KUIT(2)*(YIN(1)-XIN(1))
      EUIT(1,1) = VMEE(PIN,MIN)
      EUIT(1,2) = VMET(PIN,MIN)
      EUIT(2,1) = VMTE(PIN,MIN)
      EUIT(2,2) = VMTT(PIN,MIN)
      PUIT(1,1) = PIN(1,1) - VMEE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(1,2) = PIN(1,2) - VMET(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,1) = PIN(2,1) - VMTE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,2) = PIN(2,2) - VMTT(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      RETURN
      END
C
      SUBROUTINE KALM2(XIN,AIN,PIN,QIN,XUIT,HUIT,GUIT,PUIT)
C                                        calculates X(k+1/k) and
C                                        P(k+1/k)
      REAL XIN(2), AIN(2,2), PIN(2,2), QIN(1,1), XUIT(2), HUIT(2,2)
      REAL GUIT(2,2), PUIT(2,2)
C
      XUIT(1)   = VMVECE(AIN,XIN)
      XUIT(2)   = VMVECT(AIN,XIN)
      HUIT(1,1) = AIN(1,1)
      HUIT(1,2) = AIN(2,1)
      HUIT(2,1) = AIN(1,2)
      HUIT(2,2) = AIN(2,2)
      GUIT(1,1) = VMEE(PIN,HUIT)
      GUIT(1,2) = VMET(PIN,HUIT)
      GUIT(2,1) = VMTE(PIN,HUIT)
      GUIT(2,2) = VMTT(PIN,HUIT)
      PUIT(1,1) = VMEE(AIN,GUIT) + QIN(1,1)
      PUIT(1,2) = VMET(AIN,GUIT)
      PUIT(2,1) = VMTE(AIN,GUIT)
      PUIT(2,2) = VMTT(AIN,GUIT)
      RETURN
      END
C
      SUBROUTINE KALM3(XIN,TIN,AIN,PIN,QIN,XUIT,HUIT,GUIT,PUIT)
C                                        calculates X(k+1/k) and
C                                        P(k+1/k)
      REAL XIN(2), AIN(2,2), TIN, PIN(2,2), QIN(1,1), XUIT(2)
      REAL HUIT(2,2), GUIT(2,2), PUIT(2,2)
C
      XUIT(1)   = VMVECE(AIN,XIN) - XIN(1)*XIN(2)*TIN
      XUIT(2)   = VMVECT(AIN,XIN)
      HUIT(1,1) = AIN(1,1)
      HUIT(1,2) = AIN(2,1)
      HUIT(2,1) = AIN(1,2)
      HUIT(2,2) = AIN(2,2)
      GUIT(1,1) = VMEE(PIN,HUIT)
      GUIT(1,2) = VMET(PIN,HUIT)
      GUIT(2,1) = VMTE(PIN,HUIT)
      GUIT(2,2) = VMTT(PIN,HUIT)
      PUIT(1,1) = VMEE(AIN,GUIT) + QIN(1,1)
      PUIT(1,2) = VMET(AIN,GUIT)
      PUIT(2,1) = VMTE(AIN,GUIT)
```

```
      PUIT(2,2) = VMTT(AIN,GUIT)
      RETURN
      END
C
C ************************************************************************
C                                                    multiply matrices
      FUNCTION VNEE(B,D)
      REAL B(2,2), D(2,2)
      VNEE = B(1,1)*D(1,1) + B(1,2)*D(2,1)
      RETURN
      END
C
      REAL FUNCTION VMET(B,D)
      REAL B(2,2), D(2,2)
      VMET = B(1,1)*D(1,2) + B(1,2)*D(2,2)
      RETURN
      END
C
      REAL FUNCTION VMTE(B,D)
      REAL B(2,2), D(2,2)
      VMTE = B(2,1)*D(1,1) + B(2,2)*D(2,1)
      RETURN
      END
C
      REAL FUNCTION VMTT(B,D)
      REAL B(2,2), D(2,2)
      VMTT = B(2,1)*D(1,2) + B(2,2)*D(2,2)
      RETURN
      END
C
C                                                    mat*vector
      REAL FUNCTION VMVECE(B,Z)
      REAL B(2,2), Z(2)
      VMVECE = B(1,1)*Z(1) + B(1,2)*Z(2)
      RETURN
      END
C
      REAL FUNCTION VMVECT(B,Z)
      REAL B(2,2), Z(2)
      VMVECT = B(2,1)*Z(1) + B(2,2)*Z(2)
      RETURN
      END
```

```
[redacted] PROGRAM KFHTT
      PARAMETER (NOBS=701)
C
      INTEGER I
      REAL YDATA(NOBS), HDATA(NOBS), V2DATA(NOBS)
      REAL R(1,1), QM(1,1), QW(1,1), AM(2,2), AW(2,2), M(2,2), PM(2,2)
      REAL PW(2,2), ME, MV, WE, WV, TAUB, MVH, MEH, WVH, WEH
      REAL X(1), XMV(2), XME(2), XWV(2), XWE(2), Y(1)
      REAL XNEW(2), KNEW(2), ENEW(2,2), PNEW(2,2), GNEW(2,2), HNEW(2,2)
C
      OPEN(UNIT=100,FILE='CONC',STATUS='OLD')
      OPEN(UNIT=200,FILE='HOOGTE',STATUS='OLD')
      OPEN(UNIT=300,FILE='VOPP2',STATUS='OLD')
      OPEN(UNIT=400,FILE='HTT1',STATUS='NEW')
      OPEN(UNIT=500,FILE='HOOK1',STATUS='NEW')
      READ(100,*,END=tt)YDATA
      READ(200,*,END=tt)HDATA
      READ(300,*,END=tt)V2DATA
C                                                    values
      ME       = 2.0E-3
      MV       = 0.5E-3
      WE       = 3.0E-5
      WV       = 3.5E-5
      R(1,1)   = 2.5E-5
      QM(1,1)  = 5E-4
      QW(1,1)  = 5E-4
      AM(1,1)  = 1.0
      AM(2,1)  = 0.0
      AM(2,2)  = 1.0
      AW(2,1)  = 0.0
      AW(2,2)  = 1.0
      M(1,1)   = 1.0
      M(1,2)   = 0.0
      M(2,1)   = 0.0
      M(2,2)   = 0.0
C                                                    initial values
      X(1)     = YDATA(15)
      XME(2)   = 0.08
      XMV(2)   = 0.18
      XWE(2)   = 0.06
      XWV(2)   = 0.08
      PM(1,1)  = 2.5E-5
      PM(1,2)  = 3E-4
      PM(2,1)  = 3E-4
      PM(2,2)  = 3.6E-3
      PW(1,1)  = 2.5E-5
      PW(1,2)  = 1E-4
      PW(2,1)  = 1E-4
      PW(2,2)  = 4E-4
C
      WRITE(400,99998)
      WRITE(500,99998)
      WRITE(400,99999) 1, 0, ' N', ' N',X(1), XMV(2), XME(2), XWV(2),
     &                 XWE(2)
C
      DO 10 I=16, NOBS
C
         Y(1)     = YDATA(I)
         TAUB     = (4.831*V2DATA(I-15))*((HDATA(I-15)-1)**(-0.30))
         IF (HDATA(I).GE.17.4) THEN
            IF (TAUB.GE.XMV(2)) THEN
C                                                    update
               XMV(1)    = X(1)
               CALL KALM1(PM,R,XMV,Y,M,KNEW,XNEW,ENEW,PNEW)
               XMV(1)   = XNEW(1)
               XMV(2)   = XNEW(2)
               PM(1,1) = PNEW(1,1)
               PM(1,2) = PNEW(1,2)
               PM(2,1) = PNEW(2,1)
               PM(2,2) = PNEW(2,2)
               WRITE(400,99999) I-15, I-15, ' E', ' V',XMV(1), XMV(2),
     &                          XME(2), XWV(2), XWE(2)
C                                                    prediction
               MVH      = (MV)/(HDATA(I))
               AM(1,2) = (-1)*MVH*TAUB*((XMV(2))**(-2))
               CALL KALM4(XMV,MVH,TAUB,AM,PM,QM,XNEW,HNEW,GNEW,PNEW)
               XMV(1)   = XNEW(1)
               XMV(2)   = XNEW(2)
               PM(1,1) = PNEW(1,1)
               PM(1,2) = PNEW(1,2)
               PM(2,1) = PNEW(2,1)
               PM(2,2) = PNEW(2,2)
               WRITE(500,99999) I-14, I-15, ' E', ' V', XMV(1), XMV(2)
     &                          XME(2), XWV(2), XWE(2)
               X(1)     = XMV(1)
            ELSE
               IF (TAUB.LE.XWV(2)) THEN
C                                                    update
                  XWV(1)   = X(1)
                  CALL KALM1(PW,R,XWV,Y,M,KNEW,XNEW,ENEW,PNEW)
                  XWV(1)   = XNEW(1)
                  XWV(2)   = XNEW(2)
                  PW(1,1) = PNEW(1,1)
                  PW(1,2) = PNEW(1,2)
                  PW(2,1) = PNEW(2,1)
                  PW(2,2) = PNEW(2,2)
                  WRITE(400,99999) I-15, I-15, ' S', ' V', XWV(1),
     &                             XMV(2), XME(2), XWV(2), XWE(2)
C                                                    prediction
                  WVH       = (WV)/(HDATA(I))
                  AW(1,1) = 1 + WVH*((TAUB)/(XWV(2))-1)
                  AW(1,2) = (-1)*XWV(1)*WVH*TAUB*((XWV(2))**(-2))
                  CALL KALM5(XWV,WVH,TAUB,AW,PW,QW,XNEW,HNEW,GNEW,PNEW
                  XWV(1)   = XNEW(1)
                  XWV(2)   = XNEW(2)
                  PW(1,1) = PNEW(1,1)
                  PW(1,2) = PNEW(1,2)
                  PW(2,1) = PNEW(2,1)
                  PW(2,2) = PNEW(2,2)
                  WRITE(500,99999) I-14, I-15, ' S', ' V', XWV(1),
     &                             XMV(2), XME(2), XWV(2), XWE(2)
                  X(1)     = XWV(1)
               ELSE
                  WRITE(400,99999) I-15, I-15, ' N', ' V', X(1),
     &                             XMV(2), XME(2), XWV(2),XWE(2)
                  WRITE(500,99999) I-14, I-15, ' N', ' V', X(1),
     &                             XMV(2), XME(2), XWV(2), XWE(2)
               ENDIF
            ENDIF
         ELSE
            IF (TAUB.GE.XME(2)) THEN
C                                                    update
               XME(1)    = X(1)
               CALL KALM1(PM,R,XME,Y,M,KNEW,XNEW,ENEW,PNEW)
               XME(1)   = XNEW(1)
               XME(2)   = XNEW(2)
               PM(1,1) = PNEW(1,1)
               PM(1,2) = PNEW(1,2)
               PM(2,1) = PNEW(2,1)
               PM(2,2) = PNEW(2,2)
               WRITE(400,99999) I-15, I-15, ' E', ' E',XME(1), XMV(2),
     &                          XME(2), XWV(2), XWE(2)
C                                                    prediction
               MEH      = (ME)/(HDATA(I))
               AM(1,2) = (-1)*MEH*TAUB*((XME(2))**(-2))
               CALL KALM4(XME,MEH,TAUB,AM,PM,QM,XNEW,HNEW,GNEW,PNEW)
               XME(1)   = XNEW(1)
               XME(2)   = XNEW(2)
               PM(1,1) = PNEW(1,1)
               PM(1,2) = PNEW(1,2)
               PM(2,1) = PNEW(2,1)
               PM(2,2) = PNEW(2,2)
               WRITE(500,99999) I-14, I-15, ' E', ' E', XME(1), XMV(2)
     &                          XME(2), XWV(2), XWE(2)
               X(1)     = XME(1)
            ELSE
               IF (TAUB.LE.XWE(2)) THEN
C                                                    update
                  XWE(1)   = X(1)
                  CALL KALM1(PW,R,XWE,Y,M,KNEW,XNEW,ENEW,PNEW)
                  XWE(1)   = XNEW(1)
                  XWE(2)   = XNEW(2)
                  PW(1,1) = PNEW(1,1)
                  PW(1,2) = PNEW(1,2)
                  PW(2,1) = PNEW(2,1)
                  PW(2,2) = PNEW(2,2)
                  WRITE(400,99999) I-15, I-15, ' S', ' E', XWE(1),
     &                             XMV(2), XME(2), XWV(2), XWE(2)
C                                                    prediction
                  WEH       = (WE)/(HDATA(I))
                  AW(1,1) = 1 + WEH*((TAUB)/(XWE(2))-1)
                  AW(1,2) = (-1)*XWE(1)*WEH*TAUB*((XWE(2))**(-2))
                  CALL KALM5(XWE,WEH,TAUB,AW,PW,QW,XNEW,HNEW,GNEW,PNEW
                  XWE(1)   = XNEW(1)
                  XWE(2)   = XNEW(2)
                  PW(1,1) = PNEW(1,1)
                  PW(1,2) = PNEW(1,2)
                  PW(2,1) = PNEW(2,1)
                  PW(2,2) = PNEW(2,2)
                  WRITE(500,99999) I-14, I-15, ' S', ' E', XWE(1),
     &                             XMV(2), XME(2), XWV(2), XWE(2)
                  X(1)     = XWE(1)
               ELSE
                  WRITE(400,99999) I-15, I-15, ' N', ' E', X(1),
     &                             XMV(2), XME(2), XWV(2),XWE(2)
                  WRITE(500,99999) I-14, I-15, ' N', ' E', X(1),
     &                             XMV(2), XME(2), XWV(2), XWE(2)
               ENDIF
            ENDIF
         ENDIF
   10 CONTINUE
C
99998 FORMAT (' k/j  ', 'ES', 'EV', '  C(kj)', '  TEv(kj)',
     &        '  TEe(kj) ', '  TDv(kj)', '  TDe(kj)')
99999 FORMAT (I4, '/', I3, 2A2, 5F8.6)
      END
C
C
C
C
C
C
```

```
C ********************************************************************
C
      SUBROUTINE KALM1(PIN,RIN,XIN,YIN,MIN,KUIT,XUIT,EUIT,PUIT)
C                                            calculates X(k+1),
C                                            K(k+1) and P(k+1)
      REAL PIN(2,2), RIN(1,1), QIN(1,1), XIN(2), YIN(1), MIN(2,2)
      REAL KUIT(2), XUIT(2), EUIT(2,2), PUIT(2,2)
C
      KUIT(1)    = PIN(1,1)/(PIN(1,1)+RIN(1,1))
      KUIT(2)    = PIN(2,1)/(PIN(1,1)+RIN(1,1))
      XUIT(1)    = XIN(1) + KUIT(1)*(YIN(1)-XIN(1))
      XUIT(2)    = XIN(2) + KUIT(2)*(YIN(1)-XIN(1))
      EUIT(1,1) = VMEE(PIN,MIN)
      EUIT(1,2) = VMET(PIN,MIN)
      EUIT(2,1) = VMTE(PIN,MIN)
      EUIT(2,2) = VMTT(PIN,MIN)
      PUIT(1,1) = PIN(1,1)  - VMEE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(1,2) = PIN(1,2)  - VMET(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,1) = PIN(2,1)  - VMTE(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      PUIT(2,2) = PIN(2,2)  - VMTT(EUIT,PIN)/(PIN(1,1)+RIN(1,1))
      RETURN
      END
C
      SUBROUTINE KALM4(XIN,MHOED,BTAU,AIN,PIN,QIN,XUIT,HUIT,GUIT,PUIT)
C                                            calculates X(k+1/k) and
C                                            P(k+1/k)
      REAL XIN(2), AIN(2,2), MHOED, BTAU, PIN(2,2), QIN(1,1), XUIT(2)
      REAL HUIT(2,2), GUIT(2,2), PUIT(2,2)
C
      XUIT(1)    = VMVECE(AIN,XIN) + (2*MHOED*BTAU)/(XIN(2)) - MHOED
      XUIT(2)    = VMVECT(AIN,XIN)
      HUIT(1,1) = AIN(1,1)
      HUIT(1,2) = AIN(2,1)
      HUIT(2,1) = AIN(1,2)
      HUIT(2,2) = AIN(2,2)
      GUIT(1,1) = VMEE(PIN,HUIT)
      GUIT(1,2) = VMET(PIN,HUIT)
      GUIT(2,1) = VMTE(PIN,HUIT)
      GUIT(2,2) = VMTT(PIN,HUIT)
      PUIT(1,1) = VMEE(AIN,GUIT)  + QIN(1,1)
      PUIT(1,2) = VMET(AIN,GUIT)
      PUIT(2,1) = VMTE(AIN,GUIT)
      PUIT(2,2) = VMTT(AIN,GUIT)
      RETURN
      END
C
      SUBROUTINE KALM5(XIN,WHOED,BTAU,AIN,PIN,QIN,XUIT,HUIT,GUIT,PUIT)
C                                            calculates X(k+1/k) and
C                                            P(k+1/k)
      REAL XIN(2), AIN(2,2), WHOED, BTAU, PIN(2,2), QIN(1,1), XUIT(2)
      REAL HUIT(2,2), GUIT(2,2), PUIT(2,2)
C
      XUIT(1)    = VMVECE(AIN,XIN)  + (XIN(1)*WHOED*BTAU)/(XIN(2))
      XUIT(2)    = VMVECT(AIN,XIN)
      HUIT(1,1) = AIN(1,1)
      HUIT(1,2) = AIN(2,1)
      HUIT(2,1) = AIN(1,2)
      HUIT(2,2) = AIN(2,2)
      GUIT(1,1) = VMEE(PIN,HUIT)
      GUIT(1,2) = VMET(PIN,HUIT)
      GUIT(2,1) = VMTE(PIN,HUIT)
      GUIT(2,2) = VMTT(PIN,HUIT)
      PUIT(1,1) = VMEE(AIN,GUIT)  + QIN(1,1)
      PUIT(1,2) = VMET(AIN,GUIT)
      PUIT(2,1) = VMTE(AIN,GUIT)
      PUIT(2,2) = VMTT(AIN,GUIT)
      RETURN
      END
C
C ********************************************************************
C                                            multiply matrices
      FUNCTION VMEE(B,D)
      REAL B(2,2), D(2,2)
      VMEE = B(1,1)*D(1,1) + B(1,2)*D(2,1)
      RETURN
      END
C
      REAL FUNCTION VMET(B,D)
      REAL B(2,2), D(2,2)
      VMET = B(1,1)*D(1,2) + B(1,2)*D(2,2)
      RETURN
      END
C
      REAL FUNCTION VMTE(B,D)
      REAL B(2,2), D(2,2)
      VMTE = B(2,1)*D(1,1) + B(2,2)*D(2,1)
      RETURN
      END
C
      REAL FUNCTION VMTT(B,D)
      REAL B(2,2), D(2,2)
      VMTT = B(2,1)*D(1,2) + B(2,2)*D(2,2)
      RETURN
      END
C
C                                            mat*vector
      REAL FUNCTION VMVECE(B,Z)
      REAL B(2,2), Z(2)
      VMVECE = B(1,1)*Z(1) + B(1,2)*Z(2)
      RETURN
      END
C
      REAL FUNCTION VMVECT(B,Z)
      REAL B(2,2), Z(2)
      VMVECT = B(2,1)*Z(1) + B(2,2)*Z(2)
      RETURN
      END
```

```
%[rill]
%
% Reads input of 'qull'
%
%
format long;
ydat;
vtwee;
hoog;
tijd;
TEL= input('Enter the Tce-LW (TEL):     ')
TEH= input('Enter the Tce-HW (TEH):     ')
TDL= input('Enter the Tcd-LW (TDL):     ')
TDH= input('Enter the Tcd-HW (TDH):     ')
ML = input('Enter the initial M-LW (ML):     ')
MH = input('Enter the initial M-HW (MH):     ')
WL = input('Enter the initial W-LW (WL):     ')
WH = input('Enter the initial W-HW (WH):     ')
qu1= input('Enter q22 erosion LW (qu1):    ')
qu2= input('Enter q22 erosion HW (qu2):    ')
qu3= input('Enter q22 sedimentation LW (qu3):     ')
qu4= input('Enter q22 sedimentation HW (qu4):     ')
```

```
%[qull]
%
% Determines the process noise covariance matrix (to estimate M and W).
% Shows the likelihood functions of Q11, with known Q22
%
%
format long;
R    = 7.0e-8;
RUIS = 10e-11;
C    = [1 0];
ALFA = 724;
for j = 1:10
   RN(j)= 10*RUIS;
   RUIS = RN(j);
   Q1   = [RUIS 0.0
           0.0  qu1];
   Q2   = [RUIS 0.0
           0.0  qu2];
   Q3   = [RUIS 0.0
           0.0  qu3];
   Q4   = [RUIS 0.0
           0.0  qu4];
   X    = 0.061;
   X1   = [X
           ML];
   X2   = [X
           MH];
   X3   = [X
           WL];
   X4   = [X
           WH];
   P1   = [R     0.0
           0.0  1.0];
   P2   = P1;
   P3   = P1;
   P4   = P1;
   LIK1 = 0.0;
   LIK2 = 0.0;
   LIK3 = 0.0;
   LIK4 = 0.0;
   HOD1 = 0.0;
   HOD2 = 0.0;
   HOD3 = 0.0;
   HOD4 = 0.0;
   lik1 = 0.0;
   lik2 = 0.0;
   lik3 = 0.0;
   lik4 = 0.0;
   hod1 = 0.0;
   hod2 = 0.0;
   hod3 = 0.0;
   hod4 = 0.0;
   for i = 1:2147
     Tb = 4.831*V(i)/((H(i)-1)^(0.30));
%
%
%
     if H(i) >= 17.40
        if Tb <= TDH
%                                              deposition HW
           X4(1) = X;
           H4    = C*P4*C' + R;
           G4    = Y(i) - C*X4;
           lik4  = log(det(H4));
           hod4  = G4'*inv(H4)*G4;
           K     = P4*C'*inv(H4);
           XN    = X4 + K*(G4);
           PN    = P4 - P4*C'*inv(H4)*C*P4;
           X4    = XN;
           P4    = PN;
           TERM  = ALFA*((Tb/TDH)-1)/H(i);
           A     = [1+X4(2)*TERM  X4(1)*TERM
                    0.0              1.0        ];
           L     = [(-1)*TERM*X4(1)*X4(2)
                    0.0                   ];
           XN    = A*X4 + L;
           PN    = A*P4*A' + Q4;
           X4    = XN;
           P4    = PN;
           X     = X4(1);
        else
           if Tb >= TEH
%                                              erosion HW
              X2(1) = X;
              H2    = C*P2*C' + R;
              G2    = Y(i) - C*X2;
              lik2  = log(det(H2));
              hod2  = G2'*inv(H2)*G2;
              K     = P2*C'*inv(H2);
              XN    = X2 + K*(G2);
              PN    = P2 - P2*C'*inv(H2)*C*P2;
              X2    = XN;
              P2    = PN;
              A     = [1.0  ALFA*((Tb/TEH)-1)/H(i)
                       0.0  1.0                     ];
              XN    = A*X2;
              PN    = A*P2*A' + Q2;
              X2    = XN;
              P2    = PN;
              X     = X2(1);
           else
%                                              no erosion, no deposition HW
           end
        end
     else
%
%
        if Tb <= TDL
%                                              deposition LW
           X3(1) = X;
           H3    = C*P3*C' + R;
           G3    = Y(i) - C*X3;
           lik3  = log(det(H3));
           hod3  = G3'*inv(H3)*G3;
           K     = P3*C'*inv(H3);
           XN    = X3 + K*(G3);
           PN    = P3 - P3*C'*inv(H3)*C*P3;
           X3    = XN;
           P3    = PN;
           TERM  = ALFA*((Tb/TDH)-1)/H(i);
           A     = [1+X3(2)*TERM  X3(1)*TERM
                    0.0              1.0     ];
           L     = [(-1)*TERM*X3(1)*X3(2)
                    0.0                   ];
           XN    = A*X3 + L;
           PN    = A*P3*A' + Q3;
           X3    = XN;
           P3    = PN;
           X     = X3(1);
        else
           if Tb >= TEL
%                                              erosion LW
              X1(1) = X;
              H1    = C*P1*C' + R;
              G1    = Y(i) - C*X1;
              lik1  = log(det(H1));
              hod1  = G1'*inv(H1)*G1;
              K     = P1*C'*inv(H1);
              XN    = X1 + K*(G1);
              PN    = P1 - P1*C'*inv(H1)*C*P1;
              X1    = XN;
              P1    = PN;
              A     = [1.0  ALFA*((Tb/TEH)-1)/H(i)
                       0.0  1.0                     ];
              XN    = A*X1;
              PN    = A*P1*A' + Q1;
              X1    = XN;
              P1    = PN;
              X     = X1(1);
           else
%                                              no erosion, no deposition LW
           end
        end
     end
     LIK1 = lik1 + LIK1;
     LIK2 = lik2 + LIK2;
     LIK3 = lik3 + LIK3;
     LIK4 = lik4 + LIK4;
     HOD1 = hod1 + HOD1;
     HOD2 = hod2 + HOD2;
     HOD3 = hod3 + HOD3;
     HOD4 = hod4 + HOD4;
   end
%
%
%                                              bepalen van L-
   ML1(j) = 0.5*(LIK1 + HOD1);
   ML2(j) = 0.5*(LIK2 + HOD2);
   ML3(j) = 0.5*(LIK3 + HOD3);
   ML4(j) = 0.5*(LIK4 + HOD4);
end
semilogx(RN,ML1)
title('Likelihood Q11 Erosion LW')
pause
semilogx(RN,ML2)
title('Likelihood Q11 Erosion HW')
pause
semilogx(RN,ML3)
title('Likelihood Q11 Sedimentation LW')
pause
semilogx(RN,ML4)
title('Likelihood Q11 Sedimentation HW')
pause
```

```
%ri22
%
% Reads input of 'qu22'
%
%
format long;
ydat;
vtwee;
hoog;
tijd;
TEL= input('Enter the Tce-LW (TEL):     ')
TEH= input('Enter the Tce-HW (TEH):     ')
TDL= input('Enter the Tcd-LW (TDL):     ')
TDH= input('Enter the Tcd-HW (TDH):     ')
ML = input('Enter the initial M-LW (ML):     ')
MH = input('Enter the initial M-HW (MH):     ')
WL = input('Enter the initial W-LW (WL):     ')
WH = input('Enter the initial W-HW (WH):     ')
qu1= input('Enter q11 erosion LW (qu1):     ')
qu2= input('Enter q11 erosion HW (qu2):     ')
qu3= input('Enter q11 sedimentation LW (qu3):     ')
qu4= input('Enter q11 sedimentation HW (qu4):     ')
```

```
%qu22
%
% Determines the process noise covariance matrix (to estimate M and W).
% Shows the likelihood functions of Q22, with known Q11.
%
%
format long;
R    = 7.0e-8;
RUIS = 10e-11;
C    = [1 0];
ALFA = 724;
for j = 1:10
   RN(j)= 10*RUIS;
   RUIS = RN(j);
   Q1   = [qu1  0.0
           0.0  RUIS];
   Q2   = [qu2  0.0
           0.0  RUIS];
   Q3   = [qu3  0.0
           0.0  RUIS];
   Q4   = [qu4  0.0
           0.0  RUIS];
   X    = 0.061;
   X1   = [X
           ML];
   X2   = [X
           MH];
   X3   = [X
           WL];
   X4   = [X
           WH];
   P1   = [R    0.0
           0.0  1.0];
   P2   = P1;
   P3   = P1;
   P4   = P1;
   LIK1 = 0.0;
   LIK2 = 0.0;
   LIK3 = 0.0;
   LIK4 = 0.0;
   HOD1 = 0.0;
   HOD2 = 0.0;
   HOD3 = 0.0;
   HOD4 = 0.0;
   lik1 = 0.0;
   lik2 = 0.0;
   lik3 = 0.0;
   lik4 = 0.0;
   hod1 = 0.0;
   hod2 = 0.0;
   hod3 = 0.0;
   hod4 = 0.0;
   for i = 1:2147
     Tb = 4.831*V(i)/((H(i)-1)^(0.30));
%
%
%
      if H(i) >= 17.40
         if Tb <= TDH
%                                               deposition HW
            X4(1) = X;
            H4    = C*P4*C' + R;
            G4    = Y(i) - C*X4;
            lik4  = log(det(H4));
            hod4  = G4'*inv(H4)*G4;
            K     = P4*C'*inv(H4);
            XN    = X4 + K*(G4);
            PN    = P4 - P4*C'*inv(H4)*C*P4;
            X4    = XN;
            P4    = PN;
            TERM  = ALFA*((Tb/TDH)-1)/H(i);
            A     = [1+X4(2)*TERM  X4(1)*TERM
                     0.0             1.0        ];
            L     = [(-1)*TERM*X4(1)*X4(2)
                     0.0                   ];
            XN    = A*X4 + L;
            PN    = A*P4*A' + Q4;
            X4    = XN;
            P4    = PN;
            X     = X4(1);
         else
            if Tb >= TEH
%                                               erosion HW
               X2(1) = X;
               H2    = C*P2*C' + R;
               G2    = Y(i) - C*X2;
               lik2  = log(det(H2));
               hod2  = G2'*inv(H2)*G2;
               K     = P2*C'*inv(H2);
               XN    = X2 + K*(G2);
               PN    = P2 - P2*C'*inv(H2)*C*P2;
               X2    = XN;
               P2    = PN;
               A     = [1.0  ALFA*((Tb/TEH)-1)/H(i)
                        0.0  1.0                     ];
               XN    = A*X2;
               PN    = A*P2*A' + Q2;
               X2    = XN;
               P2    = PN;
               X     = X2(1);
            else
%                                               no erosion, no deposition HW
            end
         end
      else
%
%
         if Tb <= TDL
%                                               deposition LW
            X3(1) = X;
            H3    = C*P3*C' + R;
            G3    = Y(i) - C*X3;
            lik3  = log(det(H3));
            hod3  = G3'*inv(H3)*G3;
            K     = P3*C'*inv(H3);
            XN    = X3 + K*(G3);
            PN    = P3 - P3*C'*inv(H3)*C*P3;
            X3    = XN;
            P3    = PN;
            TERM  = ALFA*((Tb/TDH)-1)/H(i);
            A     = [1+X3(2)*TERM  X3(1)*TERM
                     0.0             1.0        ];
            L     = [(-1)*TERM*X3(1)*X3(2)
                     0.0                   ];
            XN    = A*X3 + L;
            PN    = A*P3*A' + Q3;
            X3    = XN;
            P3    = PN;
            X     = X3(1);
         else
            if Tb >= TEL
%                                               erosion LW
               X1(1) = X;
               H1    = C*P1*C' + R;
               G1    = Y(i) - C*X1;
               lik1  = log(det(H1));
               hod1  = G1'*inv(H1)*G1;
               K     = P1*C'*inv(H1);
               XN    = X1 + K*(G1);
               PN    = P1 - P1*C'*inv(H1)*C*P1;
               X1    = XN;
               P1    = PN;
               A     = [1.0  ALFA*((Tb/TEH)-1)/H(i)
                        0.0  1.0                     ];
               XN    = A*X1;
               PN    = A*P1*A' + Q1;
               X1    = XN;
               P1    = PN;
               X     = X1(1);
            else
%                                               no erosion, no deposition LW
             end
         end
      end
      LIK1 = lik1 + LIK1;
      LIK2 = lik2 + LIK2;
      LIK3 = lik3 + LIK3;
      LIK4 = lik4 + LIK4;
      HOD1 = hod1 + HOD1;
      HOD2 = hod2 + HOD2;
      HOD3 = hod3 + HOD3;
      HOD4 = hod4 + HOD4;
   end
%
%
%                                               bepalen van L-
   ML1(j) = 0.5*(LIK1 + HOD1);
   ML2(j) = 0.5*(LIK2 + HOD2);
   ML3(j) = 0.5*(LIK3 + HOD3);
   ML4(j) = 0.5*(LIK4 + HOD4);
end
semilogx(RN,ML1)
title('Likelihood Q22 Erosion LW')
pause
semilogx(RN,ML2)
title('Likelihood Q22 Erosion HW')
pause
semilogx(RN,ML3)
title('Likelihood Q22 Sedimentation LW')
pause
semilogx(RN,ML4)
title('Likelihood Q22 Sedimentation HW')
pause
```

```
%rkalf
%
% Reads input of 'kalfil'
%
%
format long;
ydat;
vtwee;
hoog;
tijd;
TEL= input('Enter the Tce-LW (TEL):   ')
TEH= input('Enter the Tce-HW (TEH):   ')
TDL= input('Enter the Tcd-LW (TDL):   ')
TDH= input('Enter the Tcd-HW (TDH):   ')
ML = input('Enter the initial M-LW (ML):   ')
MH = input('Enter the initial M-HW (MH):   ')
WL = input('Enter the initial W-LW (WL):   ')
WH = input('Enter the initial W-HW (WH):   ')
Q1= input('Enter Q erosion LW (Q1):   ')
Q2= input('Enter Q erosion HW (Q2):   ')
Q3= input('Enter Q sedimentation LW (Q3):   ')
Q4= input('Enter Q sedimentation HW (Q4):   ')
```

```
%kalfil
%
% Estimates M and W using the method of (extended) Kalman filtering.
% Q and Tau-critical can be read with 'rkalf'.
%
%                                                   initial values
format long;
R     = 7.0e-8;
C     = [1 0];
ALFA = 724;
X     = 0.061;
X1    = [X
         ML];
X2    = [X
         MH];
X3    = [X
         WL];
X4    = [X
         WH];
P1    = [R     0.0
         0.0  1.0];
P2    = P1;
P3    = P1;
P4    = P1;
for i = 1:2147
   Tb = 4.831*V(i)/((H(i)-1)^(0.30));
   ME(i) = X1(2);
   MV(i) = X2(2);
   WE(i) = X3(2);
   WV(i) = X4(2);
%
   if H(i) >= 17.40
     if Tb <= TDH
%                                                   deposition HW
       X4(1) = X;
       H4     = C*P4*C' + R;
       G4     = Y(i) - C*X4;
       K      = P4*C'*inv(H4);
       XN     = X4 + K*(G4);
       PN     = P4 - P4*C'*inv(H4)*C*P4;
       X4     = XN;
       P4     = PN;
       WV(i) = X4(2);
       CC(i) = X4(1);
       TERM  = ALFA*((Tb/TDH)-1)/H(i);
       A      = [1+X4(2)*TERM  X4(1)*TERM
                 0.0             1.0       ];
       L      = [(-1)*TERM*X4(1)*X4(2)
                 0.0                     ];
       XN     = A*X4 + L;
       PN     = A*P4*A' + Q4;
       X4     = XN;
       P4     = PN;
       X      = X4(1);
     else
       if Tb >= TEH
%                                                   erosion HW
         X2(1) = X;
         H2     = C*P2*C' + R;
         G2     = Y(i) - C*X2;
         K      = P2*C'*inv(H2);
         XN     = X2 + K*(G2);
         PN     = P2 - P2*C'*inv(H2)*C*P2;
         X2     = XN;
         P2     = PN;
         MV(i) = X2(2);
         CC(i) = X2(1);
         A      = [1.0  ALFA*((Tb/TEH)-1)/H(i)
                   0.0  1.0                        ];
         XN     = A*X2;
         PN     = A*P2*A' + Q2;
         X2     = XN;
         P2     = PN;
         X      = X2(1);
       else
%                                                   no erosion, no deposition HW
         if i > 1
           CC(i) = CC(i-1);
         else
           CC(i) = X;
         end
       end
     end
   else
%
%
     if Tb <= TDL
%                                                   deposition LW
       X3(1) = X;
       H3     = C*P3*C' + R;
       G3     = Y(i) - C*X3;
       K      = P3*C'*inv(H3);
       XN     = X3 + K*(G3);
       PN     = P3 - P3*C'*inv(H3)*C*P3;
       X3     = XN;
       P3     = PN;
       WE(i) = X3(2);
       CC(i) = X3(1);
       TERM  = ALFA*((Tb/TDH)-1)/H(i);
       A      = [1+X3(2)*TERM  X3(1)*TERM
                 0.0             1.0       ];
       L      = [(-1)*TERM*X3(1)*X3(2)
                 0.0                     ];
       XN     = A*X3 + L;
       PN     = A*P3*A' + Q3;
       X3     = XN;
       P3     = PN;
       X      = X3(1);
     else
       if Tb >= TEL
%                                                   erosion LW
         X1(1) = X;
         H1     = C*P1*C' + R;
         G1     = Y(i) - C*X1;
         K      = P1*C'*inv(H1);
         XN     = X1 + K*(G1);
         PN     = P1 - P1*C'*inv(H1)*C*P1;
         X1     = XN;
         P1     = PN;
         ME(i) = X1(2);
         CC(i) = X1(1);
         A      = [1.0  ALFA*((Tb/TEH)-1)/H(i)
                   0.0  1.0                        ];
         XN     = A*X1;
         PN     = A*P1*A' + Q1;
         X1     = XN;
         P1     = PN;
         X      = X1(1);
       else
%                                                   no erosion, no depositi

         if i > 1
           CC(i) = CC(i-1);
         else
           CC(i) = X;
         end
       end
     end
   end
end
% plot(T,ME)
% title('M_hat Erosion LW')
% pause
% plot(T,MV)
% title('M_hat Erosion HW')
% pause
% plot(T,WE)
% title('W_hat Sedimentation LW')
% pause
% plot(T,WV)
% title('W_hat Sedimentation HW')
% pause
plot(T,Y,':',T,CC)
title('C_hat - & Observations ..')
```

```
%kfmean
%
% Computes the mean values of the estimates of 'kalfil'
% started at k = 1000.
%
%
format long;
som1 = 0;
som2 = 0;
som3 = 0;
som4 = 0;
for i = 1:2147
   if i >= 1000
      S1   = ME(i) + som1;
      S2   = MV(i) + som2;
      S3   = WE(i) + som3;
      S4   = WV(i) + som4;
      som1 = S1;
      som2 = S2;
      som3 = S3;
      som4 = S4;
   end
end
M1 = som1/1148
pause
M2 = som2/1148
pause
W3 = som3/1148
pause
W4 = som4/1148
pause
```

```
%[rmeh1]
%
% Reads input of 'mehra3'
%
%
format long;
ydat;
vtwee;
hoog;
tijd;
TEL = input('Enter the Tce-LW (TEL): ')
TEH = input('Enter the Tce-HW (TEH): ')
TDL = input('Enter the Tcd-LW (TDL): ')
TDH = input('Enter the Tcd-HW (TDH): ')
ML  = input('Enter M-LW: ')
MH  = input('Enter M-HW: ')
WL  = input('Enter W-LW: ')
WH  = input('Enter W-HW: ')
```

```
%[mehra3]
%
% To identify the process noise covariance matrix using
% the iterative method, described by Mehra in 1970.
% (Uses 5 batches of 430 points).
%
%
%                                                        initial values
format long;
it  = input('Enter the number of the iteration to start with [first is 0):
Q1  = input('Enter the initial Q11-erosion-LW: ')
Q2  = input('Enter the initial Q11-erosion-HW: ')
Q3  = input('Enter the initial Q11-sedimentation-LW: ')
Q4  = input('Enter the initial Q11-sedimentation-HW: ')
X1   = 0.061;
X2   = X1;
X3   = X1;
X4   = X1;
R    = 0.7e-7;
ALFA = 724;
A1   = 1;
A2   = 1;
GA3  = 0;
GA4  = 0;
tel3 = 0;
tel4 = 0;
if it ~= 4
   grens = 430*(it + 1);
else
   grens = 2147;
end
for i = 1:grens
 if i >= it*430 + 1
 Tb = 4.831*V(i)/((H(i)-1)^(0.30));
 if H(i) >= 17.40
    if Tb <= TDH
    tn   = tel4 + 1;
    tel4 = tn;
    A4   = 1 + ALFA*WH*((Tb/TDH)-1)/H(i);
    GN   = A4 + GA4;
    GA4  = GN;
    end
 else
     if Tb <= TDL
     tn   = tel3+1;
     tel3 = tn;
     A3   = 1 + ALFA*WL*((Tb/TDL)-1)/H(i);
     GN   = A3 + GA3;
     GA3  = GN;
     end
 end
 end
end
PHI3 = GA3/tel3;
PHI4 = GA4/tel4;
MN1 = 0.5*(R*A1^2 + Q1 - R) + 0.5*sqrt((R-R*A1^2-Q1)^2 + 4*Q1*R);
MN2 = 0.5*(R*A2^2 + Q2 - R) + 0.5*sqrt((R-R*A2^2-Q2)^2 + 4*Q2*R);
MN3 = 0.5*(R*PHI3^2 + Q3 - R) + 0.5*sqrt((R-R*PHI3^2-Q3)^2 + 4*Q3*R);
MN4 = 0.5*(R*PHI4^2 + Q4 - R) + 0.5*sqrt((R-R*PHI4^2-Q4)^2 + 4*Q4*R);
KN1  = MN1*inv(MN1 + R);
KN2  = MN2*inv(MN2 + R);
KN3  = MN3*inv(MN3 + R);
KN4  = MN4*inv(MN4 + R);
h    = 0;
j    = 0;
k    = 0;
l    = 0;
P1   = MN1;
P2   = MN2;
P3   = MN3;
P4   = MN4;
%
%
%                                                        model
for i = 1:grens
 if i >= it*430 + 1
 Tb = 4.831*V(i)/((H(i)-1)^(0.30));
 if H(i) >= 17.40
    if Tb <= TDH
       lnew   = l+1;
       l      = lnew;
       AP4(l)= P4;
       n4(l) = Y(i) - X4;
       XN     = X4 + KN4*n4(l);
       PN     = P4 - P4*inv(P4+R)*P4;
       X4     = XN;
       P4     = PN;
       XN     = PHI4*X4;
       PN     = PHI4*P4*PHI4' + Q4;
       X4     = XN;
       P4     = PN;
    else
       if Tb >= TEH
          jnew   = j+1;
          j      = jnew;
          AP2(j)= P2;
          n2(j) = Y(i) - X2;
          XN     = X2 + KN2*n2(j);
          PN     = P2 - P2*inv(P2+R)*P2;
          X2     = XN;
          P2     = PN;
          XN     = A2*X2 + ALFA*MH*((Tb/TEH)-1)/H(i);
          PN     = A2*P2*A2' + Q2;
          X2     = XN;
          P2     = PN;
       end
    end
 else
    if Tb <= TEL
       knew   = k+1;
       k      = knew;
       AP3(k)= P3;
       n3(k) = Y(i) - X3;
       XN     = X3 + KN3*n3(k);
       PN     = P3 - P3*inv(P3+R)*P3;
       X3     = XN;
       P3     = PN;
       XN     = PHI3*X3;
       PN     = PHI3*P3*PHI3' + Q3;
       X3     = XN;
       P3     = PN;
    else
       if Tb >= TEL
          hnew   = h+1;
          h      = hnew;
          AP1(h)= P1;
          n1(h) = Y(i) - X1;
          XN     = X1 + KN1*n1(h);
          PN     = P1 - P1*inv(P1+R)*P1;
          X1     = XN;
          P1     = PN;
          XN     = A1*X1 + ALFA*ML*((Tb/TEL)-1)/H(i);
          PN     = A1*P1*A1' + Q1;
          X1     = XN;
          P1     = PN;
       end
    end
 end
 end
end
%
%                                                        ERROR ?
if h < 30
   disp('WARNING!')
   disp('N erosion LW < 30')
   h
end
if j < 30
   disp('WARNING!')
   disp('N erosion HW < 30')
   j
end
if k < 30
   disp('WARNING!')
   disp('N sedimentation LW < 30')
   k
end
if l < 30
   disp('WARNING!')
   disp('N sedimentation HW < 30')
   l
end
%
%                                                        C^1(0) & A.m
som = 0;
AMS = 0;
for i = 1:h
   E   = n1(i)*n1(i);
   S   = E + som;
   som = S;
   rest= AP1(i) +AMS;
   AMS = rest;
end
CN1  = som/h;
AE1  = AMS/h;
%                                                        ro_dakjel(k)
for I1 = 1:30
   som = 0;
   for i = I1+1:h
      E   = n1(i)*n1(i-I1);
      S   = E + som;
      som = S;
   end
   RO1(I1)   = som/(h*CN1);
   N1(I1)    = I1;
   PLS1(I1) = 1.96/sqrt(h);
   MIN1(I1) = -1.96/sqrt(h);
end
%
%                                                        C^2(0) & A.m
som = 0;
AMS = 0;
for i = 1:j
```

```
   E    = n2(i)*n2(i);
   S    = E + som;
   som = S;
   rest= AP2(i) +AMS;
   AMS = rest;
end
CN2  = som/j;
AE2  = AMS/j;
%                                                   ro_dakje2(k)
for I2 = 1:30
   som = 0;
   for i = I2+1:j
      E   = n2(i)*n2(i-I2);
      S   = E + som;
      som = S;
   end
   RO2(I2)  = som/(j*CN2);
   N2(I2)   = I2;
   PLS2(I2) = 1.96/sqrt(j);
   MIN2(I2) = -1.96/sqrt(j);
end
%
%                                                   C^3(0) & A.mse3
som = 0;
AMS = 0;
for i = 1:k
   E   = n3(i)*n3(i);
   S   = E + som;
   som = S;
   rest= AP3(i) +AMS;
   AMS = rest;
end
CN3  = som/k;
AE3  = AMS/k;
%                                                   ro_dakje3(k)
for I3 = 1:30
   som = 0;
   for i = I3+1:k
      E   = n3(i)*n3(i-I3);
      S   = E + som;
      som = S;
   end
   RO3(I3)  = som/(k*CN3);
   N3(I3)   = I3;
   PLS3(I3) = 1.96/sqrt(k);
   MIN3(I3) = -1.96/sqrt(k);
end
%
%                                                   C^4(0) & A.mse4
som = 0;
AMS = 0;
for i = 1:l
   E   = n4(i)*n4(i);
   S   = E + som;
   som = S;
   rest= AP4(i) +AMS;
   AMS = rest;
end
CN4 = som/l;
AE4 = AMS/l;
%                                                   ro_dakje4(k)
for I4 = 1:30
   som = 0;
   for i = I4+1:l
      E   = n4(i)*n4(i-I4);
      S   = E + som;
      som = S;
   end
   RO4(I4)  = som/(l*CN4);
   N4(I4)   = I4;
   PLS4(I4) = 1.96/sqrt(l);
   MIN4(I4) = -1.96/sqrt(l);
end
%
%
%                                                   to calculate Q^
T1    = KN1*CN1 + pinv(A1)*RO1(1)*CN1;
T2    = KN2*CN2 + pinv(A2)*RO2(1)*CN2;
T3    = KN3*CN3 + pinv(PHI3)*RO3(1)*CN3;
T4    = KN4*CN4 + pinv(PHI4)*RO4(1)*CN4;
qu10 = T1' - A1^2*T1 - A1*(-KN1*T1' - T1*KN1 + KN1^2*CN1)*A1;
QU1  = Q1 + (1/(it+1))*(qu10-Q1);
qu20 = T2' - A2^2*T2 - A2*(-KN2*T2' - T2*KN2 + KN2^2*CN2)*A2;
QU2  = Q2 + (1/(it+1))*(qu20-Q2);
qu30 = T3' - PHI3^2*T3 - PHI3*(-KN3*T3' - T3*KN3 + KN3^2*CN3)*PHI3;
QU3  = Q3 + (1/(it+1))*(qu30-Q3);
qu40 = T4' - PHI4^2*T4 - PHI4*(-KN4*T4' - T4*KN4 + KN4^2*CN4)*PHI4;
QU4  = Q4 + (1/(it+1))*(qu40-Q4);
%
%
%
%                                                      RESULTS!
disp('log-likelihood erosion LW:')
LIK1 = -CN1/(MN1+R) - log(abs(MN1+R))
pause
disp('Actual mse erosion LW:')
AE1
pause
disp('Calculated mse erosion LW:')
MN1
pause
disp('estimate of Q erosion LW:')
QU1
pause
disp('log-likelihood erosion HW:')
LIK2 = -CN2/(MN2+R) - log(abs(MN2+R))
pause
disp('Actual mse erosion HW:')
AE2
pause
disp('Calculated mse erosion HW:')
MN2
pause
disp('estimate of Q erosion HW:')
QU2
pause
disp('log-likelihood sedimentation LW:')
LIK3 = -CN3/(MN3+R) - log(abs(MN3+R))
pause
disp('Actual mse sedimentation LW:')
AE3
pause
disp('Calculated mse sedimentation LW:')
MN3
pause
disp('estimate of Q sedimentation LW:')
QU3
pause
disp('log-likelihood sedimentation HW:')
LIK4 = -CN4/(MN4+R) - log(abs(MN4+R))
pause
disp('Actual mse sedimentation HW:')
AE4
pause
disp('Calculated mse sedimentation HW:')
MN4
pause
disp('estimate of Q sedimentation HW:')
QU4
pause
plot(N1,RO1,'x',N1,PLS1,':',N1,MIN1,':')
title('ro Erosion LW it')
pause
plot(N2,RO2,'x',N2,PLS2,':',N2,MIN2,':')
title('ro Erosion HW it')
pause
plot(N3,RO3,'x',N3,PLS3,':',N3,MIN3,':')
title('ro Sedimentation LW it')
pause
plot(N4,RO4,'x',N4,PLS4,':',N4,MIN4,':')
title('ro Sedimentation HW it')
pause
```

```
%[ritil]
%
% Reads input of 'qutall'
%
%
format long;
ydat;
vtwee;
hoog;
tijd;
ML  = input('Enter the M-LW (ML):     ')
MH  = input('Enter the M-HW (MH):     ')
WL  = input('Enter the W-LW (WL):     ')
WH  = input('Enter the W-HW (WH):     ')
TEL = input('Enter the initial Tau-erosion LW (TEL):   ')
TEH = input('Enter the initial Tau-erosion HW (TEH):   ')
TDL = input('Enter the initial Tau-sedimentation LW (TDL):   ')
TDH = input('Enter the initial Tau-sedimentation HW (TDH):   ')
qu1 = input('Enter q22 erosion LW (qu1):    ')
qu2 = input('Enter q22 erosion HW (qu2):    ')
qu3 = input('Enter q22 sedimentation LW (qu3):    ')
qu4 = input('Enter q22 sedimentation HW (qu4):    ')
```

```
%[qutall]
%
% Determines the process noise covariance matrix (to estimate
% Tau-critical).
% Shows the likelihood functions of Q11, with known Q22.
%
%
format long;
R    = 7.0e-8;
RUIS = 10e-11;
C    = [1 0];
ALFA = 724;
for j = 1:10
   RN(j)= 10*RUIS;
   RUIS = RN(j);
   Q1   = [RUIS 0.0
           0.0  qu1];
   Q2   = [RUIS 0.0
           0.0  qu2];
   Q3   = [RUIS 0.0
           0.0  qu3];
   Q4   = [RUIS 0.0
           0.0  qu4];
   TEL  = 0.14;
   TEH  = 0.20;
   TDL  = 0.08;
   TDH  = 0.08;
   X    = 0.061;
   X1   = [X
           TEL];
   X2   = [X
           TEH];
   X3   = [X
           TDL];
   X4   = [X
           TDH];
   P1   = [R    0.0
           0.0  1.0];
   P2   = P1;
   P3   = P1;
   P4   = P1;
   LIK1 = 0.0;
   LIK2 = 0.0;
   LIK3 = 0.0;
   LIK4 = 0.0;
   HOD1 = 0.0;
   HOD2 = 0.0;
   HOD3 = 0.0;
   HOD4 = 0.0;
   lik1 = 0.0;
   lik2 = 0.0;
   lik3 = 0.0;
   lik4 = 0.0;
   hod1 = 0.0;
   hod2 = 0.0;
   hod3 = 0.0;
   hod4 = 0.0;
   for i = 1:2147
     Tb = 4.831*V(i)/((H(i)-1)^(0.30));
%
%
%
     if H(i) >= 17.40
        if Tb <= TDH
%                                                    deposition HW
           X4(1) = X;
           H4    = C*P4*C' + R;
           G4    = Y(i) - C*X4;
           lik4  = log(det(H4));
           hod4  = G4'*inv(H4)*G4;
           K     = P4*C'*inv(H4);
           XN    = X4 + K*(G4);
           PN    = P4 - P4*C'*inv(H4)*C*P4;
           X4    = XN;
           P4    = PN;
           TDH   = X4(2);
           TERM  = ALFA*WH/H(i);
           A     = [1+TERM*((Tb/X4(2))-1)   -X4(1)*TERM*Tb/(X4(2))^2
                    0.0                      1.0                      ];
           L     = [TERM*X4(1)*Tb/X4(2)
                    0.0                 ];
           XN    = A*X4 + L;
           PN    = A*P4*A' + Q4;
           X4    = XN;
           P4    = PN;
           X     = X4(1);
        else
           if Tb >= TEH
%                                          erosion HW
              X2(1) = X;
              H2    = C*P2*C' + R;
              G2    = Y(i) - C*X2;
              lik2  = log(det(H2));
              hod2  = G2'*inv(H2)*G2;
              K     = P2*C'*inv(H2);
              XN    = X2 + K*(G2);
              PN    = P2 - P2*C'*inv(H2)*C*P2;
              X2    = XN;
              P2    = PN;
              TEH   = X2(2);
              TERM  = ALFA*MH/H(i);
              A     = [1.0  -TERM*Tb/(X2(2))^2
                        0.0  1.0                ];
              L     = [TERM*(2*(Tb/X2(2))-1)
                        0.0                   ];
              XN    = A*X2 + L;
              PN    = A*P2*A' + Q2;
              X2    = XN;
              P2    = PN;
              X     = X2(1);
%          else
%                                          no erosion, no deposition HW
           end
        end
     else
%
%
        if Tb <= TDL
%                                          deposition LW
           X3(1) = X;
           H3    = C*P3*C' + R;
           G3    = Y(i) - C*X3;
           lik3  = log(det(H3));
           hod3  = G3'*inv(H3)*G3;
           K     = P3*C'*inv(H3);
           XN    = X3 + K*(G3);
           PN    = P3 - P3*C'*inv(H3)*C*P3;
           X3    = XN;
           P3    = PN;
           TDL   = X3(2);
           TERM  = ALFA*WL/H(i);
           A     = [1+TERM*((Tb/X3(2))-1)   -X3(1)*TERM*Tb/(X3(2))^2
                    0.0                      1.0                      ];
           L     = [TERM*X3(1)*Tb/X3(2)
                    0.0                 ];
           XN    = A*X3 + L;
           PN    = A*P3*A' + Q3;
           X3    = XN;
           P3    = PN;
           X     = X3(1);
        else
           if Tb >= TEL
%                                          erosion LW
              X1(1) = X;
              H1    = C*P1*C' + R;
              G1    = Y(i) - C*X1;
              lik1  = log(det(H1));
              hod1  = G1'*inv(H1)*G1;
              K     = P1*C'*inv(H1);
              XN    = X1 + K*(G1);
              PN    = P1 - P1*C'*inv(H1)*C*P1;
              X1    = XN;
              P1    = PN;
              TEL   = X1(2);
              TERM  = ALFA*ML/H(i);
              A     = [1.0  -TERM*Tb/(X1(2))^2
                        0.0  1.0                ];
              L     = [TERM*2*((Tb/X1(2))-1)
                        0.0                   ];
              XN    = A*X1 + L;
              PN    = A*P1*A' + Q1;
              X1    = XN;
              P1    = PN;
              X     = X1(1);
%          else
%                                          no erosion, no deposition LW
           end
        end
     end
     LIK1 = lik1 + LIK1;
     LIK2 = lik2 + LIK2;
     LIK3 = lik3 + LIK3;
     LIK4 = lik4 + LIK4;
     HOD1 = hod1 + HOD1;
     HOD2 = hod2 + HOD2;
     HOD3 = hod3 + HOD3;
     HOD4 = hod4 + HOD4;
   end
%
%
%                                          bepalen van L-
   ML1(j) = 0.5*(LIK1 + HOD1);
   ML2(j) = 0.5*(LIK2 + HOD2);
   ML3(j) = 0.5*(LIK3 + HOD3);
   ML4(j) = 0.5*(LIK4 + HOD4);
end
semilogx(RN,ML1)
title('Likelihood Q11 Erosion LW')
% pause
% semilogx(RN,ML2)
% title('Likelihood Q11 Erosion HW')
% pause
% semilogx(RN,ML3)
% title('Likelihood Q11 Sedimentation LW')
% pause
% semilogx(RN,ML4)
% title('Likelihood Q11 Sedimentation HW')
% pause
```

```
% rit22
%
% Reads input of 'quta22'
%
%
format long;
ydat;
vtwee;
hoog;
tijd;
ML= input('Enter the M-LW (ML):     ')
MH= input('Enter the M-HW (MH):     ')
WL= input('Enter the W-LW (WL):     ')
WH= input('Enter the W-HW (WH):     ')
TEL = input('Enter the initial Tau-erosion LW (TEL):  ')
TEH = input('Enter the initial Tau-erosion HW (TEH):  ')
TDL = input('Enter the initial Tau-sedimentation LW (TDL):  ')
TDH = input('Enter the initial Tau-sedimentation HW (TDH):  ')
qu1= input('Enter q11 erosion LW (qu1):    ')
qu2= input('Enter q11 erosion HW (qu2):    ')
qu3= input('Enter q11 sedimentation LW (qu3):    ')
qu4= input('Enter q11 sedimentation HW (qu4):    ')
```

```
% quta22
%
% Determines the process noise covariance matrix (to estimate
% Tau-critical).
% Shows the likelihood functions of Q22, with known Q11.
%
%
format long;
R    = 7.0e-8;
RUIS = 10e-11;
C    = [1 0];
ALFA = 724;
for j = 1:10
   RN(j)= 10*RUIS;
   RUIS = RN(j);
   Q1   = [qu1  0.0
           0.0  RUIS];
   Q2   = [qu2  0.0
           0.0  RUIS];
   Q3   = [qu3  0.0
           0.0  RUIS];
   Q4   = [qu4  0.0
           0.0  RUIS];
   TEL  = 0.14;
   TEH  = 0.20;
   TDL  = 0.08;
   TDH  = 0.08;
   X    = 0.061;
   X1   = [X
           TEL];
   X2   = [X
           TEH];
   X3   = [X
           TDL];
   X4   = [X
           TDH];
   P1   = [R    0.0
           0.0  1.0];
   P2   = P1;
   P3   = P1;
   P4   = P1;
   LIK1 = 0.0;
   LIK2 = 0.0;
   LIK3 = 0.0;
   LIK4 = 0.0;
   HOD1 = 0.0;
   HOD2 = 0.0;
   HOD3 = 0.0;
   HOD4 = 0.0;
   lik1 = 0.0;
   lik2 = 0.0;
   lik3 = 0.0;
   lik4 = 0.0;
   hod1 = 0.0;
   hod2 = 0.0;
   hod3 = 0.0;
   hod4 = 0.0;
   for i = 1:2147
     Tb = 4.831*V(i)/((H(i)-1)^(0.30));
%
%
%
     if H(i) >= 17.40
        if Tb <= TDH
%                                                deposition HW
           X4(1) = X;
           H4    = C*P4*C' + R;
           G4    = Y(i) - C*X4;
           lik4  = log(det(H4));
           hod4  = G4'*inv(H4)*G4;
           K     = P4*C'*inv(H4);
           XN    = X4 + K*(G4);
           PN    = P4 - P4*C'*inv(H4)*C*P4;
           X4    = XN;
           P4    = PN;
           TDH   = X4(2);
           TERM  = ALFA*WH/H(i);
           A     = [1+TERM*((Tb/X4(2))-1)   -X4(1)*TERM*Tb/(X4(2))^2
                     0.0                     1.0                       ];
           L     = [TERM*X4(1)*Tb/X4(2)
                     0.0                 ];
           XN    = A*X4 + L;
           PN    = A*P4*A' + Q4;
           X4    = XN;
           P4    = PN;
           X     = X4(1);
        else
           if Tb >= TEH
%                                                erosion HW
              X2(1) = X;
              H2    = C*P2*C' + R;
              G2    = Y(i) - C*X2;
              lik2  = log(det(H2));
              hod2  = G2'*inv(H2)*G2;
              K     = P2*C'*inv(H2);
              XN    = X2 + K*(G2);
              PN    = P2 - P2*C'*inv(H2)*C*P2;
              X2    = XN;
              P2    = PN;
              TEH   = X2(2);
              TERM  = ALFA*MH/H(i);
              A     = [1.0  -TERM*Tb/(X2(2))^2
                        0.0  1.0                ];
              L     = [TERM*(2*(Tb/X2(2))-1)
                        0.0                ];
              XN    = A*X2 + L;
              PN    = A*P2*A' + Q2;
              X2    = XN;
              P2    = PN;
              X     = X2(1);
%          else
%                                                no erosion, no deposition HW
           end
        end
     else
%
%
        if Tb <= TDL
%                                                deposition LW
           X3(1) = X;
           H3    = C*P3*C' + R;
           G3    = Y(i) - C*X3;
           lik3  = log(det(H3));
           hod3  = G3'*inv(H3)*G3;
           K     = P3*C'*inv(H3);
           XN    = X3 + K*(G3);
           PN    = P3 - P3*C'*inv(H3)*C*P3;
           X3    = XN;
           P3    = PN;
           TDL   = X3(2);
           TERM  = ALFA*WL/H(i);
           A     = [1+TERM*((Tb/X3(2))-1)   -X3(1)*TERM*Tb/(X3(2))^2
                     0.0                     1.0                       ];
           L     = [TERM*X3(1)*Tb/X3(2)
                     0.0                 ];
           XN    = A*X3 + L;
           PN    = A*P3*A' + Q3;
           X3    = XN;
           P3    = PN;
           X     = X3(1);
        else
           if Tb >= TEL
%                                                erosion LW
              X1(1) = X;
              H1    = C*P1*C' + R;
              G1    = Y(i) - C*X1;
              lik1  = log(det(H1));
              hod1  = G1'*inv(H1)*G1;
              K     = P1*C'*inv(H1);
              XN    = X1 + K*(G1);
              PN    = P1 - P1*C'*inv(H1)*C*P1;
              X1    = XN;
              P1    = PN;
              TEL   = X1(2);
              TERM  = ALFA*ML/H(i);
              A     = [1.0  -TERM*Tb/(X1(2))^2
                        0.0  1.0                ];
              L     = [TERM*2*((Tb/X1(2))-1)
                        0.0                ];
              XN    = A*X1 + L;
              PN    = A*P1*A' + Q1;
              X1    = XN;
              P1    = PN;
              X     = X1(1);
%          else
%                                                no erosion, no deposition LW
           end
        end
     end
     LIK1 = lik1 + LIK1;
     LIK2 = lik2 + LIK2;
     LIK3 = lik3 + LIK3;
     LIK4 = lik4 + LIK4;
     HOD1 = hod1 + HOD1;
     HOD2 = hod2 + HOD2;
     HOD3 = hod3 + HOD3;
     HOD4 = hod4 + HOD4;
   end
%
%
%                                                bepalen van L~
   ML1(j) = 0.5*(LIK1 + HOD1);
   ML2(j) = 0.5*(LIK2 + HOD2);
   ML3(j) = 0.5*(LIK3 + HOD3);
   ML4(j) = 0.5*(LIK4 + HOD4);
end
semilogx(RN,ML1)
title('Likelihood Q22 Erosion LW')
% pause
% semilogx(RN,ML2)
% title('Likelihood Q22 Erosion HW')
% pause
% semilogx(RN,ML3)
% title('Likelihood Q22 Sedimentation LW')
% pause
% semilogx(RN,ML4)
% title('Likelihood Q22 Sedimentation HW')
% pause
```

```
%rkalf2
%
% Reads input of 'kalfil2'
%
%
format long;
ydat;
vtwee;
hoog;
tijd;
ML = input('Enter the M-LW (ML):    ')
MH = input('Enter the M-HW (MH):    ')
WL = input('Enter the W-LW (WL):    ')
WH = input('Enter the W-HW (WH):    ')
TEL= input('Enter the initial Tau-erosion LW (TEL):   ')
TEH= input('Enter the initial Tau-erosion HW (TEH):   ')
TDL= input('Enter the initial Tau-sedimentation LW (TDL):   ')
TDH= input('Enter the initial Tau-sedimentation HW (TDH):   ')
Q1 = input('Enter Q erosion LW (Q1):    ')
Q2 = input('Enter Q erosion HW (Q2):    ')
Q3 = input('Enter Q sedimentation LW (Q3):    ')
Q4 = input('Enter Q sedimentation HW (Q4):    ')
```

```
%kalfil2
%
% Estimates Tau-critical using the method of extended Kalman filtering.
% Q, M and W can be read with 'kalf2'
%
%
%                                          initial values
format long;
R     = 7.0e-6;
C     = [1 0];
ALFA  = 724;
X     = 0.061;
X1    = [X
         TEL];
X2    = [X
         TEH];
X3    = [X
         TDL];
X4    = [X
         TDH];
P1    = [R     0.0
         0.0   1.0];
P2    = P1;
P3    = P1;
P4    = P1;
for i = 1:2147
      Tb = 4.831*V(i)/((H(i)-1)^(0.30));
      TU1(i) = X1(2);
      TU2(i) = X2(2);
      TU3(i) = X3(2);
      TU4(i) = X4(2);
%
      if H(i) >= 17.40
         if Tb <= TDH
%                                                       deposition HW
            X4(1) = X;
            H4    = C*P4*C' + R;
            G4    = Y(i) - C*X4;
            K     = P4*C'*inv(H4);
            XN    = X4 + K*(G4);
            PN    = P4 - P4*C'*inv(H4)*C*P4;
            X4    = XN;
            P4    = PN;
            TDH   = X4(2);
            TU4(i)= X4(2);
            CC(i) = X2(1);
            TERM  = ALFA*WH/H(i);
            A     = [1+TERM*((Tb/X4(2))-1)   -X4(1)*TERM*Tb/(X4(2))^2
                     0.0                      1.0                      ];
            L     = [TERM*X4(1)*Tb/X4(2)
                     0.0                    ];
            XN    = A*X4 + L;
            PN    = A*P4*A' + Q4;
            X4    = XN;
            P4    = PN;
            X     = X4(1);
         else
            if Tb >= TEH
%                                                       erosion HW
               X2(1) = X;
               H2    = C*P2*C' + R;
               G2    = Y(i) - C*X2;
               K     = P2*C'*inv(H2);
               XN    = X2 + K*(G2);
               PN    = P2 - P2*C'*inv(H2)*C*P2;
               X2    = XN;
               P2    = PN;
               TU2(i)= X2(2);
               CC(i) = X2(1);
               TEH   = X2(2);
               TERM  = ALFA*MH/H(i);
               A     = [1.0  -TERM*Tb/(X2(2))^2
                        0.0  1.0                  ];
               L     = [TERM*(2*(Tb/X2(2))-1)
                        0.0                    ];
               XN    = A*X2 + L;
               PN    = A*P2*A' + Q2;
               X2    = XN;
               P2    = PN;
               X     = X2(1);
            else
%                                                       no erosion, no deposition
               if i > 1
                  CC(i) = CC(i-1);
               else
                  CC(i) = X;
               end
            end
         end
      else
%
%
         if Tb <= TDL
%                                                       deposition LW
            X3(1) = X;
            H3    = C*P3*C' + R;
            G3    = Y(i) - C*X3;
            K     = P3*C'*inv(H3);
            XN    = X3 + K*(G3);
            PN    = P3 - P3*C'*inv(H3)*C*P3;
            X3    = XN;
            P3    = PN;
            TU3(i)= X3(2);
            CC(i) = X3(1);
            TDL   = X3(2);
            TERM  = ALFA*WL/H(i);
            A     = [1+TERM*((Tb/X3(2))-1)   -X3(1)*TERM*Tb/(X3(2))^2
                     0.0                      1.0
            L     = [TERM*X3(1)*Tb/X3(2)
                     0.0                    ];
            XN    = A*X3 + L;
            PN    = A*P3*A' + Q3;
            X3    = XN;
            P3    = PN;
            X     = X3(1);
         else
            if Tb >= TEL
%                                                       erosion LW
               X1(1) = X;
               H1    = C*P1*C' + R;
               G1    = Y(i) - C*X1;
               K     = P1*C'*inv(H1);
               XN    = X1 + K*(G1);
               PN    = P1 - P1*C'*inv(H1)*C*P1;
               X1    = XN;
               P1    = PN;
               TU1(i)= X1(2);
               CC(i) = X1(1);
               TEL   = X1(2);
               TERM  = ALFA*ML/H(i);
               A     = [1.0  -TERM*Tb/(X1(2))^2

                        0.0  1.0                  ];
               L     = [TERM*2*((Tb/X1(2))-1)
                        0.0                    ];
               XN    = A*X1 + L;
               PN    = A*P1*A' + Q1;
               X1    = XN;
               P1    = PN;
               X     = X1(1);
            else
%                                                       no erosion, no deposition LW
               if i > 1
                  CC(i) = CC(i-1);
               else
                  CC(i) = X;
               end
            end
         end
      end
end
%
plot(T,Y,':',T,CC)
title('C_hat - & Observations ..')
% plot(T,TU1)
% title('Tau-ce_hat Erosion LW')
% pause
% plot(T,TU2)
% title('Tau-ce_hat Erosion HW')
% pause
% plot(T,TU3)
% title('Tau-cd_hat Sedimentation LW')
% pause
% plot(T,TU4)
% title('Tau-cd_hat Sedimentation HW')
% pause
```

```
%kfmean2
%
% Computes the mean values of the estimates of 'kalfil2',
% started with k = 1000.
%
%
format long;
som1 = 0;
som2 = 0;
som3 = 0;
som4 = 0;
for i = 1:2147
   if i >= 1000
      S1   = TU1(i) + som1;
      S2   = TU2(i) + som2;
      S3   = TU3(i) + som3;
      S4   = TU4(i) + som4;
      som1 = S1;
      som2 = S2;
      som3 = S3;
      som4 = S4;
   end
end
TAU1 = som1/1148
pause
TAU2 = som2/1148
pause
TAU3 = som3/1148
pause
TAU4 = som4/1148
pause
```

```
% rgeg
%
% Reads input of 'model'
%
%
format long;
ydat;
vtwee;
hoog;
tijd;
```

```
% model
%
% Model, input read by 'rgeg'
% Tau-b with a time-delay
%
  format long
  TEE = input('Enter Tce_eb:   ')
  TEV = input('Enter Tce_vload:    ')
  TDE = input('Enter Tcd_eb:   ')
  TDV = input('Enter Tcd_vload:  ')
  ME  = input('Enter Meb:    ')
  MV  = input('Enter Mvload:   ')
  WE  = input('Enter Web:   ')
  WV  = input('Enter Wvload:   ')
  B   = input('Enter the initial C:    ')
  ALFA = 724;
  TAUB = 0;
  for i = 1:2147
    if i >= 50
      l = i-49;
      TT(l) = T(i);
      YY(l) = Y(i);
      if H(i) >= 17.40
         j = round(0.03/(sqrt(V(i))*0.008380));
         if (i-j)>0
           TAUB = 4.831*V(i-j)*((H(i-j)-1)^(-0.30));
         end
         if TAUB <= TDV
            BN = B + ALFA*B*WV*((TAUB/TDV)-1)/H(i);
            B  = BN;
         else
            if TAUB >= TEV
               BN = B + ALFA*MV*((TAUB/TEV) - 1)/H(i);
               B  = BN;
            end
         end
      else
         j = round(0.13/(sqrt(V(i))*0.008380));
         if (i-j) > 0
           TAUB = 4.831*V(i-j)*((H(i-j)-1)^(-0.30));
         end
         if TAUB <= TDE
            BN = B + ALFA*B*WE*((TAUB/TDE)-1)/H(i);
            B  = BN;
         else
            if TAUB >= TEE
               BN = B + ALFA*ME*((TAUB/TEE)-1)/H(i);
               B  = BN;
            end
         end
      end
      W(l) = B;
    end
  end
  plot(TT,YY,':',TT,W)
```

Ministerie van Verkeer en Waterstaat

Directoraat-Generaal Rijkswaterstaat

Rijksinstituut voor Kust en Zee/*RIKZ (voorheen Dienst Getijdewateren)*

ONDERZOEK NAAR TRENDMATIGE VERANDERINGEN VAN SLIBCONCENTRATIES IN DE WESTERSCHELDE

Een onderzoeksrapport

Middelburg, 16 augustus 1994

Omschrijving van het onderzoek

Het onderzoek is erop gericht om een trend te vinden in het slibgehalte in de Westerschelde over de periode 1970-1990. Gedurende deze 20 jaar werden op tenminste negen plaatsen in de Westerschelde 1 à 2 maal per week, deels op willekeurige tijdstippen in het getij, watermonsters genomen om het slibgehalte, het chloridegehalte en de temperatuur te meten. Dit om trendveranderingen in het slibgehalte t.g.v. het menselijk ingrijpen (baggeren, storten, etc.) op te sporen.

Deze metingen blijken weinig bruikbaar voor trendanalyse, doordat de spreiding groot is. Dit heeft diverse oorzaken. Continue metingen, d.w.z. metingen om de tien minuten op drie lokaties, tonen een sterke afhankelijkheid van het slibgehalte van de getijfase cq. de stroomsnelheid aan. Bovendien treden gedurende een springtij-doodtijcyclus extra veranderingen op. Behalve getij-invloeden zijn ook seizoensinvloeden waarneembaar. In de winter is de afvoer van de Schelde in de regel groter dan in de zomer en is er meer slib in suspensie t.g.v. de golfwerking. Een trendmatige verandering zal slechts aantoonbaar zijn wanneer deze getij- en seizoensinvloeden worden geëlimineerd.

Eliminatie verstorende factoren

Voor de eliminatie van het getij wordt een model opgesteld dat weergeeft hoe de slibconcentratie afhangt van de stroomsnelheid. Nemen we nu aan dat er een verband is tussen de stroomsnelheid en het getijverschil, dan is het getij grotendeels uit de 20-jarige reeks te elimineren. Dit idee is niet nieuw, maar de uitwerking wel. Het gevonden verband zal afhankelijk zijn van de geometrie op de plaats van de meting. Verder zal de ebstroom mogelijk een ander verband laten zien dan de vloedstroom. Het model wordt daarom ook geijkt op 13-uurs metingen die in de buurt van een meetpunt uitgevoerd zijn.

M.b.v. dit model wordt naast de dagelijkse variatie van het getij en het doodtij-springtijeffect ook de 18,6-jarige cyclus geëlimineerd. Nadat de metingen gecorrigeerd zijn, d.w.z. nadat de getij-invloed geëlimineerd is, kan de seizoensinvloed op het slibgehalte worden bepaald. Na eliminatie van deze seizoensinvloed op de twintig-jarige reeks ontstaat een beeld van het verloop van de slibconcentratie over een aantal jaren in de Westerschelde.

Begonnen wordt met het opstellen van een model aan de hand van de 10-minuten-metingen gedaan bij Bath in winter '88/'89. Uit deze metingen wordt zowel het effect van de dagelijkse ongelijkheid als het doodtij-springtijeffect op het slibgehalte nauwkeurig bepaald. Het hiermee opgestelde model kan verder aangepast worden voor de 10-minuten-metingen gedaan bij het Middelgat (winter '87/-'88) en nabij Vlissingen (winter '89/'90).

Getij-eliminatie / Resultaten en tekortkomingen eerste slibmodel

In voorgaande onderzoeken is de relatie tussen het slibgehalte en het getij geschat m.b.v. correlatiemodellen. Ook is er gezocht naar een zekere trend in het verloop van de slibconcentratie zonder het getij te elimineren. Bij deze benaderingen is de spreiding groot.

Om de trendanalyse nu eens op een andere wijze uit te voeren, worden eerst de onbekende parameters van een model geschat m.b.v. het zgn. (extended) Kalman-filter en vervolgens worden deze resultaten gebruikt om de ruis ten gevolge van het getij op de metingen te corrigeren. Het is de eerste keer dat dit probleem op deze manier wordt aangepakt, d.w.z. met een stochastische modelanalyse.

Om de methode te ontwikkelen is eerst een eenvoudig model gebouwd voor de relatie tussen het slibgehalte en de stroomsnelheid. Het eerste model legt een (niet-lineair) verband tussen stroomsnelheid en slibgehalte op basis van een eenvoudig massabalansmodel voor een waterkolom. Dit model is gebaseerd op het principe dat de totale hoeveelheid slib in de waterkolom alleen beïnvloed wordt door de erosie en depositie die aan de bodem plaatsvindt en dat de verdeling van het slibgehalte over de kolom homogeen is. De verandering van de hoeveelheid slib in een waterkolom is dan gelijk aan de hoeveelheid geërodeerd slib aan de bodem minus de hoeveelheid gesedimenteerd slib, gedeeld door de hoogte van de waterkolom. In formule luidt dit:

$dC/dt = (E - D)/H$ 　　　E = erosieterm; D = sedimentatieterm;
H = waterhoogte

Er is bij deze opzet gebruik gemaakt van zgn. erosie- en depositiemodellen:

$E = M(\tau_b/\tau_{ce} - 1)$ voor $\tau_b \geq \tau_{ce}$

$= 0$ elders

$D = CW_s(\tau_b/\tau_{cd} - 1)$ voor $\tau_b \leq \tau_{cd}$

$= 0$ elders

De erosie-en depositiemodellen bevatten vier parameters die met een zgn. Kalmanfilter geschat worden. Er wordt vanuit gegaan dat deze parameters per lokatie constant zullen zijn. Constante parameters leveren immers een eenvoudig model op waaruit de variatie ten gevolge van het getij makkelijk te herleiden is.

Op dit moment wordt er door een afstudeerder van de UT onderzoek verricht naar het opstellen en verbeteren van het toegepaste Kalmanfilter, zodat de parameters van het model nauwkeurig geschat kunnen worden. Deze benadering geeft al in de beginfase opmerkelijke resultaten, zie figuur 1 t/m 5.

Voor het afstuderen wordt alleen het model voor de lokatie Bath opgesteld en worden er ideeën ontwikkeld voor het aanpassen van het model voor de situaties bij het Middelgat en bij Vlissingen. Verder wordt er een computerprogramma voor het schatten van de parameters geleverd, dat zal worden voorzien van een gebruikershandleiding. Dit programma kan na al dan niet aangepast te zijn, gebruikt worden bij verder onderzoek en kan in de toekomst nuttig zijn bij het doen van voorspellingen.

Getij-eliminatie / Verder onderzoek

Het afstudeerwerk beperkt zich tot de situatie bij Bath, terwijl er ook 10-minuten-waarden beschikbaar zijn van metingen nabij het Middelgat en bij Vlissingen. Op deze twee plaatsen is er sprake van een andere situatie: er is bijv. duidelijk verschil in slibconcentratie tussen de verschillende lagen van het water. Hiervoor moet het model uitgebreid worden met een variatie van het sedimentgehalte over de diepte. Er zullen meerdere parameters geschat moeten worden. Dit levert het definitieve model.

Nadat de aard en de stabiliteit van de parameters van de modellen op de drie bovengenoemde lokaties geïdentificeerd zijn, kunnen met de gegevens van incidentele 13-uurs metingen over de periode 1970-1982 op 9 lokaties de parameters geschat worden. Deze lokaties liggen verspreid over de hele Westerschelde. Op deze lokaties is bij de 13-uurs meting simultaan slib en stroomsnelheid gemeten. Met de modellen kan voor een meetdag de invloed van de dagelijkse variatie van het getij op het slibgehalte worden geëlimineerd.

Resteren het effect van het dagelijks getij op andere dagen dan de meetdag, het springtij-doodtijeffect en de 18,6-jarige cyclus. Om dit te elimineren uit de 14-daagse metingen zijn er stroomsnelheden op de meetdata in 1970-1990 op deze negen lokaties nodig. Omdat deze stroomsnelheden niet gemeten zijn, moeten ze op een andere manier bepaald worden. Hiervoor gaan we uit van het gemeten getijverschil bij Vlissingen in dezelfde periode. Nu kunnen we, gebruikmakend van het feit dat er in estuaria sprake is van een lineair verband tussen de stroomsnelheid en het getijverschil, uit het getijverschil bij Vlissingen op de negen lokaties de benodigde stroomsnelheden berekenen. Vullen we de snelheid in het model in dan wordt een algoritme verkregen dat de slibconcentratie over de jaren 1970-1990 normeert naar een willekeurig tijdstip. Toepassen van het algoritme op de metingen, levert zo een reeks metingen waarop de ruis van het getij is gecorrigeerd.

Naast de genoemde meetwaarden zijn er ook zgn. WORSRO-meetgegevens voorhanden over de periode 1982-1993. Hierbij is er eens in de veertien dagen bij laagwaterkentering gemeten. Bij deze metingen werd o.a. het zwevende stof-gehalte bepaald. Dit is geen aanwijsbare maat is voor het slibgehalte [zie nota OW-90.064, Maldegem & Storm] en er zal daarentegen geen gebruik worden gemaakt van de WORSRO-metingen.

Seizoenseliminatie

De seizoensinvloed wordt bepaald aan de hand van de reeds genoemde 13-uurs metingen die op 9 lokaties in de Westerschelde over de periode 1970-1982 zijn vericht. We gaan de gegevens per lokatie bekijken op een vast punt in het getij, bijv. laag-water kentering en we berekenen op basis van deze gegevens per maand een gemiddelde over alle jaren van de concentratie. Zo ontstaan per lokatie voor iedere maand gemiddelde sediment-concentraties gemeten bij laagwater kentering over een aantal jaren. Vanwege de uitgevoerde normering en de middeling over de jaren is de invloed van het getij hieruit verdwenen en hebben we een indicatie verkregen van de seizoensinvloed over deze jaren per lo catie. Hiermee zal de seizoensinvloed uit de reeks gegevens van 1970-1990, waaruit we reeds de getij-invloed haddden geëlimineerd, geëlimineerd gaan worden.

Trend

Met deze genormeerde gegevens kan een beeld verkregen worden van de trend van het slibgehalte over de periode 1970-1990. En kan dus o.a. bekeken worden of de baggerwerkzaamheden, begonnen in de jaren '70 en verminderd vanaf '85, invloed hebben gehad op het slibgehalte in de Westerschelde.

Een dergelijke aanpak zou niet alleen resultaat kunnen hebben in de Westerschelde, maar zou misschien ook aangepast kunnen worden voor de Waddenzee.

Gebruikte metingen / Modellen:

Opstellen slibmodel voor de 3 lokaties:
Bath (winter '88/'89)
Middelgat (winter '87/'88)
Vlissingen (winter '89/'90)

Opstellen modellen voor 9 lokaties
a.d.h.v. 13-uurs metingen 1970-1982

Opstellen modellen voor 9 lokaties
a.d.h.v. de 14-daagse metingen over 1970-1990
uitgaande van de 9 reeds opgestelde modellen

Elimineren van het getij-effect op de negen
meetreeksen over 1970-1990

Bepalen seizoensinvloeden

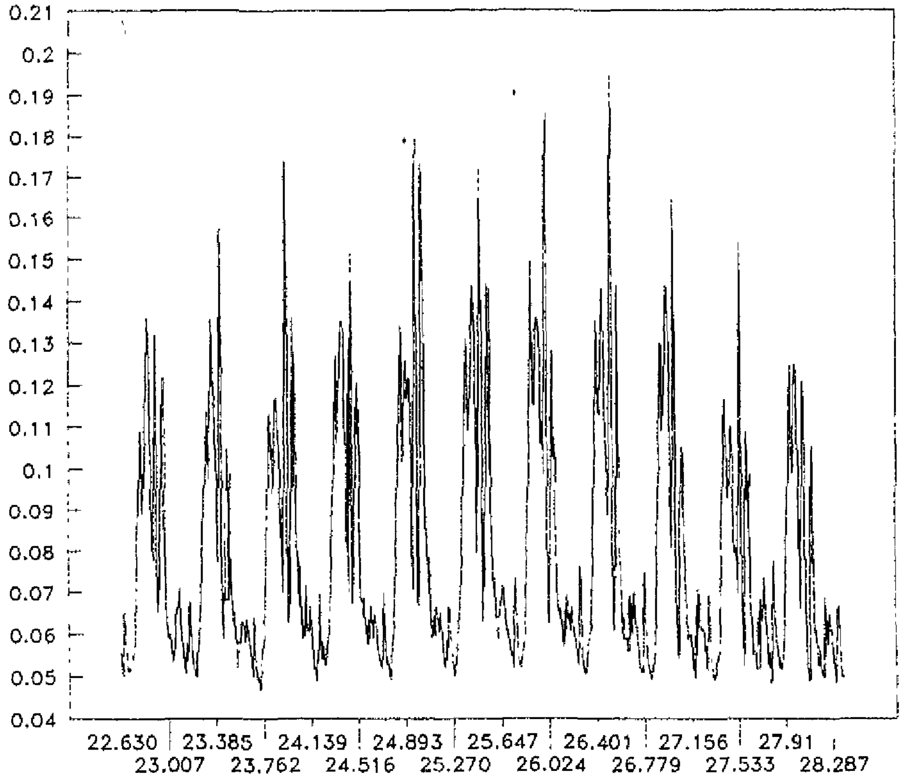
Elimineren seizoensinvloeden
reeks 1970-1990

Trend over 1970-1990

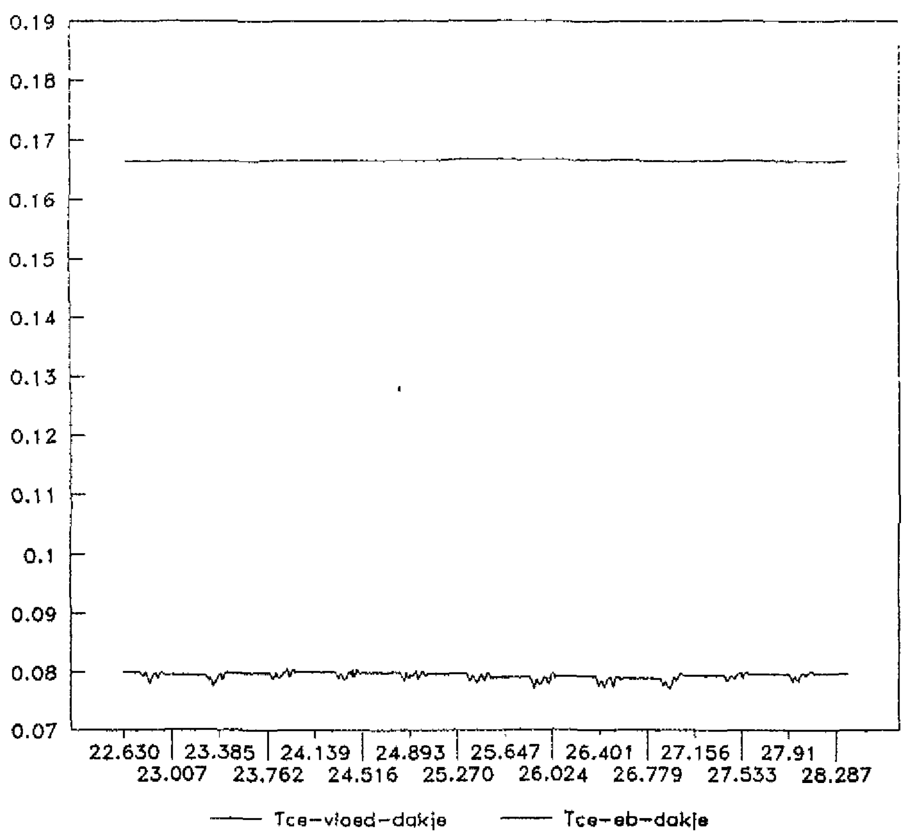
Verwachte werkzaamheden

- Uitbreiden van het massabalansmodel voor drie lagen water, waarbij de variatie van het slibgehalte over de diepte meegenomen wordt.	4 wkn
- Aanpassen van het bestaande computerprogramma aan het nieuwe model.	1 week
- Schatten van de modelparameters voor de meetwaarden bij het Middelgat.	3 wkn
- Zoeken van een representatieve reeks uit de reeks 10-minuten-metingen nabij Vlissingen, waarin tenminste een springtij-doodtijperiode voorkomt. En aanpassen van deze reeks om gebruikt te kunnen worden in de ontworpen programmatuur.	1 week
- Bekijken of het model gevonden bij het Middelgat nog aangepast dient te worden voor Vlissingen.	2 wkn
- Schatten van de parameters van het model bij Vlissingen.	3 wkn
- Beschikbaar maken van de gegevens van de 13-uurs-metingen van de 9 lokaties van de periode 1970-1982 voor de programmatuur.	4 wkn
- Bekijken of de drie bestaande modellen voldoende zijn voor de deze 9 lokaties of dat er nog enige aanpassing nodig is. En het uitvoeren van de schattingen van de modelparameters voor deze eventueel aangepaste modellen.	6 wkn

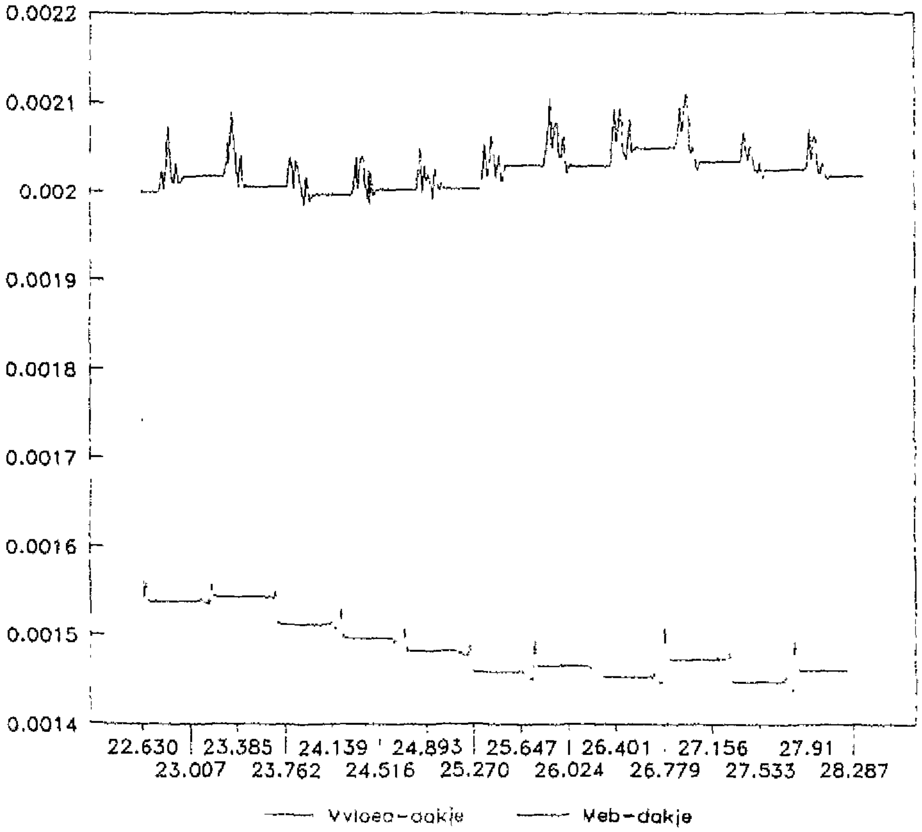
-	Verkrijgen van de gegevens van de 2-maandelijkse metingen over 1970-1990 in juiste vorm ter invoering in de bestaande modellen. En het berekenen van de benodigde stroomsnelheden.	uitbest.+ 3 wkn
-	Bepalen van het horizontale getij over de gehele periode 1970-1990 voor 9 lokaties	2 wkn
-	Elimineren van de door het model bepaalde getij-invloed op de reeks metingen van 1970-1990.	3 wkn
-	Bepalen van de seizoensinvloed.	2 wkn
-	Elimineren van de seizoensinvloed op de overgebleven reeks metingen voor alle negen lokaties.	6 wkn
-	Bekijken wat de trend is over deze 20 jaar op ieder van de 9 lokaties en bekijken wat het verloop van de slibconcentratie is in de Westerschelde, door de verschillende lokaties met elkaar te vergelijken over deze 20 jaar.	3 wkn
-	Afronding en verslaggeving van het onderzoek	6 wkn
-	Onvoorzien (10% van totaal)	5 wkn
-	TOTAAL	54 wkn



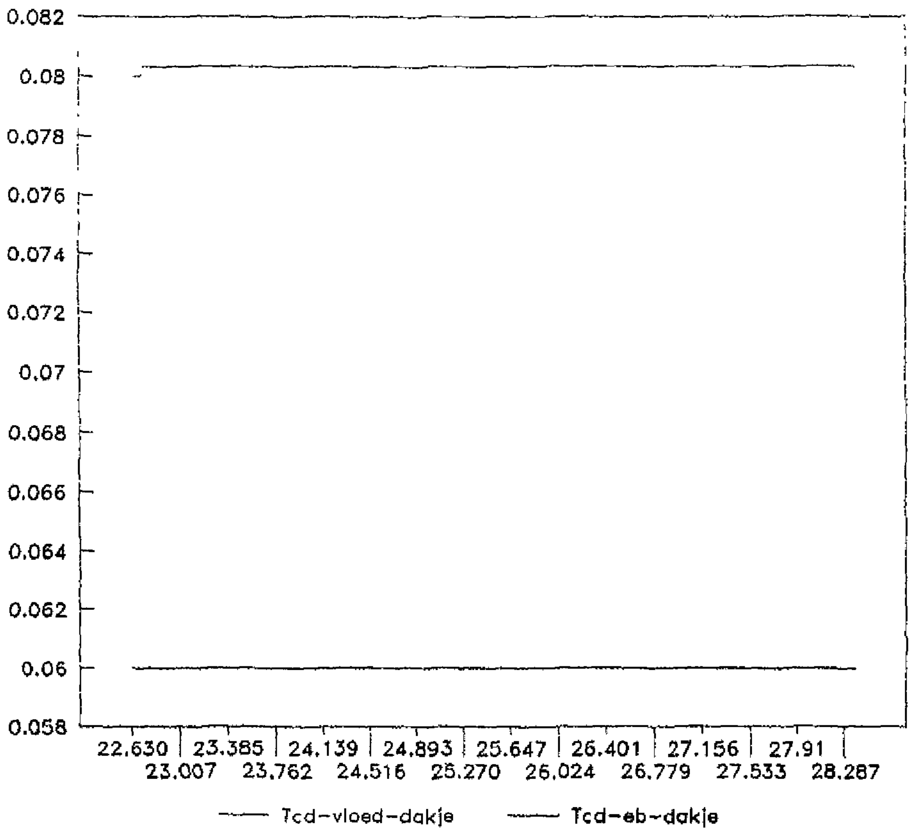
figuur 1: De dieptegemiddelde slibconcentratie van 22 nov.'88 t/m 28 nov.'88



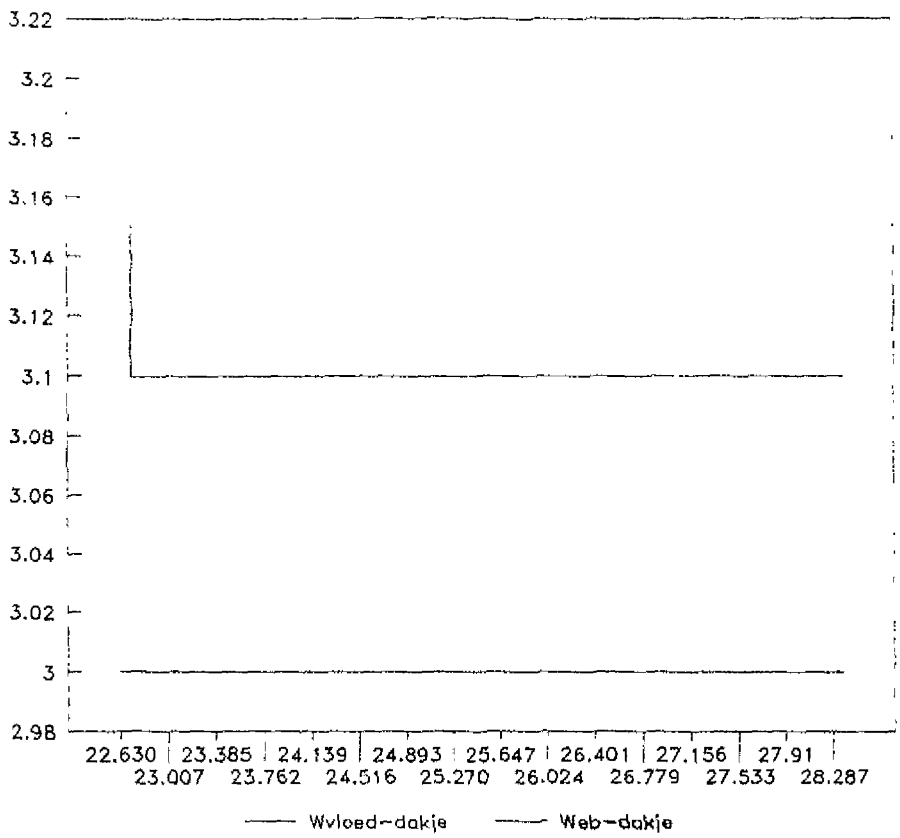
figuur 2: Schatting van τ_{ce} bij eb en bij vloed



figuur 3: Schatting van M bij eb en bij vloed



figuur 4: Schatting van τ_{cd} bij eb en bij vloed



figuur 5: Schatting van W_s bij eb en bij vloed