MODELLING AND CONTROL

OF POTABLE WATER CHLORINATION

Amélie Pastre

Submitted in fulfilment of the requirements for the degree of Master of Science in Engineering in the School of Chemical Engineering, University of Natal, Durban

Durban

September 2003

ABSTRACT

In potable water preparation, chlorination is the last step before the potable water enters the distribution network. Umgeni Water Wiggins Waterworks feeds the Southern areas of Durban. A reservoir at this facility holds treated water before it enters the distribution network. To ensure an adequate disinfection potential within the network, the free chlorine concentration in the water leaving the reservoir at the Umgeni Water Wiggins Waterworks should be between 0.8 and 1.2 mg/L. The aim of this study was to develop an effective strategy to predict and control the chlorine concentration at the exit of the reservoir. This control problem is made difficult by the wide variations in flow and level in the reservoirs, together with reactive decay of the chlorine concentration.

A Computational Fluid Dynamic study was undertaken to gain understanding of the physical processes operating in the reservoir (FLUENT software). As this kind of modelling is not yet applicable for real-time control, compartment models have been created to simulate the behaviour of the reservoir as closely as possible, using the results of the fluid dynamic simulation.

These compartment models were initially used in an extended Kalman filter (MATLAB software). In a first step, they were used to estimate the kinetic factor for chlorine consumption and in a second step, they predicted the chlorine concentration at the outlet of the reservoir. The comparison between predictions and data, allowed the validation of the compartment models.

A predictive control strategy was developed using a Dynamic Matrix Controller, and tested off-line on the compartment models. The controller manipulated the chlorine concentration in the inlet of the reservoir in order to control the chlorine concentration in the outlet of the reservoir.

Finally, the simplest compartment model was implemented on-line, using the Adroit SCADA system of the plant, in the form of a Kalman filter to estimate the chlorine decay constant, as well as a predictive model, using this continuously-updated decay parameter. The adaptive Dynamic Matrix Controller using this model was able to control the outlet chlorine concentration quite acceptably, and further improvements of the control performance are expected from ongoing tuning.

PREFACE

Umgeni Water initiated this project to be able to control the chlorine concentration at the exit of the chlorine contact reservoir at Wiggins waterworks by manipulating the chlorine concentration in the inlet of the chlorine contact reservoir.

The investigation required data collected from Umgeni Water Wiggins Waterworks, in Durban. Other studies were made in the postgraduate offices of the School of Chemical Engineering at the University of Natal, Durban under the supervision of Professor Michael Mulholland and the co-supervision of Chris Brouckaert, Professor Chris Buckley and Professor Marie Véronique Le Lann.

The following courses were completed with the corresponding credits and results achieved:

DNC4DC1	Process Dynamics and Control	65% (16 credits)
DNC5RT1	Real Time Process Data Analysis	61% (16 credits)

I hereby declare that this dissertation is my own work, unless stated to the contrary in the text, and that it has not been submitted for a degree to any other university or institution.

Signed: .....

Date: .....

As the candidate's supervisor, I have / ~~have not~~ approved this dissertation for submission

Signed: ..... Name: ..... Date: .....

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to all of those people who have directly or indirectly contributed to this work by giving their precious time and advice.

A special word of thanks is due to:

My supervisor Professor Michael Mulholland for his many contributions to this investigation, and for his encouragement and final proofreading of this dissertation. He has helped me so many times in my work and also in my stay in South Africa.

My co-supervisors Chris Brouckaert, Chris Buckley and Marie Véronique Le Lann. Their assistance has extended my understanding of this subject and is much appreciated.

The University of Natal and the Ecole Nationale Supérieure d'Ingénieurs de Génie Chimique (France) for enabling me to pursue my studies here at the University of Natal.

Professor Marie-Véronique Le Lann (Laboratoire d'Analyse et d'Architecture des Systèmes and Institut National des Sciences Appliquées - INSA in Toulouse). She has done everything to make my stay here possible. Professor Alain Liné (Laboratoire d'Ingénierie des Procédés de l'Environnement, INSA, Toulouse) for his assistance with the Computational Fluid Dynamic.

Umgeni Water Ltd, for having proposed this project. Dan Naidoo, Martin Pryor, Rachi Rajagopaul and Sagren Govendor of Umgeni Water for their encouragement, support and technical input. Also to all those from the Umgeni Water Wiggins Waterworks for their assistance in data collection.

My family, my colleagues and friends from the School of Chemical Engineering and from all around the world for their understanding, support and encouragement.

The financial support of this research from the South African National Research Foundation, the French Ministère des Affaires Etrangères, the University of Natal's Research Fund and the Water Research Commission of South Africa is gratefully acknowledged.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Alphabet

A	Matrix of model coefficients	
B	Dynamic matrix	
B_{OL}	Open loop matrix	
B_0	Offset matrix	
C	Chlorine concentration	mg/L
C_t	Observation matrix from DAEs system	
CFD	Computational Fluid Dynamic	
D	Turbulent diffusivity	m^2/s
DAE	Differential Algebraic Equation	
DMC	Dynamic Matrix Control	
EKF	Extended Kalman Filter	
F	Flow rate	ML/d
h	Water height	m
K	Kalman filter gain	
k	Chlorine decay factor	d^{-1}
M	Steady state horizon (number of steps)	
M_t	Filter covariance matrix	
MPC	Model Predictive Control	
N	Number of future control moves	
ODE	Ordinary Differential Equation	
P	Optimisation time horizon	
Q	Predictive error covariance matrix	
q	Dye quantity required	g
R	Observation error covariance matrix	
RTD	Residence Time Distribution	
t	Time	d
u	Velocity in x direction	m/s
u_t	Input (or manipulated) variable	
v	Velocity in y direction	m/s
V	Volume of the reservoir	ML
w_t	Process variable weighting matrix	
$\hat{w}$	Observation vector form DAEs system	

y	State vector	
z	Associated variable	

Subscript

$_0$	Inlet	
$_R$	Exit	

Greek alphabet

α	Fraction of the total inflow entering the reservoirs	(-)
β	Flow coefficient	$MLd^{-1}m^{-1/2}$
μ_t	Fluid turbulent viscosity	kg/m.s
ρ	Fluid density	kg/m^3
σ	Turbulent Schmidt number	(-)
τ	Nominal retention time	d

CHAPTER 1

INTRODUCTION

This chapter presents an overview of the Wiggins Waterworks. It introduces the problem of controlling the chlorine concentration at the exit of the works in light of a variable demand. The chapter concludes by outlining the objectives of the dissertation.

1.1 THE WATER TREATMENT PLANT

Umgeni Water is the largest water authority in the Kwa-Zulu Natal Province of South Africa. The area of supply covers some 24 000 square kilometres with the main boundaries being the Indian Ocean in the East, the Tugela and Mooi Rivers in the North, the Drakensberg Mountains in the West and the Mkomazi and Mzimkulu Rivers in the South (Figure 1-1).

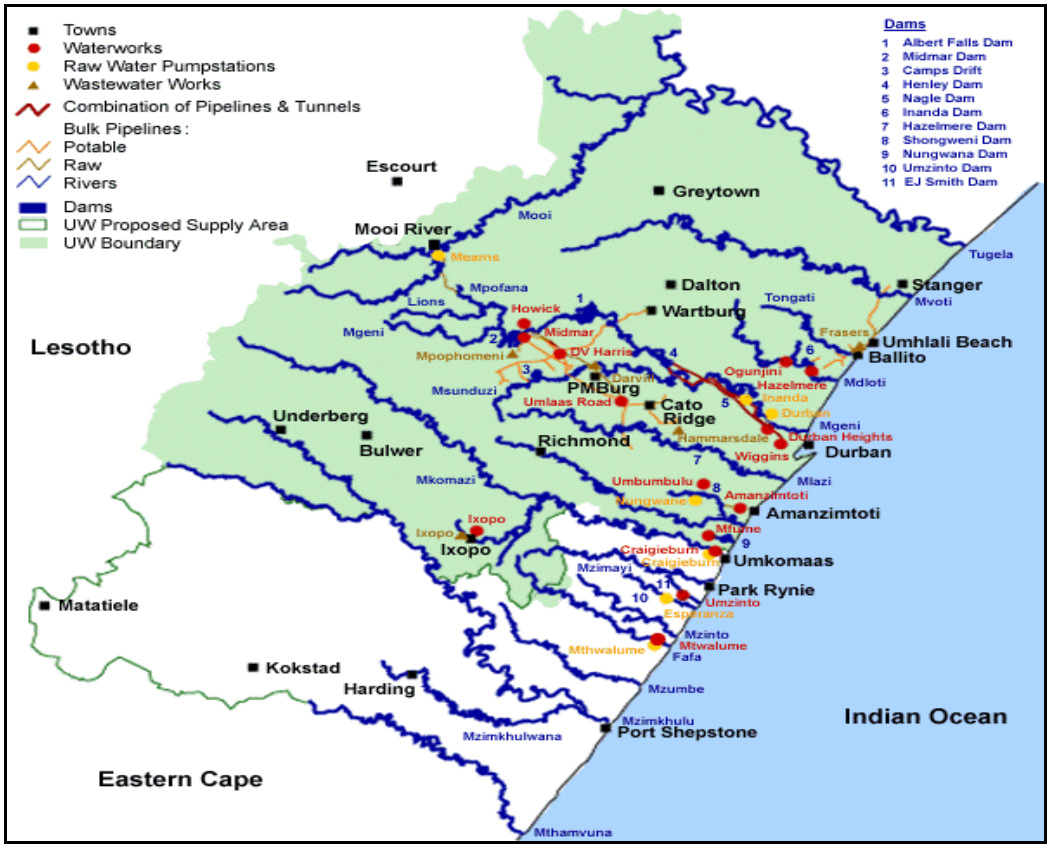


Figure 1-1: Umgeni Water operational area (Umgeni Water (2002))

Wiggins Waterworks is situated in the Cato Manor district of Durban. The design of the Wiggins Waterworks commenced in September 1980 with a commitment to increase the supply of potable water to the Durban area by the summer of 1984. Wiggins Waterworks (Figure 1-2) is designed to treat water from the Mgeni River intake, and also water supplied from Inanda Dam. A system of five tunnels and pipelines transfers raw water by gravity. Inanda Dam is the last dam situated on the Umgeni River.

Figure 1-2: Wiggins Waterworks aerial view (Umgeni Water (2002))

There are several purification steps taken at Wiggins Waterworks to make water clean and safe for domestic and industrial consumption. The process diagram of the plant is shown in Figure 1-3. Potable water flows by gravity from its 124 ML storage reservoir to southern Durban and adjacent areas. Presently the design capacity of the waterworks is 350 ML/d of raw water. The final water has a pH value ranging between 7.8 and 8.2 and a turbidity value below 0.8 NTU.

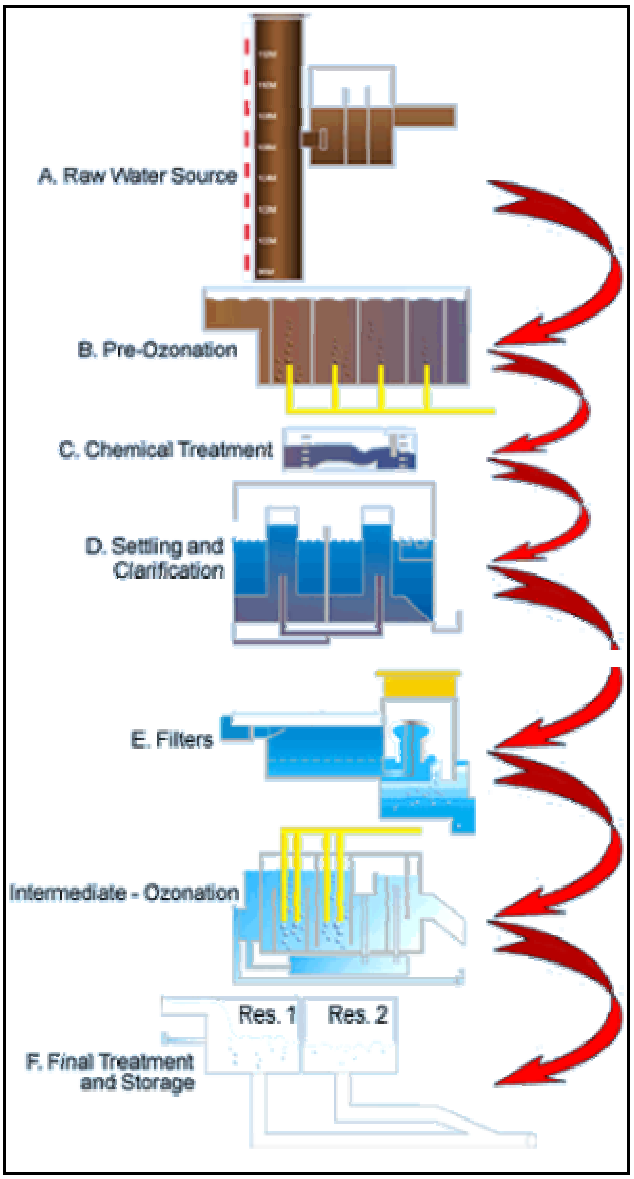


Figure 1-3: Wiggins Waterworks process diagram (Umgeni Water (2002))

1.2 CONTROL PROBLEM

At the end of the treatment plant, a chlorine contact reservoir, divided into two sections, holds treated water (Step F, Figure 1-3). Because of the variable demand, the reservoir level and residence time vary over a range of values. The maximum combined capacity of the two reservoir sections is 120 ML (7.05 m level), and the minimum capacity is 25.5 ML (1.5 m level). The daily average demand is 140 ML/d, with a daily peak about 220 ML/d. However, this can occasionally be as high as 320 ML/d, making the control of the chlorine dosing difficult. The chlorine residual concentration declines with time, so that the outlet chlorine

concentration will depend on the residence time in the reservoir. The chlorine concentration of final water should be kept between 0.8 and 1.2 mg/L: above 0.8 mg/L to ensure it maintains water disinfection within the distribution network and below 1.2 mg/L to prevent taste and odour complaints. Prior to this investigation, the chlorine dosage in the inlet of the reservoir has been manipulated manually based on the experience of the operators.

The aim of this project is to develop an effective real-time control strategy to predict and control the chlorine concentration at the exit of the plant, and hence allow optimal targeting of the chlorine concentration in water delivered into the Durban Metro region.

First, a Computational Fluid Dynamic (CFD) simulation was undertaken to gain an understanding of the physical processes in the reservoir. However, this kind of modelling is too computationally intensive to be applicable for real-time control. Therefore, three compartment models were created to estimate the kinetic decay factor of the chlorine and to control the outlet chlorine concentration. These models were solved within an extended Kalman filter. Finally, Dynamic Matrix Control algorithm was chosen to control the outlet chlorine concentration.

1.3 DISSERTATION LAYOUT

In **Chapter 2**, the background of chlorine contact reservoir technologies and process control are reviewed. **Chapter 3** presents the chlorine contact reservoir and the data collected from the plant. It considers also the viability of using a tracer test to predict the residence time distribution of the chlorine contact reservoir. The Computational Fluid Dynamic simulation is described in **Chapter 4**. Then, in **Chapter 5**, the different compartment models to predict the outlet chlorine concentration are created and solved using an extended Kalman filter. **Chapter 6** deals with the arrangements and properties of the controller, and how the controller was built. **Chapter 7** explains on-line implementation of the model predictive controller. **Chapter 8** presents the conclusions and recommendations derived from this study.

CHAPTER 2

LITERATURE REVIEW

This chapter reviews the kinetics of chlorine disinfection, the modelling of equipment hydraulics and tracer concentrations , and it introduces the concept of process control.

2.1 CHLORINE CONTACT RESERVOIR

As Faust et al. (1999) explained the disinfection can be traced back to about 2000 BC. Disinfection is defined by Desjardins (1975) as the destruction or the elimination of microorganisms liable to pass disease on to people. It was only during the 17th century that scientists could explain why certain types of water caused illness. In the 1860s and 1870s, large treatment and distribution facilities were developed to deliver potable water to the increasing urban population. Robert Koch discovered in 1881 that chlorine could deactivate water borne bacteria (American Water Works Association, 1999). Since then, the process has been improved and several steps have been added to obtain the best possible water quality. A modern potable water treatment plant may be represented schematically as shown in Figure 2-1.

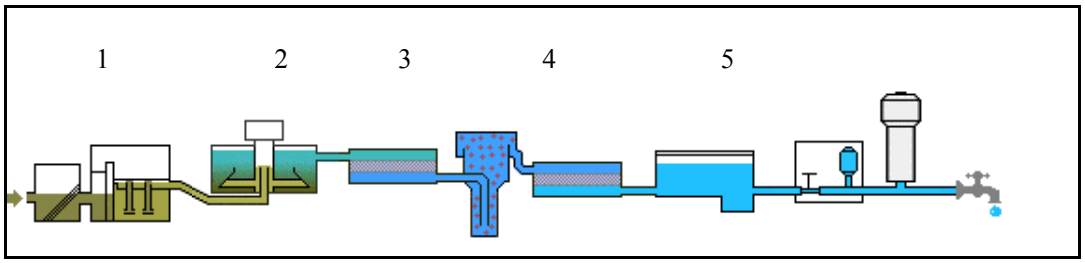


Figure 2-1: Simplified scheme of a potable water treatment plant (Lyonnaise des eaux (2002))

1. Pre-treatment

Water is taken from dams or rivers where it passes through wire screens at the intake points to remove any large solid objects.

2. Mixing, Coagulation and Flocculation

Process chemicals are mixed with the water. These alter the surface charge on colloidal solids promoting coagulation of suspended dirt particles. The particles become large enough to sink to the bottom of the reservoir in a reasonable time.

3. Sedimentation and Filtration

The clear water at the top of the clarifier is skimmed off and passed through filters filled with sand and gravel to remove suspended matter.

4. Disinfection

Finally chlorine is added to kill any remaining microbes. Samples of the treated water are tested to make sure the water is safe for drinking.

5. Distribution

The clean drinking water is normally stored in a large reservoir, whence it is distributed.

The World Health Organisation (1984) guidelines state that, to achieve virus free water, chlorinated water should receive 30 min contact time with a minimum free chlorine residual of 0.5 mg/L. On the other hand, the customer will complain about the strong taste of the chlorine if the concentration is above 1.2 mg/L. Therefore managers of drinking water supplies are concerned about the control of the free residual chlorine concentration in the water which leave the plant.

One of the process approaches that evolved since the introduction of chlorination is the chlorine contact reservoir. As van der Walt (2002) explained, this reservoir is at the end of the treatment plant and aims to achieve sufficient contact time between the dissolved chlorine and the water.

According to Laubush (1955) and Desjardins (1975), the products used most commonly to obtain disinfection by chlorine are chlorine gas (Cl_2), sodium hypochlorite (NaOCl), calcium hypo-chlorite ($Ca(OCl)_2$), mono-chloramines (NH_2Cl) and chlorine dioxide (ClO_2). Chlorine gas is the most widely used, however sodium hypochlorite is sometimes generated on-site because it is easy to manipulate and safe for the operator.

These two authors emphasise the definition of the different expressions:

- *Combined residual chlorine* is that residual chlorine existing in the water in chemical combination with ammonia or organic nitrogen compounds.
- *Free residual chlorine* is that residual chlorine existing in the water as hypochlorous acid (HOCl) or hypochlorite ion (OCl^-).
- *Total residual chlorine* is the sum of the free residual chlorine and the combined residual chlorine.

However, the term *chlorine* is generally used for *free residual chlorine concentration*. Indeed it defines the chlorine which is available to disinfect the water, and on-line instruments are available to measure the free residual chlorine concentration. Thus, in the following work, free residual chlorine will be termed as chlorine.

2.2 CHLORINE CONTACT RESERVOIR KINETICS

The chlorine concentration decreases with time because it is consumed during disinfection and can be lost at free surfaces. The chlorine decay in tanks has not been investigated to the same extent as the investigation of chlorine decay in pipelines. Viljoen et al. (1997) studied the chlorine decay in pipelines for free chlorine and for monochloramines:

- the general formulation for an n_{th} order decay reaction rate is:

$$r = -kC^n(t) \tag{2.1}$$

where C = chlorine concentration (mg/L)
k = chlorine kinetic factor ($(mg/L)^{1-n}$ /d)
t = time (d)
n = order of the reaction (-)

- free chlorine decay showed a large variation in decay rate (0.96 to 1.2 per day) and in reaction order (0.36 to 1.22)
- a first order decay formulation proved a good compromise between accuracy and simplicity. The formulation is given by:

$$r = -kC(t) \tag{2.2}$$

Hua et al. (1998), Powell et al. (1999), and van der Walt (2002) reach the same conclusion and found that the chlorine kinetic factor ranged between 0.01 per day and 6 per day. These values have to be used with caution, because they are for the chlorine decay in pipelines. In reservoirs, water volumes are more important so short-circuiting and re-circulation can be expected, changing the chlorine decay. Moreover, these values include the pipe wall reaction, which does not take place in reservoirs.

2.3 CHLORINE CONTACT RESERVOIR HYDRAULICS

The theoretical contact reservoir could be represented by a plug flow reactor. As Levenspiel (1999) explained, a plug flow reactor is characterized by the fact that the flow of fluid through the reactor is orderly with no element of fluid overtaking or mixing with any other element

ahead or behind. The necessary and sufficient condition for a plug flow is for the residence time (τ) in the reactor to be the same for all fluid elements.

$$\tau = \frac{V}{F} \tag{2.3}$$

where τ = nominal retention time (d)

V = contact reservoir volume (ML)

F = flow rate (ML/d)

However, these hydraulic conditions of plug flow are seldom achieved in practice. Diffusion and dispersion always exist, and turbulent regions (often including flow re-circulation and separation) cause the flow to be unpredictable. The result is a residence time distribution (RTD).

2.4 COMBINATION OF HYDRAULICS AND KINETICS

A plug flow reactor can be approximated by an infinite number of mixed flow reactors (Levenspiel, 1999). In the mixed reactor, the contents are well stirred and uniform throughout. Thus, the exit stream from the reactor has the same composition as the fluid within the reactor. Contact reservoirs can be modelled by a number of well-mixed flow elements organised in a certain way to simulate re-circulation or dead-volume problems (Figure 2-2)

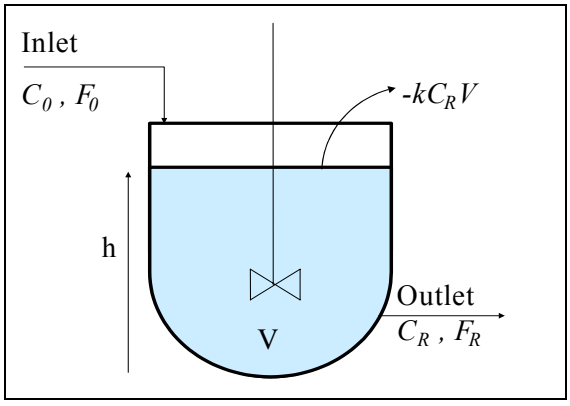


Figure 2-2: Well-mixed reactor scheme

Where C_0 = Inlet chlorine concentration (mg/L)

C_R = Outlet chlorine concentration (mg/L)

F_0 = Inlet flow rate (ML/d)

F_R = Outlet flow rate (ML/d)

V = Volume of the water (ML)

h = Height of the water (m)

k = chlorine decay factor (/d)

These unsteady balances for a well mixed reactor (Figure 2-2) can be written:

Chlorine: $$\frac{d(C_R V)}{dt} = F_0 C_0 - F_R C_R - k C_R V \quad (2.4)$$

Volume: $$\frac{dV}{dt} = F_R - F_0 \quad (2.5)$$

for the case as here where the reacting species is in dilute solution.

2.5 TRACER TEST

Tracer tests are often used to determine the hydrodynamic behaviour of the contact reservoir as well as the residence time distribution. A pulse or step in tracer concentration at the inlet allows determination of the residence time distribution. The tracer species should be conservative, easily measurable and, of course, safe for human consumption (Ducluzaux, 1999).

Several different techniques of tracing are used:

- the measurement of a tracer salt by electric conductivity since the beginning of the century (Ducluzaux, 1999).
- the measurement of radioactive tracers by the radioactivity (Molinari, 1976).
- the measurement of fluorescent tracers by *in situ* fluorometers (Molinari, 1976).
- the measurement of chemical tracers by *in situ* chemical sensors (Ducluzaux, 1999).

A fluorescent dye is rhodamine WT but it is not very stable in chlorinated water. Phloxine B is reported to be the most stable fluorescent dye. It is certified by the National Sanitation Foundation International under the condition that the concentration of the phloxine B liquid in drinking water does not exceed 0.1 μg/L (NSF International, 2000). The readings can be made directly on a continuous-flow or an individual sample without processing. An advantage is that a fluorometer can detect tracer concentration as low as 0.1 ng/L (Keystone Company, 2000). On the other hand, Ducluzaux described the advantage of the different ions (iodide, lithium) as a natural tracer, already present in the water, easy to measure with a chemical sensor (which has a precision of 0.01 μg/L), and not very expensive.

Levenspiel (1979) explained that by comparing the residence time distribution curve for the real vessel with the curves obtained with different combinations of theoretical well mixed and plug flow reactors, it is possible to simulate the residence time distribution with a compartment model. Figures 2-3 and 2-4 illustrate a residence time distribution curve by an impulse injection for a plug flow vessel and a well-mixed vessel respectively.

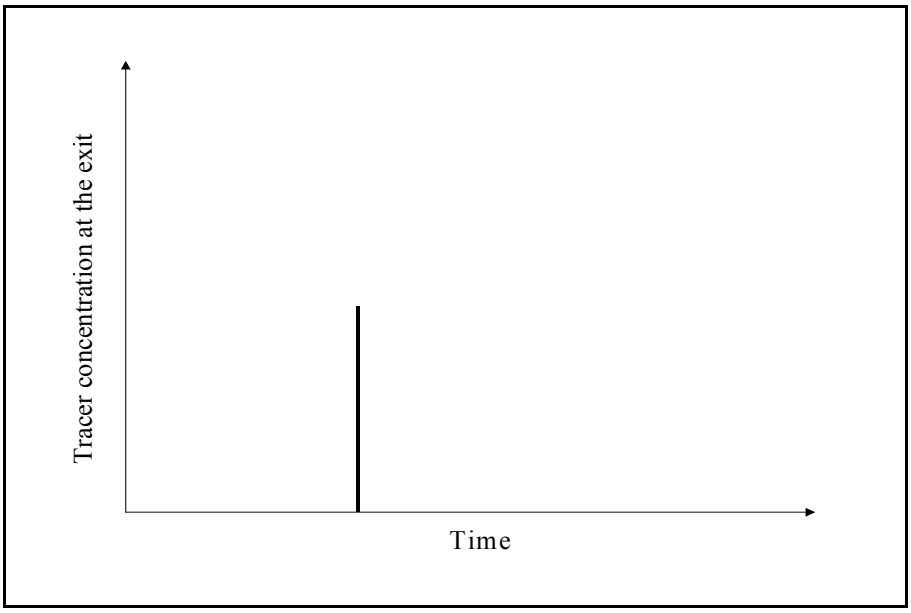


Figure 2-3: Residence time distribution curve for a plug flow vessel by an impulse injection

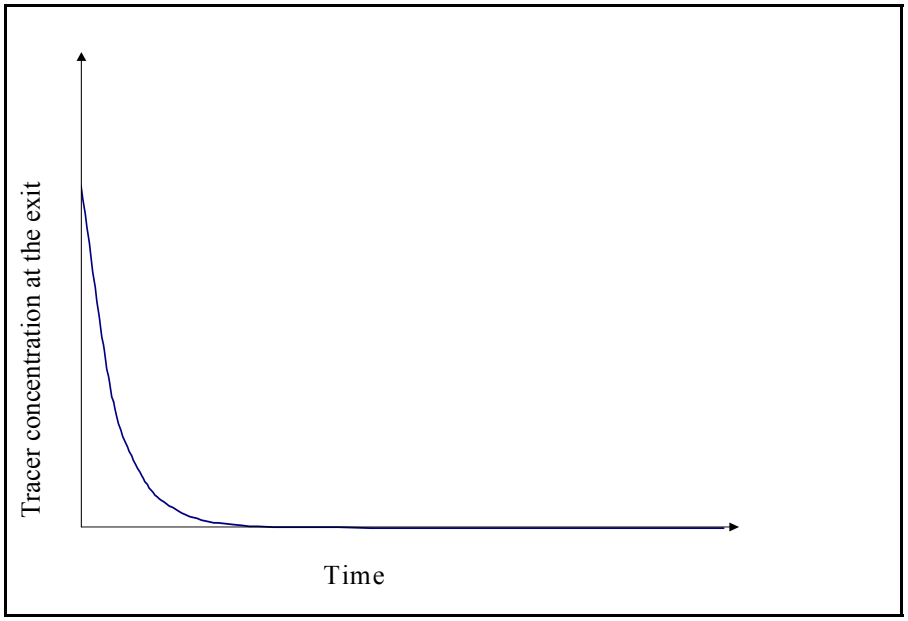


Figure 2-4: Residence time distribution curve for a well-mixed vessel by an impulse injection

2.6 COMPUTATIONAL FLUID DYNAMIC SIMULATION

As van der Watt (2002) explained, some experiences have been made to simplify the representation of contact reservoirs, in order to achieve the desired outlet chlorine concentration. One method was to consider the chlorine concentration and the theoretical

residence time. Unfortunately the theoretical residence time distribution curve does not always give a deep understanding of the hydraulic behaviour of the contact reservoir.

In the Computational Fluid Dynamics (CFD), the general conservation laws (conservation of mass and momentum) are applied to specific vessel geometries. The solution of a CFD analysis is uniquely characterised by the boundary conditions, overcoming the limit of the conventional approach explained in the previous paragraph.

Leclerc et al. (1998) and Chataigner et al. (1999) compared CFD results with experimental residence time distribution curves. This comparison showed that CFD results agreed remarkably well with the conventional turbulence models, with errors in the residence time distribution curves ranging from 6.7 to 9.3%. These small errors proves than CFD can be used not only for qualitative, but also quantitative predictions.

Van der Walt's study (2002) used the FLO++ code (Le Grange, 1998) to simulate the turbulent flow patterns. The Navier-Stokes equations and the k-ε turbulence model were used to simulate turbulent flow. Chlorine transport was modelled by a convection-diffusion formulation with a sink term representing chlorine decay. The scalar transport equation in two dimensions is given by:

$$\frac{\partial}{\partial t}(\rho C)+\frac{\partial}{\partial x}(\rho uC)+\frac{\partial}{\partial y}(\rho vC)=\frac{\partial}{\partial x}[(D+\frac{\mu_t}{\sigma_{CX}})(\frac{\partial C}{\partial x})]+\frac{\partial}{\partial y}[(D+\frac{\mu_t}{\sigma_{CY}})(\frac{\partial C}{\partial y})]-\rho kC \qquad (2.6)$$

Where ρ = fluid density (kg/m^3)

μ_t = fluid turbulent viscosity (kg/m.s)

σ = turbulent Schmidt number (-)

D = turbulent diffusivity (m^2/s)

u, v = velocities in x- and y-directions (m/s)

C = chlorine concentration (mg/kg)

k = rate of chlorine decay (/s)

The first term represents the unsteady chlorine concentration term, the second and third terms represent the convection of chlorine, the fourth and fifth terms represent the diffusion of chlorine and the last term represents the first order chlorine decay sink term. In his study, van der Walt concluded that the CFD is able to model the hydraulics and the chlorine decay accurately.

However this kind of modelling is too detailed and computationally intensive to be applicable for real-time control, so a simplified control model will be required. The CFD is also not suitable for on-line application.

2.7 CONTROL OF WATER PROCESSES

As Johnson et al. (1997) introduced, the most widely practised chlorination control is to inject an overdose of chlorine at the inlet to the contact reservoir and adjust to the desired free residual chlorine level in the effluent stream, by addition of sulphur dioxide or sodium bisulphite. However this method of control may not be optimal. They described reliable predictions of retention time distributions and the use of programmable logic controllers (which are not detailed), coupled with an understanding of chlorine disinfection kinetics, and so offered a potential for more efficient chlorine dosing.

Other authors, Sérodes et al. (2001), created a methodology for developing decision-making tools for chlorine disinfection control: Chlorocast©. This methodology is based on the creation of a database for typical situations and the use of an artificial neural network.

These methods work with a well known fixed kinetic factor and residence time distribution curve. However this kinetic factor depends on the temperature and the quality of the water, as well as the control input, the initial concentration of free residual chlorine in the flow, which enters the chlorine contact reservoir (Powell et al., 1999).

The problem is that, during a year, this kinetic factor has to be recalculated, depending on the situation (dry season, rainy season, winter, summer). Moreover depending on the plant, it is not always possible to obtain a residence time distribution curve.

2.8 EXTENDED KALMAN FILTER AND PARAMETER IDENTIFICATION

The equations for the contact reservoir are multivariable and non-linear with a mixture of differential and algebraic equations (DAE), with state variable outputs (chlorine concentration), inputs (flow, level), associated variables (consumer demand) and physical parameters (chlorine decay factor). Algorithms that are suitable for real-time usage and based on successive updating of the model parameters are generally *recursive*. There is a large number of recursive identification algorithms described in the literature. Ogunnaike et al. (1995), Åström et al. (1995) and Ljung (1999) overview these techniques.

A popular technique used to solve the recursive problem is the Kalman filter. This technique, attributable to Kalman et al. (1960) solves the recursive estimation problem. It has been described by Catlin (1980) and Balakrishnan (1984).

Chui et al (1987) reported that the Kalman filter has been designed to estimate the state vector in a linear model. If the model is non linear, a linearisation procedure can be performed in deriving the filtering equations; the Kalman filter obtained in this manner is called the *extended Kalman filter* (EKF). The EKF has found many important real-time applications, one of which is the adaptive parameter identification. The EKF thus can be used to identify, in real time, the chlorine decay factor by fitting the model outputs to observed chlorine levels.

2.9 CONTROL

2.9.1 Adaptive Control

As defined by Isermann (1982), *adaptive control systems* adapt their behaviour to the changing properties of controlled processes and their signals. Many different proposals for adaptive control have been made in the past. However, their application has not been very successful, and somewhat unconvincing until the early 1970s. The development of cheaper and more reliable digital computers has meant that the field of adaptive control has been reactivated. Adaptive control algorithms have received much attention in recent years because good results have been given by some applications. Le Lann et al (1995) studied several adaptive control algorithms (adaptive PID, Self Tuning Controller, Model Reference Controller and Generalised Predictive Controller) on different types of extraction pilot plant. They obtained their best results with the Generalised Predictive Controller.

2.9.2 Model Predictive Control

The term Model Predictive Control (MPC) describes a class of computer control algorithms, which are used to find optimal control action settings by predicting their impact on the future output of the system (Garcìa et al., 1989). MPC technology was originally developed for power plant and petroleum refinery applications, but is now applied in a wide variety of manufacturing environments including chemical, food processing, automotive, aerospace, metallurgy, and pulp and paper. The reason for its popularity can be attributed to three important factors:

- Incorporation of an *explicit* process model into the control calculation. This allows the controller to deal directly with all significant features of the process dynamics.
- The plant behaviour is considered over a period which extents to a future time horizon. This means that the effects of disturbances can be anticipated and removed, allowing the controller to drive the plant more closely along a desired future trajectory

- The process input, state and output constraints can be directly considered in the control calculation, so constraint violations are less likely.

Hence MPC designs have the ability to yield high performance control systems capable of operating without expert intervention for long periods of time.

Clarke et al. (1987) and Qin et al. (1997) described MPC algorithms. The future moves of the manipulated variables are determined by minimizing the predicted deviation from the set-point subject to operating within constraints. This optimisation is repeated at each sampling time based on updated information (measurements) from the plant.

2.9.3 Dynamic matrix control

Garcìa et al. (1989) reviewed different techniques emanating from MPC: Dynamic Matrix Control, Model Algorithmic Control, Inferential Control and Internal Model Control. The DMC algorithm is currently one of the most popular and widely used MPC algorithms, because it is simple, intuitive and allows a formulation of the prediction vector in a natural way. It is based on a linearised step response model called the *convolution* model to predict the effect of possible control actions. Such a strategy enables the model-based control to anticipate where the process is heading.

Successful applications of DMC have been reported in the literature. Cutler et al. (1980) described the DMC algorithm and reported application to a fluid cracker. An algorithm based on the DMC has been developed by Mulholland et al. (1997) following the methods of Chang et al. (1983), and Morshedi et al. (1985). This algorithm has been applied to control the top and the bottom temperature of a semi-industrial distillation column.

In a recent work, Guiamba (2001) examined modelling and control issues for a complex multivariable industrial operator training plant, and developed and applied an on-line method for adapting the controller to account for non-linearity.

CHAPTER 3

THE CHLORINE CONTACT RESERVOIR CONSIDERED IN THIS STUDY

This chapter presents the chlorine contact reservoir and the data collecting from the plant. It considers also the viability of using a tracer test to predict the residence time distribution of the chlorine contact reservoir.

3.1 THE CONTACT RESERVOIR

In the last stage of the water treatment process (Figure 1-3), the water is chlorinated with NaOCl and is stored in a reservoir, which is divided into two sections. This permits a sufficient contact time between the water and the chlorine for water disinfection. In addition it provides a holding-time to allow any water quality problem to be corrected before the water enters the distribution network. Figure 3-1 is a view on the top of the reservoir, showing the inspection hole and the instrumentation room.

Figure 3-1: View of the top of the reservoir

3.1.1 Flow through the reservoir

The flow is illustrated in Figure 3-2. From the treatment plant, water is received through a pipe where chlorine addition is continuous. Water is fed into two reservoir sections, but the ratio of the split is unknown. The levels are equal in both sections due to the interconnection of the

outlets. For operational reasons, the flow rate through the water treatment process preceding the reservoir is kept as steady as possible. At the exit of this reservoir, the treated water is transferred to the Durban Unicity buffer reservoirs according to the consumers' demand.

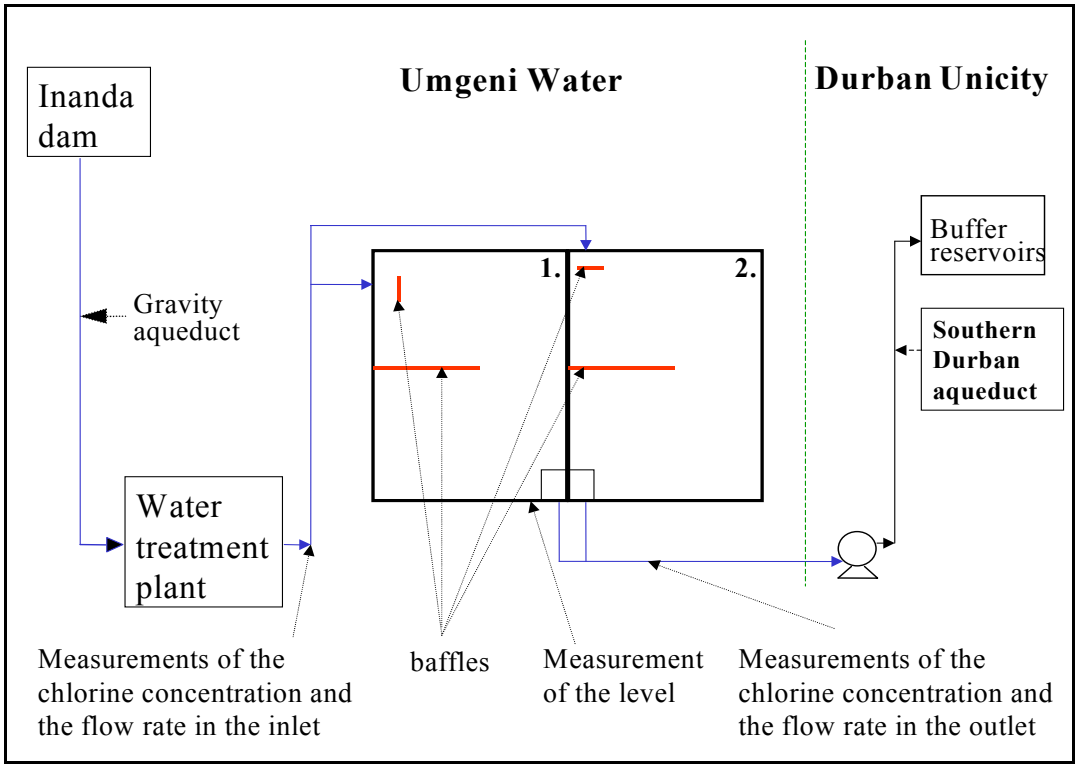


Figure 3-2: Plan view of the Wiggins Water treatment plant

The buffer reservoirs supply directly to consumers. When any of the buffer reservoirs reaches the minimum level, individual pumps and valves drawing from the Durban Unicity southern aqueduct are switched on. With many buffer reservoirs drawing water on demand, the water drawn from the Wiggins reservoir will follow the user demand profile (Figure 3-3).

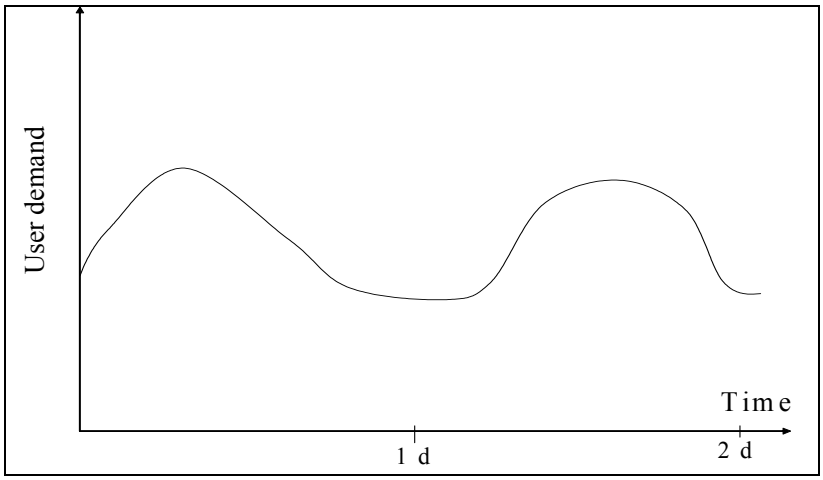


Figure 3-3: User demand profile

However, several large pumps at the Wiggins works are also switched on periodically to transfer water to Durban Heights waterworks, which serves a different distribution area due to its greater altitude. When these pumps switch on, there is a noticeable step impact on the Wiggins outflow, with consequent decrease in level (Figure 3-4).

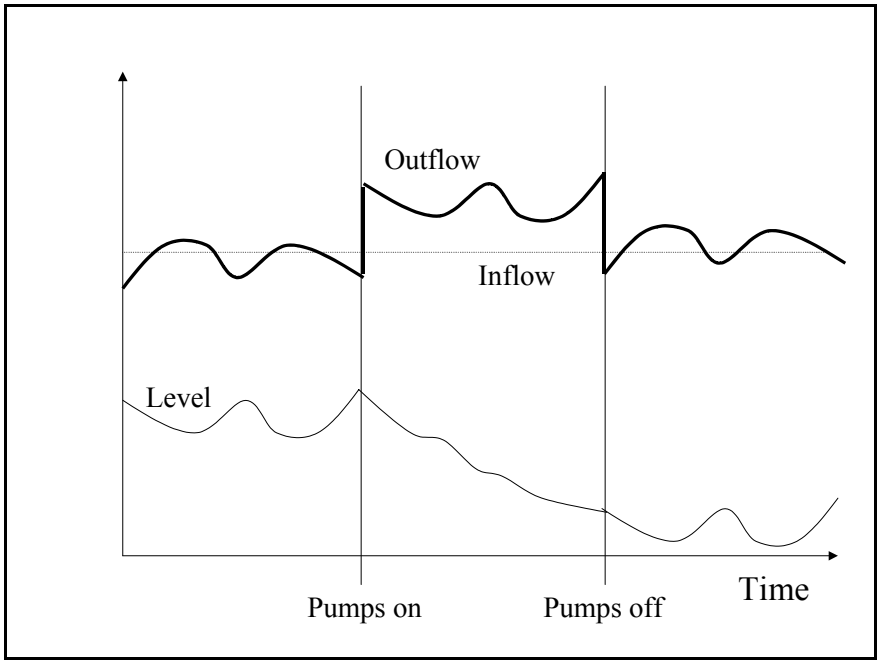


Figure 3-4: Illustration of the inflow, outflow and level in the reservoir

Depending on the difference between inflow and outflow, the water level inside the reservoir may rise or fall. The amount of chlorine required is dependent on the quantity of the water present in the reservoir, and the flow rate. The longer the residence time, the more chlorine required per unit of flow out of the reservoir.

3.1.2 Geometry of the reservoir sections

These reservoir sections are large: 116.5 m by 76 m. The wall height is 7 m but the baffles shown in Figure 3-2 are 2 m in height. The smaller baffle is 5 m in length, placed directly in front of the inlet; whereas the larger baffle is 48 m long situated in the middle of each section. The inlets of 1 m in diameter are positioned 1.5 m above the floor and the outlets of 3 m in diameter are from a trough in the floor.

3.1.3 Instrumentation

The inflow of the reservoir is measured by a crump weir and ultrasonic level device (Milltronic Multiranger Plus, accuracy: ± 0.25% of range, range is set to 900 mm). The total outflow of the reservoir is measured by magnetic flowmeters (Kent ABB, calibrated by positive displacement

tests on the storage reservoir, accuracy: ±0.5%). The inlet chlorine concentration to the reservoir is measured by an on-line chlorine analyser (Wallace & Tiernan, calibration against DPD method, accuracy: repeatable to 0.1 mg/L). The outlet chlorine concentration from the reservoir is also measured by an on-line chlorine analyser (Endress & Hauser, calibration against DPD method, accuracy: repeatable to 0.1 mg/L) (Figure 3-5). An ultrasonic level measurement device measures the level of the reservoir (Milltronic Multiranger Plus, calibrated against the survey detail of the reservoir and on empty distance, accuracy: ±0.25% of range, range used equal to 7.5 m).

Figure 3-5: On-line pH meter, Chlorometer at the exit of the reservoir

A 4 to 20 mA driver board drives these circuits and provides an input voltage for an A/D converter scanned by a PC SCADA (Supervisory Control and Data Acquisition) system (ADROIT). This system processes the data, executes control loops and stores data at regular intervals to assist with short-term operational decisions as well as long-term planning.

3.2 DATA COLLECTION

The Adroit SCADA system stores data at 5 min intervals. Each data record of interest (inlet and outlet chlorine concentration, inflow rate, outflow rate and level for the entire reservoir) can be saved. Each month (from August 2001 to April 2002), at least one week of data were collected from the plant, analysed using Microsoft Excel and then directly imported to a Matlab file.

Figure 3-6 presents a typical set of data:

- The inflow is maintained as steady as possible
- The outflow exhibits the step characteristic forms that were explained in the section 3-1
- The level rises when the valves shut and so the outflow rate becomes close to 0 ML/d, and drops off when the outflow is greater than the inflow.

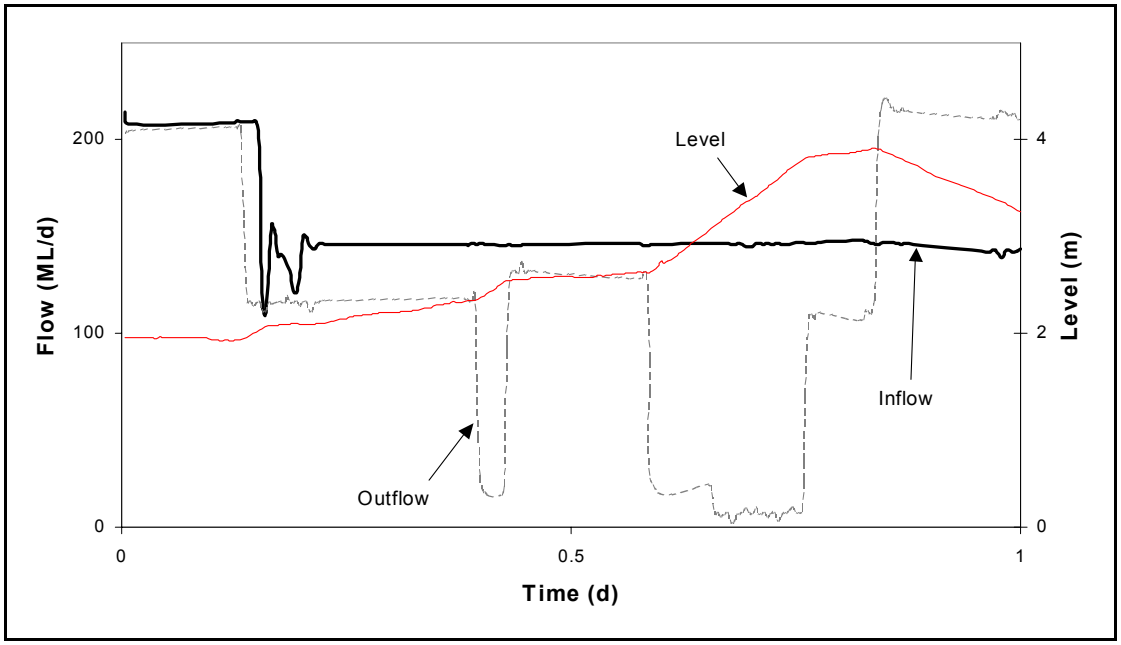


Figure 3-6: Flows and level (25/08/01)

Figure 3-7 shows the chlorine concentration and the level plotted for the same period of time.

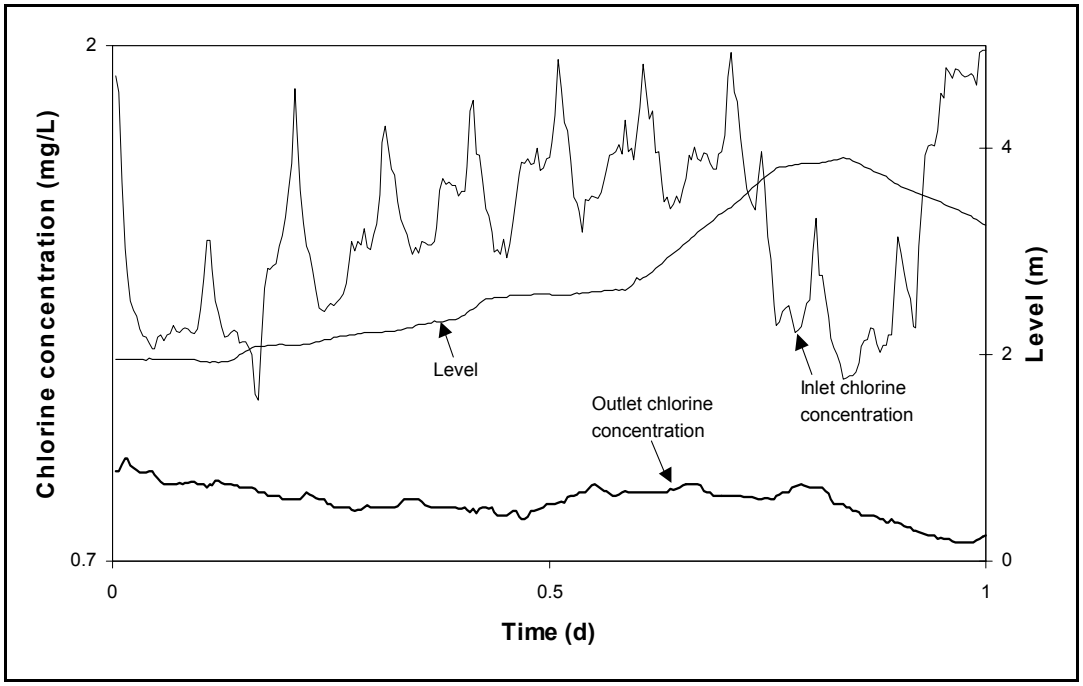


Figure 3-7: Chlorine concentration and level (25/08/01)

Some of the treated water is used to back wash the filters every 2 h. As the flow is measured at the head of works, this will not show on the flow meter. However the chlorine dosing does not account for this fluctuation in flow, therefore *spikes* occur in the reservoir inlet chlorine concentration as shown in Figure 3-7.

The main problem is that there is no apparent correlation between the inlet chlorine concentration variations and the outlet chlorine concentration variations. Indeed, with a fixed inlet flow rate and a certain range of level and inlet chlorine concentration, the outlet chlorine concentration shows variations, which seem unpredictable. Furthermore, the fact that available data are for the whole reservoir and not for each section (which do not have the same geometry) adds to difficulties in predicting the outlet chlorine concentration.

Considering that August is during the cold season in Durban, another set of data during January (hot season) is shown in Figures 3-8 and 3-9.

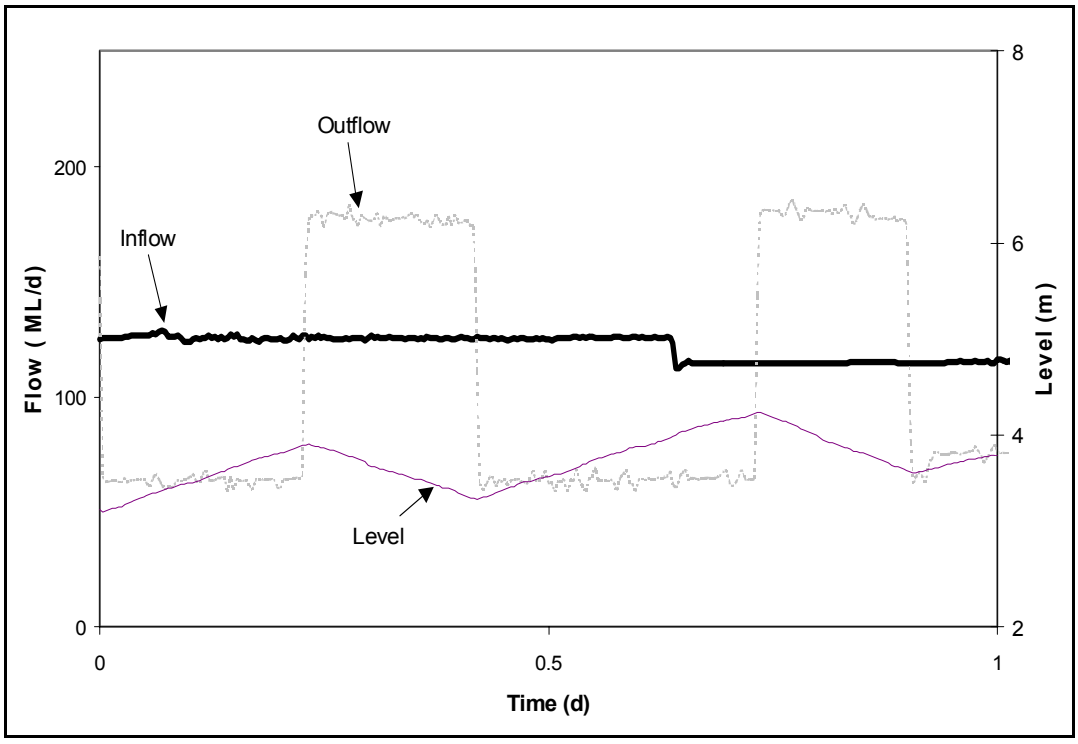


Figure 3-8: Flow rates and level (15/01/02)

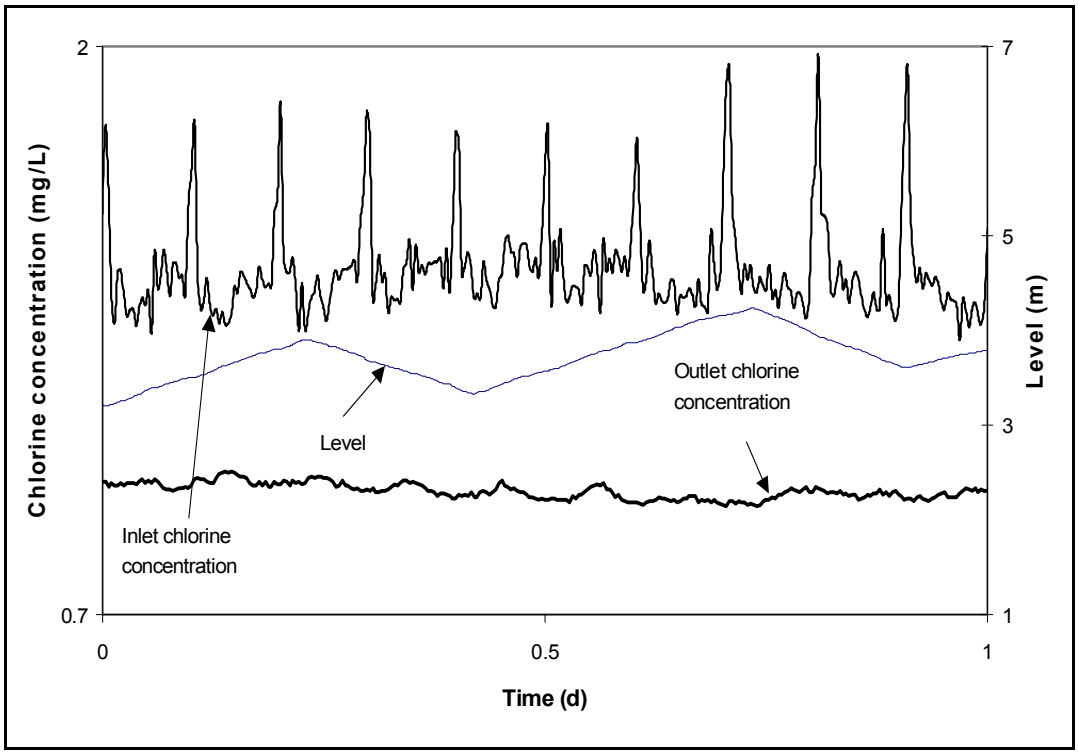


Figure 3-9: Chlorine concentration and level (15/01/02)

The data from the hot season, which is also the rainy season are much more stable. This can be explained by the fact that, as it is the rainy season, Inanda dam is full and a constant inflow can be taken from this dam. As it is the summer and the holiday season, people are using more water but always at the same period of the day so the outflow is subject to constant variations. On the contrary, in winter, people used less water and not regularly so the outflow is much more variable.

3.3 TRACER TEST

The reservoir sections are poorly designed from the point of view of chlorine concentration control: they are large with just two baffles 2 m in height, whereas the level can reach 7 m. Moreover, they do not have the same geometry and the measurement available for the inlet flow is before the division of the main inlet flow into two by-flows so the inflow for each one is unknown. To gain an understanding of the hydrodynamic behaviour of these two sections, tracer tests were considered.

Two methods are available as explained by Environment Quebec (2002)

- Constant rate injection
- Pulse injection

The most stable fluorescent dye with chlorinated potable water is phloxine B, but it is a very expensive dye. Given that the reservoir sections are very large, a large amount of tracer would have been required in carrying out a tracer test. In addition, with changes in the outflow being made frequently, thereby making it impossible to hold steady state for the necessary duration of the test, the idea of doing tracer tests was finally abandoned. To gain an understanding of the physical processes appearing in both reservoir sections, a Computational Fluid Dynamic simulation was considered since this is now able to give results very close to reality.

CHAPTER 4

COMPUTATIONAL FLUID DYNAMIC MODELLING

In this chapter, the behaviour of the reservoir is simulated using the Computational Fluid Dynamic, software FLUENT. This study led to the creation of compartment models, which are not computationally intensive but are likely to give a good approximation of the system.

Computational Fluid Dynamics (CFD) seeks to find a solution to the fundamental equations, which describe the motion of fluids. Early CFD code was often written for specialist applications and its applicability was limited to the specific problem for which it was created. Many general purpose CFD packages are now commercially available. The University of Natal uses FLUENT (V4.5 and V5.5) from Fluent Inc (Lebanon, New Hampshire, USA) for CFD software.

However the FLUENT software is not actually designed to be suitable for on-line control application. Within the framework of this project, the aim of doing CFD modelling was to gain an understanding of the physical processes in the chlorine contact reservoir, and not to create a very accurate model for predicting the outlet chlorine concentration at the exit of the reservoir. The insight gained from the CFD model was subsequently used to guide the development of simplified compartment model for control purposes.

4.1 EQUATIONS SOLVED

The chlorine contact reservoir is a turbulent system. Turbulent flows are represented by the partial differential Navier-Stokes equations for mass, momentum, energy, and species conservation. To solve these equations, FLUENT divides the space in which the problem is posed into a solution mesh, which consists of a large number of cells, which fill the entire space. The partial differential equations are discretized over the mesh and then solved iteratively. A FLUENT simulation is considered converged when all governing equations are balanced within an acceptable value of error at each point in the solution domain. In this case, the simulation was considered well converged when the normalized residuals (velocities, pressure, turbulence, eddy dissipation, viscosity, chlorine concentration) were of the order of $1x10^{-5}$. In this work, a k-ε turbulence model was chosen. This model is a semi-empirical one that has been proven to provide engineering accuracy in a wide variety of turbulent flows. This involves two additional partial differential equations for the turbulent kinetic energy (k) and its rate of dissipation (ε).

The water level in the reservoir is time dependent. Hence the time dependent option in FLUENT V4.5 that allows a simulation of varying fluid depth using a deforming mesh was used.

4.2 COMPUTATIONAL GEOMETRY

4.2.1 The reservoir

The dimension of the reservoir were provided in Section 3.1.2 and shown in Figures 3-1 and 3-2. For each section of the reservoir, computational domains were built with the dimensions of the real section. The features included were the base, vertical walls and baffles, inlets and outlets. To avoid the complexities of a free surface model, the water surface was modelled as a rigid horizontal frictionless surface. A *deforming* mesh was used to represent the change in level as the reservoir filled or emptied with time. This model was not able to represent the situation when the water level dropped below the top of the baffle walls.

To create the deforming mesh, FLUENT needs different meshes with different heights, assuming that the level of water is always equal to the height of the modelled section. An intermediate time calculates an intermediate level by linear interpolation. Therefore, 9 computational domains were built with the same area and different heights for each, which were extracted for the maxima and the minima of the level during 1.6 days. Figure 4-1 shows the real variation of the level during 1.6 days in August 2001 and the linear interpolation. The inflow is taken constant at 100 ML/d.

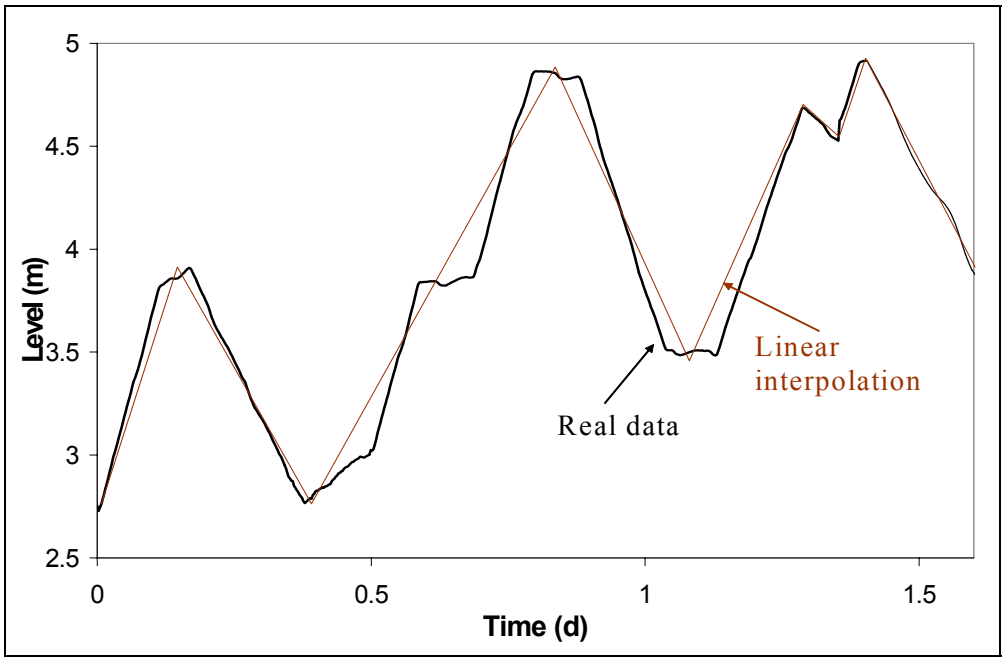


Figure 4-1: Level linearisation from 9h00 of 25th August 2001 to 20h00 of 26th August 2001

4.2.2 Mesh of the computational domain

In order to obtain a converged solution to the problem posed, the computational mesh accuracy has to be considered. In general, the finer the mesh, the more accurate the model is expected to be. However the dimensions of each section are very large, and increasing the number of cells in the mesh will increase the computational time, which must be considered in the light of the limited objectives of the CFD model.

A simple mesh was built with a spacing of 1 m between each node, which produced a mesh containing 55 932 cells, which could be solved in a reasonable time on the available computer. Figure 4-2 shows the mesh for section 1 and the Figure 4-3 shows it for the section 2.

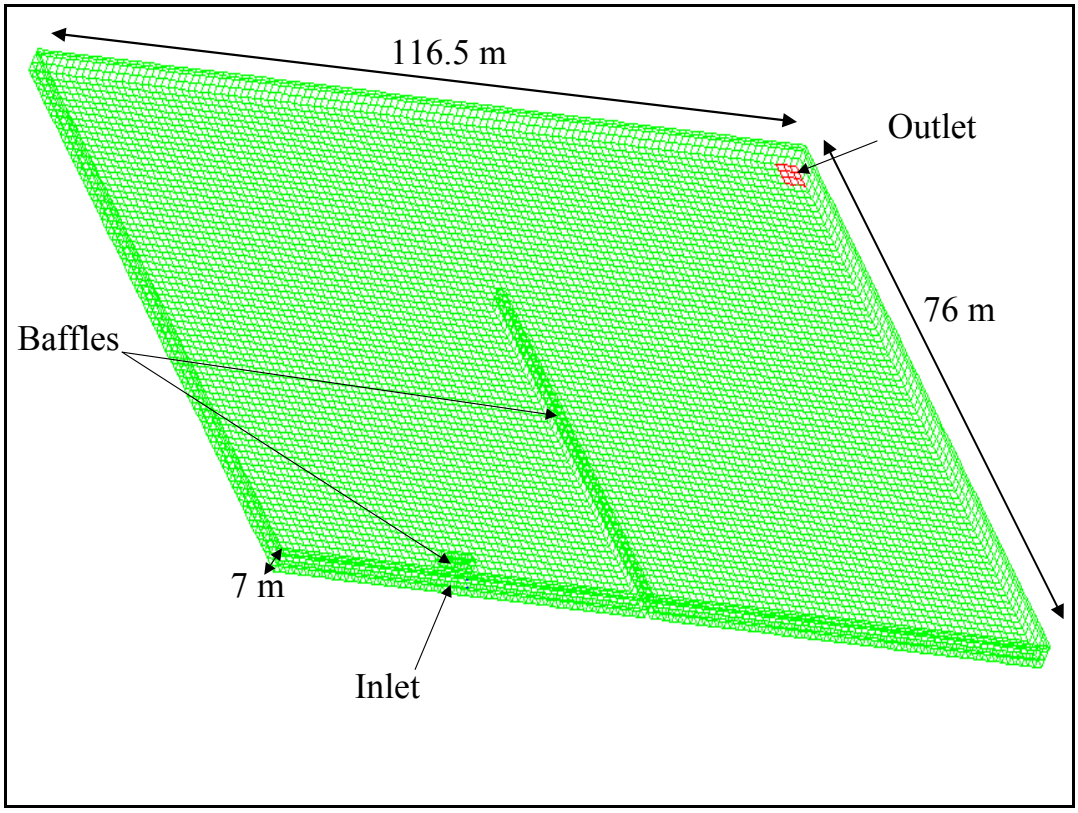


Figure 4-2: The mesh of the section 1 of the reservoir

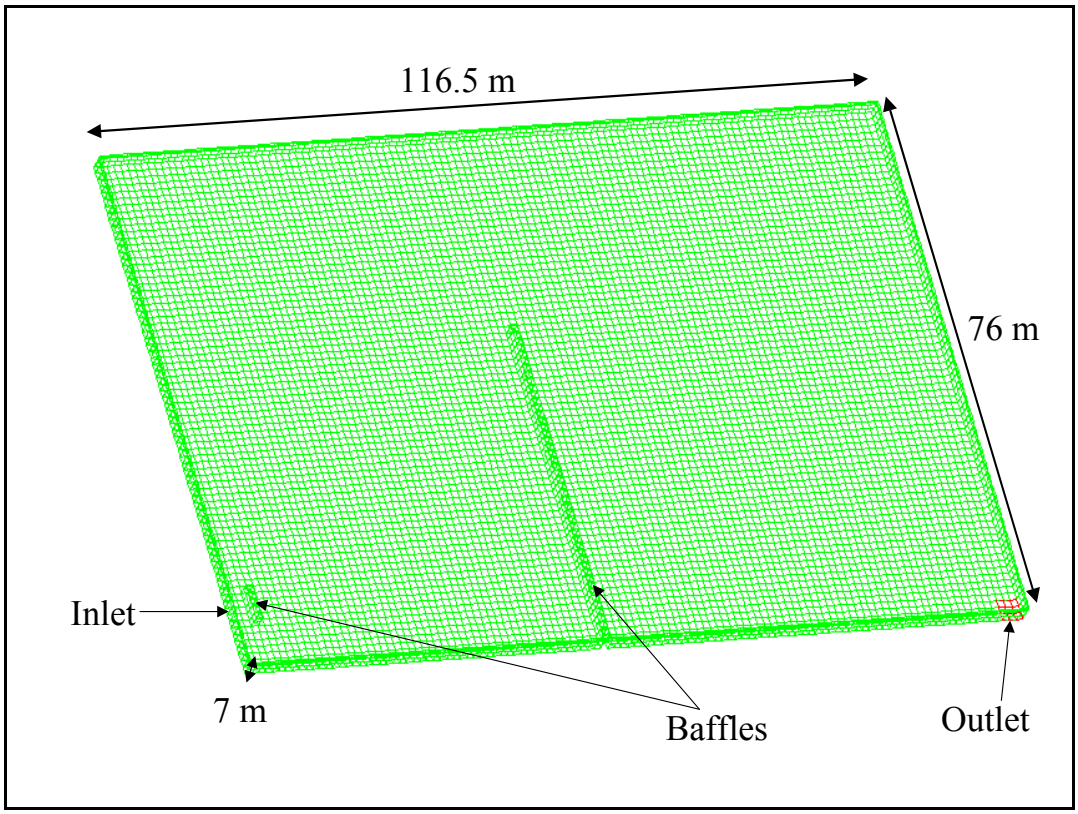


Figure 4-3: The mesh of the section 2 of the reservoir

4.3 INITIAL AND BOUNDARY CONDITIONS

4.3.1 Boundary conditions

Fluent provides a standard set of boundary conditions, which can be used to represent inlets, outlets and walls.

4.3.1.1 Inlet

The FLUENT *inlet velocity* boundary condition was used to define the flow at the inlet. The area of the two reservoir are equal, the outlets are connected and hence the water levels in the two sections are equal, it was assumed that the flow entering each section was equal to half of the overall flow. The normal flow rate is equal 100 ML/d for this set of data, which means 50 ML/d to each section, i.e. 0.6365 m/s entering through 1 m diameter pipes.

4.3.1.2 Outlet

The outlets (3 m by 3 m) are pits sunk into the floor of the reservoir. The outlet flow rates were calculated by mass balance bearing in mind the rate of change of volume in the sections.

4.3.1.3 Wall

The heights of the wall are 7 m but the baffle walls are equal to 2 m in height. At the surface of the walls, the flow rate is equal to 0 m/s.

4.3.2 Initial conditions

Time dependent partial differential equations require specifications of initial conditions throughout the domain, and boundary conditions for the entire solution period; but initial conditions of the problem are unknown because they are determined by the entire previous history.

To get round this problem, a steady state solution was obtained for the height and the inflow set in Section 4.3.1.1. Thus the steady state was considered as the initial conditions for the simulation. Although this steady state will not be an accurate representation of conditions at the start of the simulation, the influence of the initial conditions decreases as the solution time increases.

4.4 SIMULATION RESULTS

A simulation of the flow pattern was first undertaken, then FLUENT is used to simulate the chlorine decay inside both sections of the reservoir.

4.4.1 Flow patterns

The section height of all the plan views shown is equal to 3 m (ie. the view is a slide through the reservoir at the height of 3 m independently of the water level, if the water level is below 3 m, the velocity field at the free surface is shown). Since the reservoir sections never operate at steady state, this solution (Figure 4-4) cannot be taken as a detailed representation of the flow pattern in practice. However it probably represents some kind of average behaviour of the flow. It indicates the presence of preferred pathways (the light areas), particularly in section 1, for flow through the reservoir, creating relatively stagnant volumes (dark areas). The numbers 1 and 2 refer to the section 1 and 2 of the reservoir respectively.

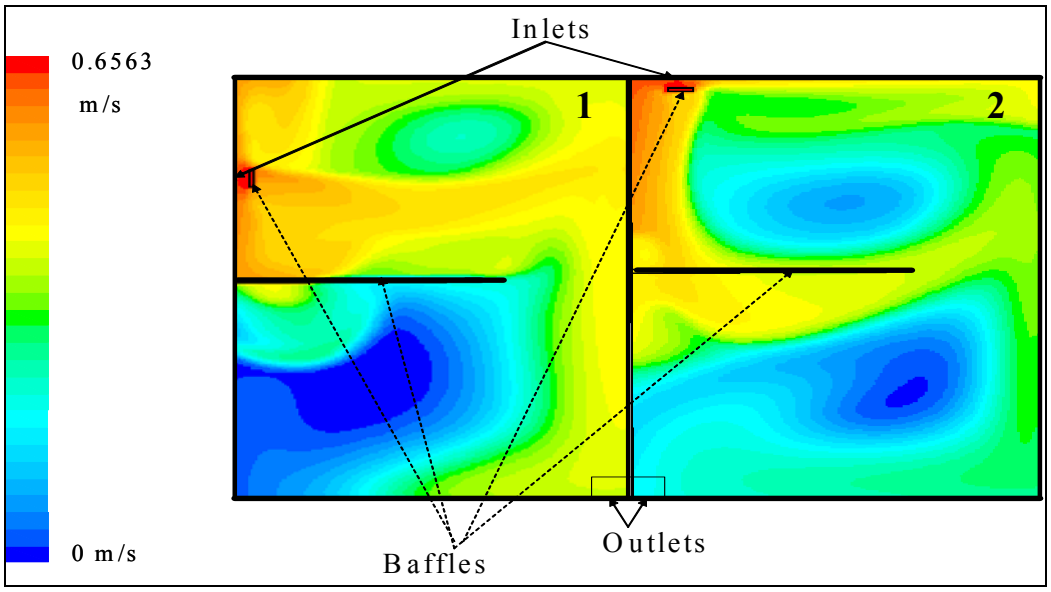


Figure 4-4: Steady state of the reservoir simulated by FLUENT (level = 2.8 m)

Starting from the initial steady state solution, the time dependent simulation evolved over a period of 1.6 d as indicated in Figure 4-1. Thus, it was hoped that the dynamic state would approach the state of the reservoir in reality. Figures 4-5 to 4-9 show the variation of flow patterns (at the height of 3 m, or less) in the sections when the level increases with time: from 2.75 m (0.5 d)(Figure 4-5) then 3.83 m (0.7 d) (Figure 4-6) to 4.77 m (1.3 d)(Figure 4-7) and then decreases to 4.09 m (Figure 4-8) to finally reach 3.5 m (Figure 4-9). This particular period of time has been chosen because the simulation has already calculated one rise and one drop of level and so the influence of the initial condition should have largely disappeared (the disc with the dissertation contains diagrams of the 44 simulations over the 1.6 d period)

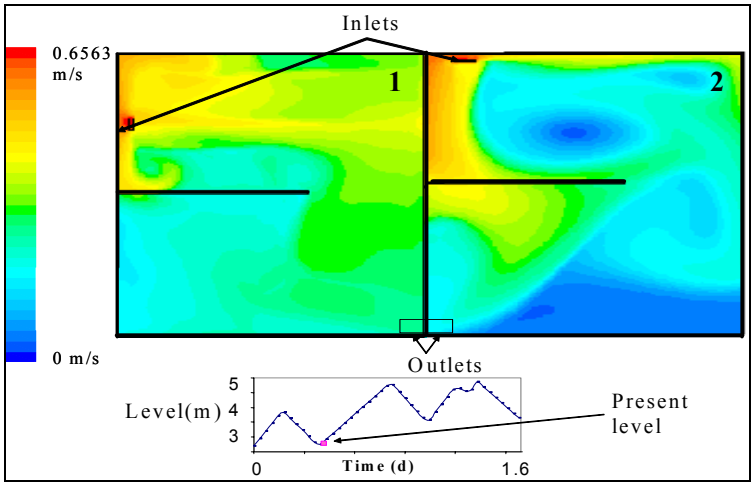


Figure 4-5: Plan view of both sections (level = 2.75 m)

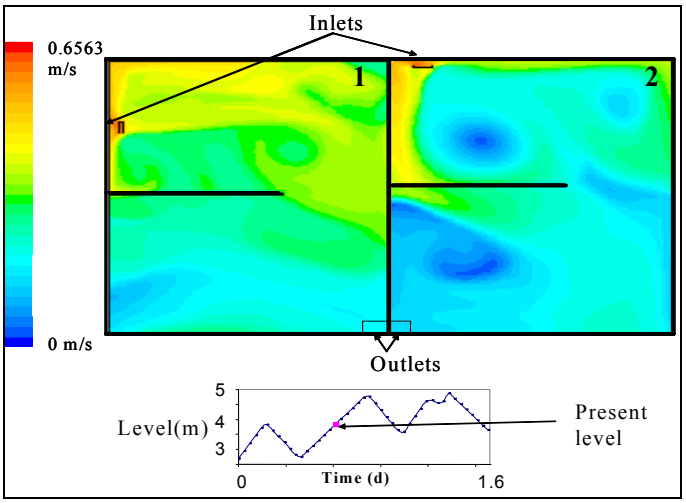


Figure 4-6: Plan view of both sections (level = 3.83 m)

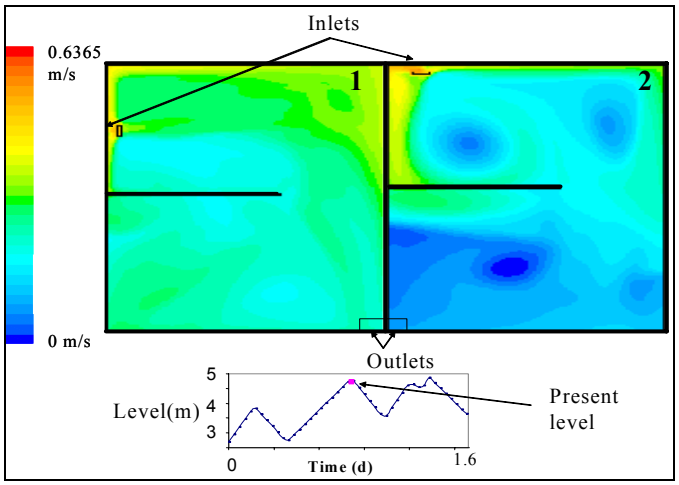


Figure 4-7: Plan view of both sections (level = 4.77 m)

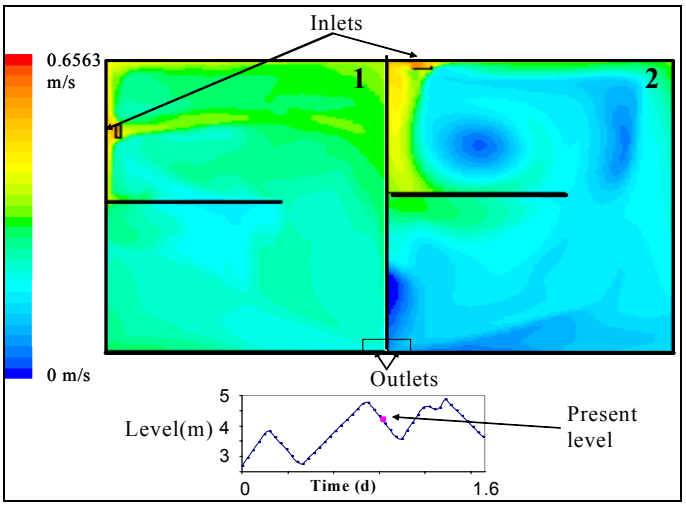


Figure 4-8: Plan view of both sections (level = 4.09 m)

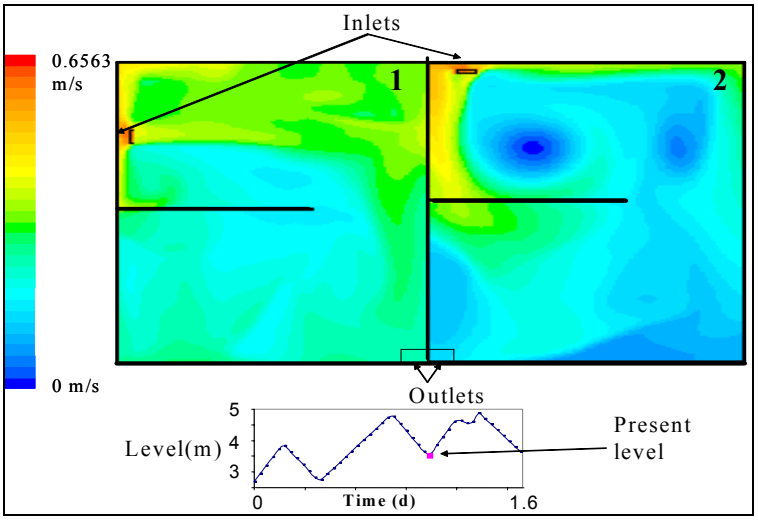


Figure 4-9: Plan view of both sections (level = 3.5 m)

As shown in these figures, when the level increases, the stagnant volumes increase in certain areas, whilst in the whole area the flow rate decreases. That is the consequence of the rising of the level: as the inflow is constant, the outflow is smaller so the water experiences a longer residence time, and the flow rates drop. As the level decreases, the reverse happens.

4.4.2 Chlorine decay simulation

The chemical reaction modelling facilities of FLUENT were used to simulate the behaviour of the chlorine in the reservoir sections. At the inlet, two species were defined: water and chlorine. The inlet chlorine concentration has been approximated by a piecewise linear representation as shown in Figure 4-10 as it was done for the level.

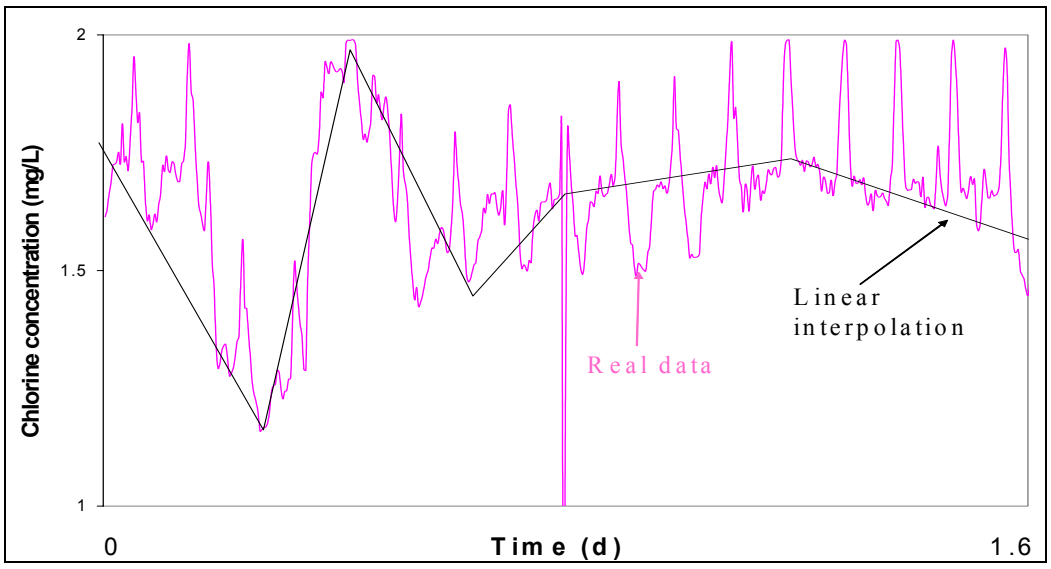


Figure 4-10: Chlorine concentration linearisation from 9h00 of 25th August 2001 to 20h00 of 26th August 2001

For the CFD simulation, the chlorine decay consumption was modelled using a first order decay with a kinetic factor equal to 2 d^{-1}. The CFD simulations were repeated with kinetic factors of 0.5, 1, and 1.5 d^{-1}. Figures from 4-11 to 4-15 illustrate the FLUENT simulation with k = 2 d^{-1}.

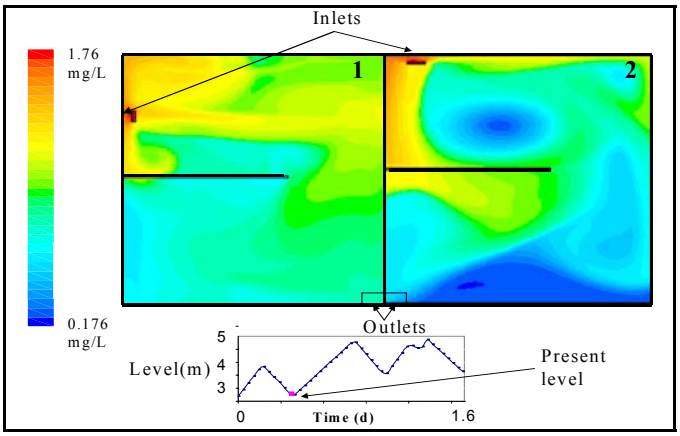


Figure 4-11: Plan view of both sections (outlet chlorine concentration: section 1 = 0.817 mg/L; section 2 = 0.812 mg/L)

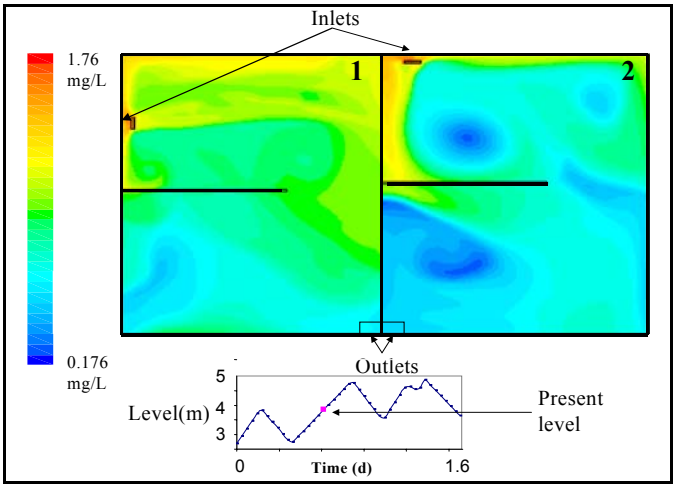


Figure 4-12: Plan view of both sections (outlet chlorine concentration: section 1 = 0.622 mg/L; section 2 = 0.709 mg/L)

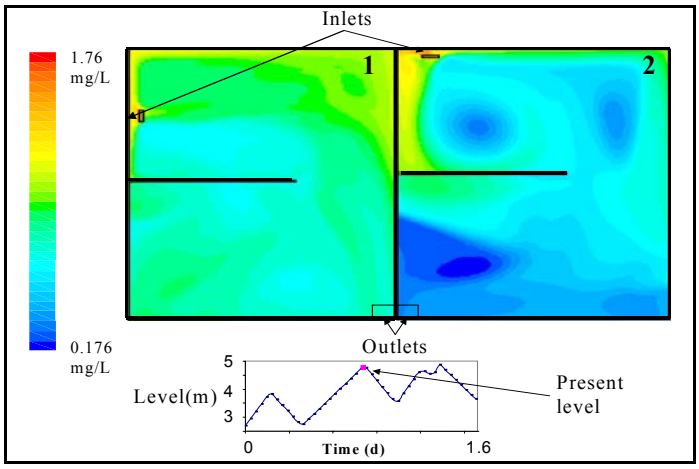


Figure 4-13: Plan view of both sections (outlet chlorine concentration: section 1 = 0.762 mg/L; section 2 = 0.661 mg/L)

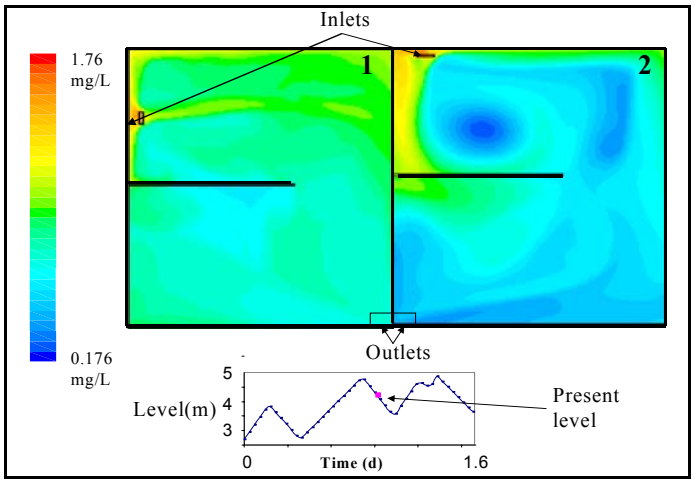


Figure 4-14: Plan view of both sections (outlet chlorine concentration: section 1 = 0.715 mg/L; section 2 = 0.662 mg/L)

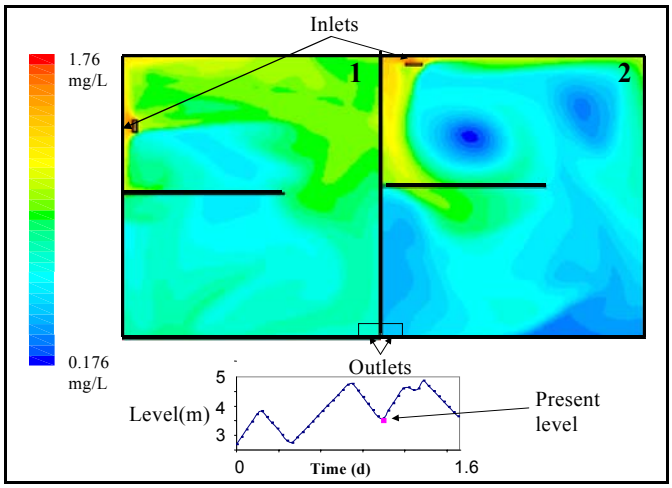


Figure 4-15: Plan view of both sections (outlet chlorine concentration: section 1 = 0.779 mg/L; section 2 = 0.773 mg/L)

As it was already seen, the flows through the two sections have preferred pathways. In those parts of the volume which are not on these preferred paths, the water experiences longer residence times, which causes the chlorine concentration to be depleted. The graphs describing the flow pattern or the chlorine concentration inside the sections consequently show similar features. The same conclusion can be reached: as the level increases, the chlorine concentration decreases in certain areas, whilst in the whole area the average of the chlorine concentration decreases (as the level increases, the residence time of the water increases, giving the chlorine more time to react, first order decay model). As the level decreases, the reverse happens. The outlet chlorine concentrations predicted by FLUENT can be compared with the real data (Figure 4-16).

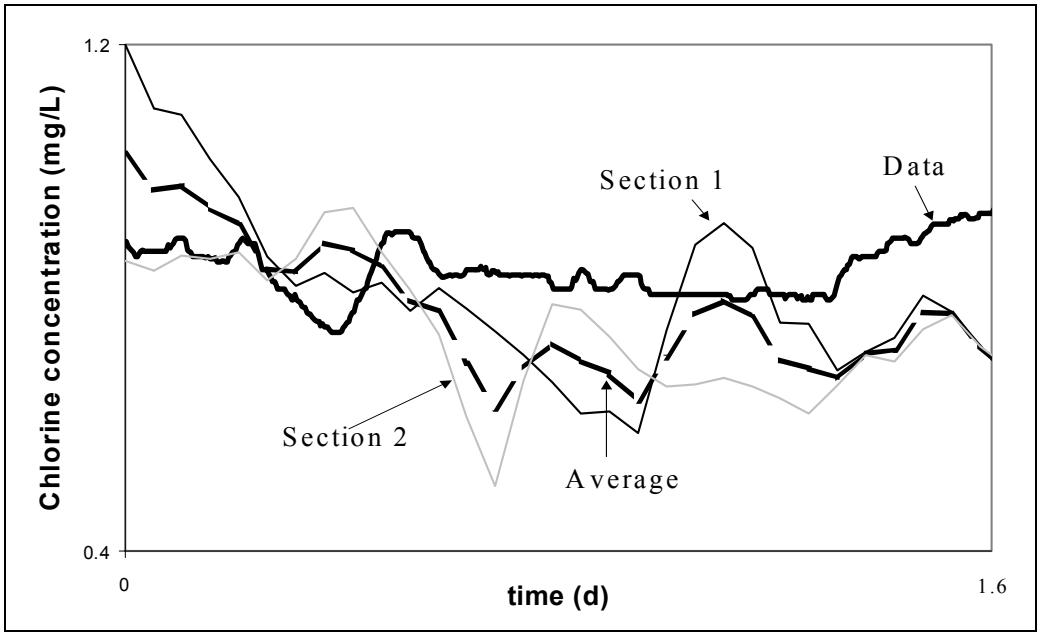


Figure 4-16: Comparison between the predicted outlet chlorine concentration and the data

On these curves, it was observed firstly that the two sections are not giving the same outlet chlorine concentration. At 0.5 d and at 0.9 d of the simulation, the two curves follow the same trend but at 1.3 d, they are completely divergent. And this cannot be explained by the level variation because at this precise time, the level is continuously increasing or decreasing. These differences are certainly due to the different geometry of the two sections.

As it has been assumed that the inflow is equally divided between the two sections, the average outlet chlorine concentration is the average of the two values obtained for the outlet from each section. The second observation was that the kinetic factor k is probably too high (the prediction are below the data, which means than too much chlorine has disappeared).

Simulations were conducted with different values for the kinetic factor k (0.5; 1; 1.5 and 2 d^{-1}). Figure 4-17 shows the average outlet chlorine concentration for each value of the kinetic factor.

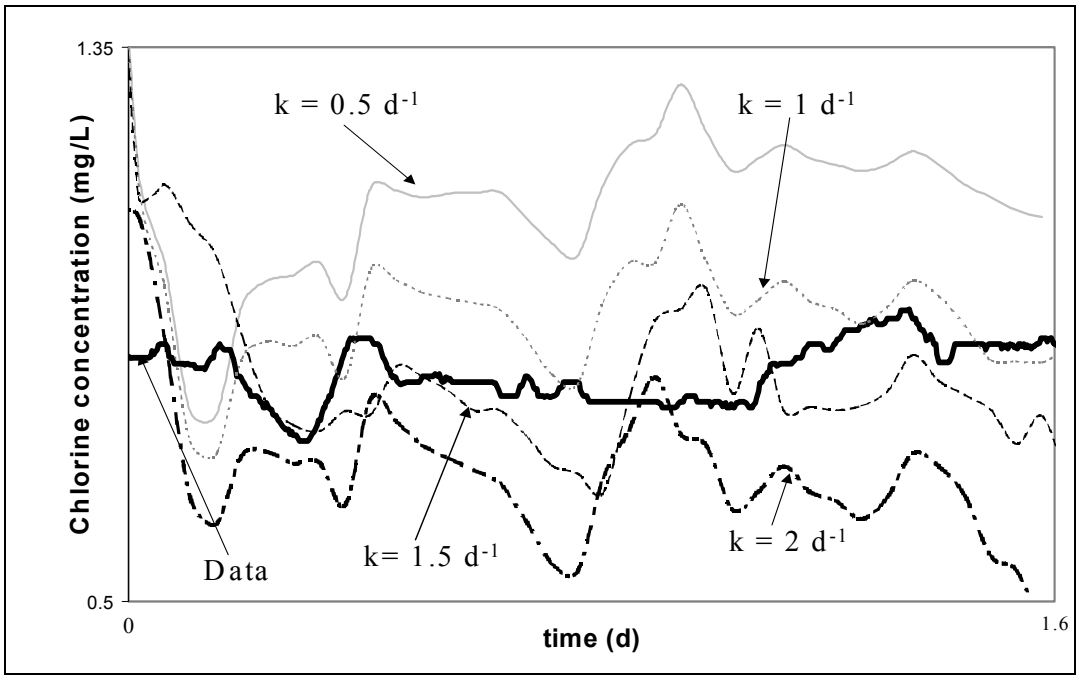


Figure 4-17: Comparison of the predicted outlet chlorine concentrations with different values of k

The best results are obtained with the kinetic factor equal to 1.5 d^{-1} (which was the average value found by a laboratory study, which was conducted after the simulations had been performed (Kandasamy and Govender, 2002)). Although the predicted outlet chlorine concentration is of the same order as the measured data, it does not match it perfectly. Others influences, for example the variation of chlorine loss with splashing at the entries of both sections, or with the water quality need to be considered.

4.5 SIMPLIFIED COMPARTMENT MODEL

Even if CFD modelling gave correct results and a good prediction of outlet chlorine concentration, this kind of modelling is too detailed and computationally intensive to be applicable for real-time control, so a simplified model is required. The FLUENT software is also not designed to be suitable for on-line application. Thus it was planned to create compartment models using the MATLAB software.

Because of the difficulties associated with undertaking tracer tests (Section 3.3), tracer tests were simulated with CFD modelling to obtain residence time distribution curves. By

interpreting theses curves, different combinations of compartments (well mixed and plug flows) likely to give a good approximation of the system were found.

4.5.1 Tracer test simulation using CFD

To simulate a tracer test in FLUENT, a certain amount of a non-reactive species is added at t = 0, for just one time step. For this case, the chlorine concentration was equal to 1g/L for the first time step only. The result of the test is presented in figure 4-18. Initial and boundary conditions are as defined in Section 4.4.

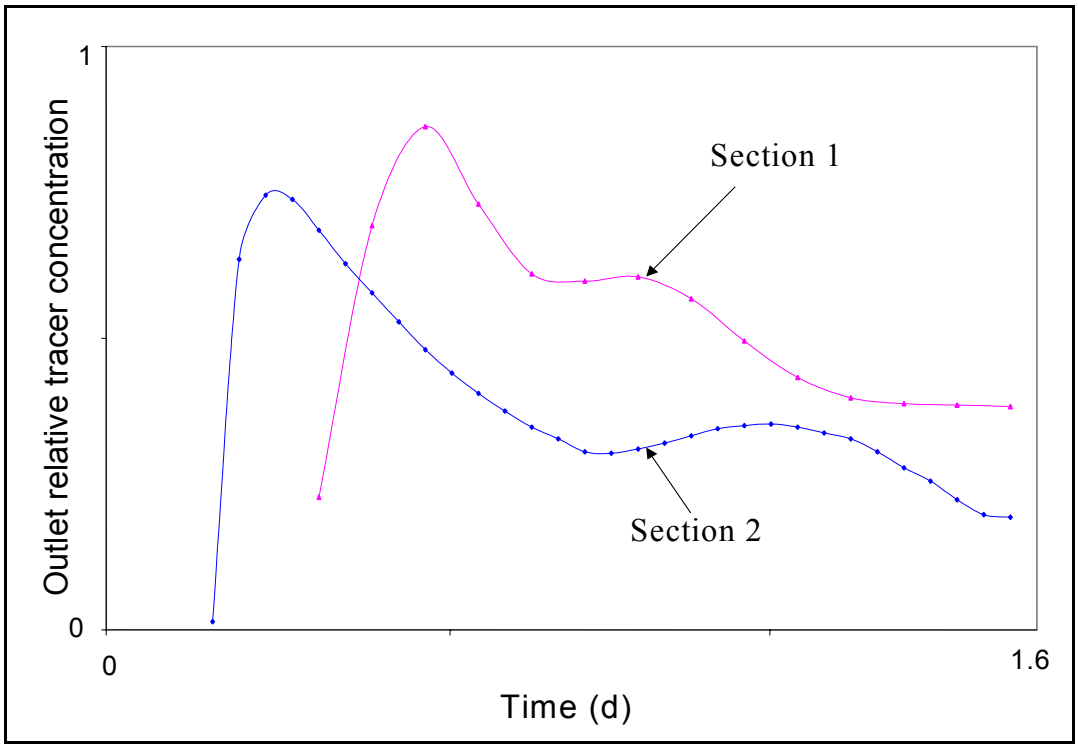


Figure 4-18: Impulse tracer test simulation using CFD

4.5.2 Interpretation

The two peaks shown in figure 4-18 for each section were explained with two different combinations of well-mixed flows and plug flows.

The first combination is based on dividing each section into two types of compartments. The first type lies on the preferred path of flow, and will be termed the *through-flow compartment* or volume. The second type falls off the preferred path. Thus, it does not interchange water directly with the inlet and outlet, but rather with the through-flow volume. This situation is similar to flow experienced by the backwater of a coastal lagoon, where water flows in and out

as a result of the action of the tides, but there is no through flow. For this reason, this type of compartment will be called a *tidal-flow compartment* or volume (Figure 4-19).

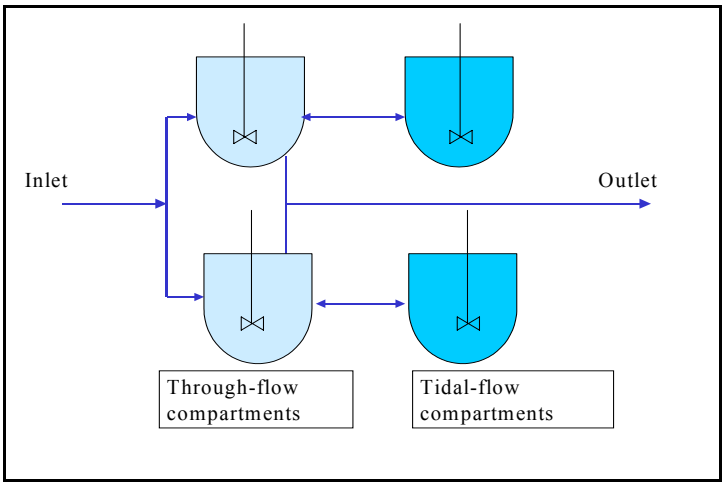


Figure 4-19: Four-compartment model

For the second combination, it was assumed that the second peak that appears in the simulation was due to a recycle going through a dead-time zone (the dead zone has been simulated by a plug flow). And, as there is a little time delay before the first peak, a well mixed compartment has been positioned before the re-circulation. (Figure 4-20).

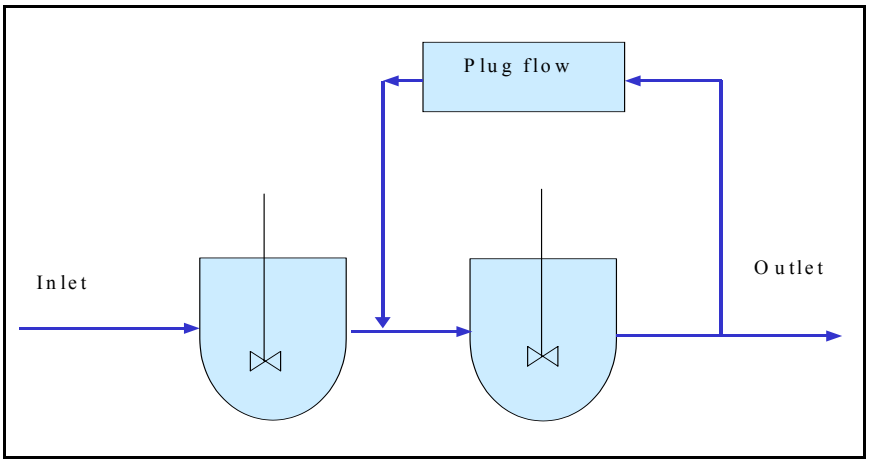


Figure 4-20: Compartment with plug-flow re-circulation

Finally, these two models were created using MATLAB programming to predict and control the outlet chlorine concentration. In the next chapter, these two compartment models are modelled with equations and solved using an extended Kalman filter.

CHAPTER 5

PROCESS MODELLING

This chapter describes the different compartment models, and how they are solved within an extended Kalman filter. Then the extended Kalman filter is tuned, and the results concerning the open-loop simulation are presented.

5.1 COMPARTMENT MODELS

If a mathematical model can be developed to represent the variation of the chlorine concentration in the reservoir, by taking the anticipated user demand patterns into account, the model could be used to calculate the chlorine dose required to provide the correct residual concentration.

Guided by the CFD modelling, various compartment models have been generated to simulate the behaviour of the plant, namely, a *four-compartment model* and a *six-compartment model*. A *one-compartment model* has also been considered for the simplicity of implementation on-line. Chlorine decay in the model was represented by first order kinetics.

5.1.1 Four-compartment model

This model was the first one studied. As described in Section 4.5.2, two different kinds of volume have been considered for each section of the reservoir: the *through-flow* compartment and the *tidal-flow* compartment. Thus 4 compartments were modelled (Figure 5-1). The fraction, α, of the total feed flow that enters compartment 1 of the reservoir, has been taken equal to 0.5 (as in the FLUENT simulation, Section 4.3.1.1).

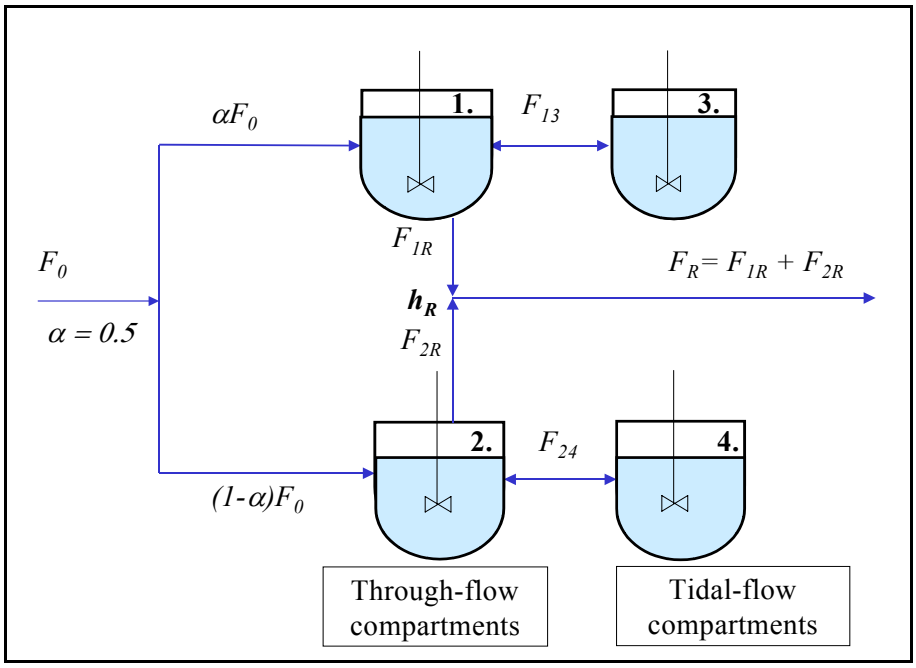


Figure 5-1: Four-compartment model

where F_0 = inlet flow (ML/d)

C_0 = inlet chlorine concentration (mg/L)

F_R = outlet flow (ML/d)

C_R = outlet chlorine concentration (mg/L)

h_R = pressure at the point where F_{1R} and F_{2R} are mixed (m)

h_i = height of water in the compartment i (m)

A_i = area of the liquid surface i (m^2)

C_i = chlorine concentration of compartment i (mg/L)

F_{ij} = flow from compartment i to j (negative if the flow goes from j to i) (m^3/d)

k = kinetic factor (d^{-1})

- **Model differential equations**

Volume balances:

$$A_1 \frac{dh_1}{dt} = \alpha F_0 - F_{13} - F_{1R} \tag{5.1}$$

$$A_2 \frac{dh_2}{dt} = (1-\alpha) F_0 - F_{24} - F_{2R} \tag{5.2}$$

$$A_3 \frac{dh_3}{dt} = F_{13} \tag{5.3}$$

$$A_4 \frac{dh_4}{dt} = F_{24} \tag{5.4}$$

Mass balances:

For the flows, we need to consider the direction of flow. The following equation uses *maximum* and *minimum* functions to describe this phenomenon.

$$A_1 \frac{dC_1 h_1}{dt} = \alpha F_0 C_0 - \max(F_{13}C_1, 0) - \min(F_{13}C_3, 0) - \max(F_{1R}C_1, 0) - \min(F_{1R}C_2, 0) - kA_1 C_1 h_1$$

so

$$A_1 h_1 \frac{dC_1}{dt} + A_1 C_1 \frac{dh_1}{dt} = \alpha F_0 C_0 - \max(F_{13}C_1, 0) - \min(F_{13}C_3, 0) - \max(F_{1R}C_1, 0) - \min(F_{1R}C_2, 0) - kA_1 C_1 h_1$$

and $A_1 \frac{dh_1}{dt}$ can be replaced by Equation 5.1 thus

$$\begin{aligned} A_1 h_1 \frac{dC_1}{dt} = {} & \alpha F_0 C_0 - \max(F_{13}C_1, 0) - \min(F_{13}C_3, 0) - \max(F_{1R}C_1, 0) - \min(F_{1R}C_2, 0) \\ & - kA_1 C_1 h_1 - C_1(\alpha F_0 - F_{13} - F_{1R}) \end{aligned} \tag{5.5}$$

The same reasoning can be made for each compartment:

$$\begin{aligned} A_2 h_2 \frac{dC_2}{dt} = {} & (1-\alpha) F_0 C_0 - \max(F_{24}C_2, 0) - \min(F_{24}C_4, 0) - \max(F_{2R}C_2, 0) - \min(F_{2R}C_1, 0) \\ & - kA_2 C_2 h_2 - C_2((1-\alpha) F_0 - F_{24} - F_{2R}) \end{aligned} \tag{5.6}$$

$$A_3 h_3 \frac{dC_3}{dt} = \max(F_{13}C_1, 0) + \min(F_{13}C_3, 0) - kA_3 C_3 h_3 - C_3 F_{13} \tag{5.7}$$

$$A_4 h_4 \frac{dC_4}{dt} = \max(F_{24}C_2, 0) + \min(F_{24}C_4, 0) - kA_4 C_4 h_4 - C_4 F_{24} \tag{5.8}$$

- **Model algebraic equations**

Pressure balances (the function *sgn* determines the sign of the expression in brackets):

$$F_{13} = \beta_{13} \operatorname{sgn}(h_1 - h_3) \sqrt{|h_1 - h_3|} \tag{5.9}$$

$$F_{24} = \beta_{24} \operatorname{sgn}(h_2 - h_4) \sqrt{|h_2 - h_4|} \tag{5.10}$$

$$F_{1R} = \beta_{1R} \operatorname{sgn}(h_1 - h_R) \sqrt{|h_1 - h_R|} \tag{5.11}$$

$$F_{2R} = \beta_{2R} \operatorname{sgn}(h_2 - h_R) \sqrt{|h_2 - h_R|} \tag{5.12}$$

The flow coefficients for reservoir compartment interconnection β_{1R}, β_{2R}, β_{13} and β_{24} are set as high as possible (100 ML/(d$\sqrt{\text{m}}$)) to balance the levels almost instantly.

Mass balances:

$$F_R = F_{1R} + F_{2R} \tag{5.13}$$

$$F_R C_R = \max(F_{1R}C_1, 0) + \min(F_{2R}C_1, 0) + \max(F_{2R}C_2, 0) + \min(F_{1R}C_2, 0) \tag{5.14}$$

5.1.2 *Six-compartment model*

The data logged by the Waterworks SCADA system are for the whole reservoir, and are not available for each section of the reservoir. In a second approach, the individial sections are not considered in detail but as one hypothetical reservoir, allowing for re-circulation flow. The six-compartment model consists of two well-mixed flows and one plug-flow recycle. However, a plug-flow involves dead-time, which is difficult to insert in a system containing differential and algebraic equations. Therefore, the plug flow has been approximated by four mixed compartments in series (Figure 5-2). Where α is the fraction of the outflow from the hypothetical compartment (compartment 2) which goes directly to the exit.

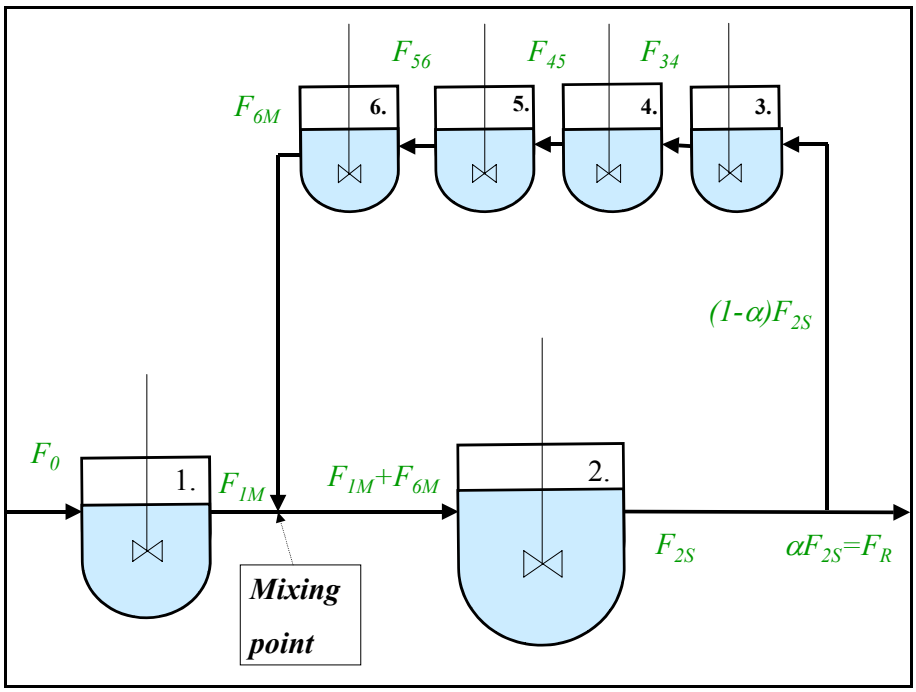


Figure 5-2: Six-compartment model

- **Model differential equations**

Volume balances:

$$A_1 \frac{dh_1}{dt} = F_0 - F_{1M} \tag{5.15}$$

$$A_2 \frac{dh_2}{dt} = F_{2S} - (F_{1M} + F_{6M}) \tag{5.16}$$

$$A_3 \frac{dh_3}{dt} = (1-\alpha)F_{2S} - F_{34} \tag{5.17}$$

$$A_4 \frac{dh_4}{dt} = F_{34} - F_{45} \tag{5.18}$$

$$A_5 \frac{dh_5}{dt} = F_{45} - F_{56} \tag{5.19}$$

$$A_6 \frac{dh_6}{dt} = F_{56} - F_{6M} \tag{5.20}$$

Mass balances:

As was done previously (for equation 5.5):

$$A_1 h_1 \frac{dC_1}{dt} = F_0 C_0 - F_1 C_1 - kA_1 h_1 C_1 - C_1 (F_0 - F_1) \tag{5.21}$$

A simplification can be made:

$$A_1 h_1 \frac{dC_1}{dt} = F_0 (C_0 - C_1) - kA_1 h_1 C_1 \tag{5.22}$$

The same reasoning can be used for each compartment:

C_M = chlorine concentration at the mixing point (Figure 5-2).

$$A_2 h_2 \frac{dC_2}{dt} = (F_{1M} + F_{6M})(C_M - C_2) - kA_2 h_2 C_2 \tag{5.23}$$

$$A_3 h_3 \frac{dC_3}{dt} = (1-\alpha) F_{2S} (C_2 - C_3) - kA_3 h_3 C_3 \tag{5.24}$$

$$A_4 h_4 \frac{dC_4}{dt} = F_{34} (C_3 - C_4) - kA_4 h_4 C_4 \tag{5.25}$$

$$A_5 h_5 \frac{dC_5}{dt} = F_{45} (C_4 - C_5) - kA_5 h_5 C_5 \tag{5.26}$$

$$A_6 h_6 \frac{dC_6}{dt} = F_{56} (C_5 - C_6) - kA_6 h_6 C_6 \tag{5.27}$$

- **Model algebraic equation**

Pressure balances:

$$F_{1M} = \beta_{1M} \operatorname{sgn}(h_1 - h_2) \sqrt{|h_1 - h_2|} \tag{5.28}$$

$$F_{34} = \beta_{34} \operatorname{sgn}(h_3 - h_4) \sqrt{|h_3 - h_4|} \tag{5.29}$$

$$F_{45} = \beta_{45} \operatorname{sgn}(h_4 - h_5) \sqrt{|h_4 - h_5|} \tag{5.30}$$

$$F_{56} = \beta_{56} \operatorname{sgn}(h_5 - h_6) \sqrt{|h_5 - h_6|} \tag{5.31}$$

$$F_{62} = \beta_{62} \operatorname{sgn}(h_2 - h_6) \sqrt{|h_6 - h_2|} \tag{5.32}$$

All flow coefficients for reservoir compartment interconnections are equal to 100 ML/(d$\sqrt{\text{m}}$), to balance the model almost instantly.

Mass balances:

$$F_R = \alpha F_{2S} \tag{5.33}$$

$$C_M(F_{1M} + F_{6M}) = C_1 F_{1M} + C_6 F_{6M} \tag{5.34}$$

5.1.3 One-compartment model

In order to implement the model on-line, the reservoir has been considered as a single compartment (Figure 5-3).

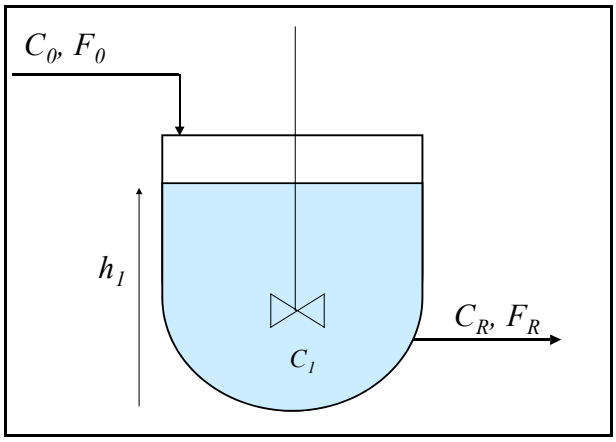


Figure 5-3: One-compartment model

- **Model differential equations**

Volume balance:

$$A_1 \frac{dh_1}{dt} = F_1 - F_0 \tag{5.35}$$

Mass balance:

$$A_1 h_1 \frac{dC_1}{dt} = F_1(C_0 - C_1) - kA_1 h_1 C_1 \tag{5.36}$$

- **Model algebraic equation**

Mass balances:

$$F_R = F_1 \tag{5.37}$$

$$C_R = C_1 \tag{5.38}$$

5.2 EXTENDED KALMAN FILTER

All of the mathematical models described above are non-linear with a mix of differential and algebraic equations (DAE). The kinetic factor k is an unknown parameter, and it is needed by the model to represent the outlet chlorine variation. The Kalman filter is a stochastic filter that allows the estimation of the states of the system based on a linear model. The extended Kalman filter (EKF) uses local linearisation to extend the scope of the Kalman filter to systems described by non-linear ordinary differential equations (ODE). This scheme has been applied to the state and parameter estimation using models described by ODEs.

5.2.1 Linearisation of DAEs

By applying Taylor series expansion and truncating after the first order term, the process model is linearised taking into account the DAE nature of the models. The differential equations can be re-written in this form:

$$\frac{dy}{dt} = f(y,z) \tag{5.39}$$

As well as the algebraic equations:

$$0 = g(y,z) \tag{5.40}$$

where y is the vector of state variables, and z the vector of additional variables used in the equations.

The EKF algorithm is detailed in Appendix A. The Jacobian is calculated assuming that f and g functions are differentiable in their arguments. Notice that for a local linearisation a perturbation method is used in the EKF algorithm. The Jacobian matrices are re-evaluated at every iteration by perturbing each variable in turn, thus the values of each element of the matrices change slowly as the process moves to a new operating point. A good approximation of the initial operating point is required to accelerate the convergence. The developed EKF algorithm has the advantage of reducing the problem of singularity since both excess equations and excess variables may be specified. The solution simply achieves the best least squares fit to this specification. Where there is no reason to change an excess variable, it is simply left at its original value.

The linear model obtained has the form given by equation A.6 (see extended Kalman filter algorithm in Appendix A) as:

$$\begin{pmatrix} \dot{y} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} f_0 \\ h_0 \end{pmatrix} + \begin{bmatrix} A & B \\ 0 & E \end{bmatrix} \begin{pmatrix} y \\ z \end{pmatrix} \tag{5.41}$$

5.2.2 Discrete model

Because discrete time formulations are most especially suited to computer simulation of process behaviour, a discrete-time model of the process based on the linear model, and the relationship of model states to the measurements w_t are respectively given by equations A.3 and A.4 as follows (the intermediate steps are described in Appendix A, Equation A.7 to A.10):

$$x_{t+\Delta t} = A_t x_t + B_t u_t \tag{5.42}$$

$$w_t = C_t x_t \tag{5.43}$$

where x is the state vector, u the input vector of independent variables, t represents the time and w the system output. C is an observation matrix while A and B are matrices of appropriate dimensions (n x n and n x m respectively, for an n-dimensional state and an m-dimensional input). They typically correspond to values of physical coefficients and property constants.

5.2.3 Kalman filter

With the equations 5.42 and 5.43, the transient response of the model can thus be founded using the Kalman filter. The equations 5.42 and 5.43 can be augmented as follows:

$$x_{t+\Delta t} = A_t x_t + B_t u_t + \delta_{t+\Delta t} \tag{5.44}$$

$$w_t = C_t x_t + \mu_t \tag{5.45}$$

where δ and μ are process and measurement noise contributions acting on the states and measured outputs respectively. They result from both measurement imperfections and disturbances affecting the process. They are considered to be random variables with normal distributions and zero means, with covariances E:

$$\mathrm{E}\left\{\bar{\delta}\,\bar{\delta}^T\right\} = R \tag{5.46}$$

$$\mathrm{E}\left\{\bar{\mu}\,\bar{\mu}^T\right\} = Q \tag{5.47}$$

and for uncorrelated δ and μ:

$$\mathrm{E}\left\{\bar{\delta}\,\bar{\mu}^T\right\} = [0] \tag{5.48}$$

The Kalman filter interpretation of this system taking into account the expected errors, is that the filter gain K, is calculated on each time step for adjusting x as follows:

$$K_t = M_t C_t^T \left[C_t M_t C_t^T + R \right]^{-1} \tag{5.49}$$

$$x_{t+\Delta t} = A_t x_t + B_t u_t + K_t \left[\hat{w}_t - C_t x_t \right] \tag{5.50}$$

$$M_{t+\Delta t} = A_t \left[I - K_t C_t \right] M_t A_t^T + Q \tag{5.51}$$

where $\hat{w}_t$ represents the equivalent set of measurements, M_i is the filter covariance matrix (initially small and diagonal), R and Q are usually diagonal. Higher prediction errors Q relative to observation errors R force the filter to follow observations more closely, whilst specifying higher observation errors R makes the model less sensitive to observations.

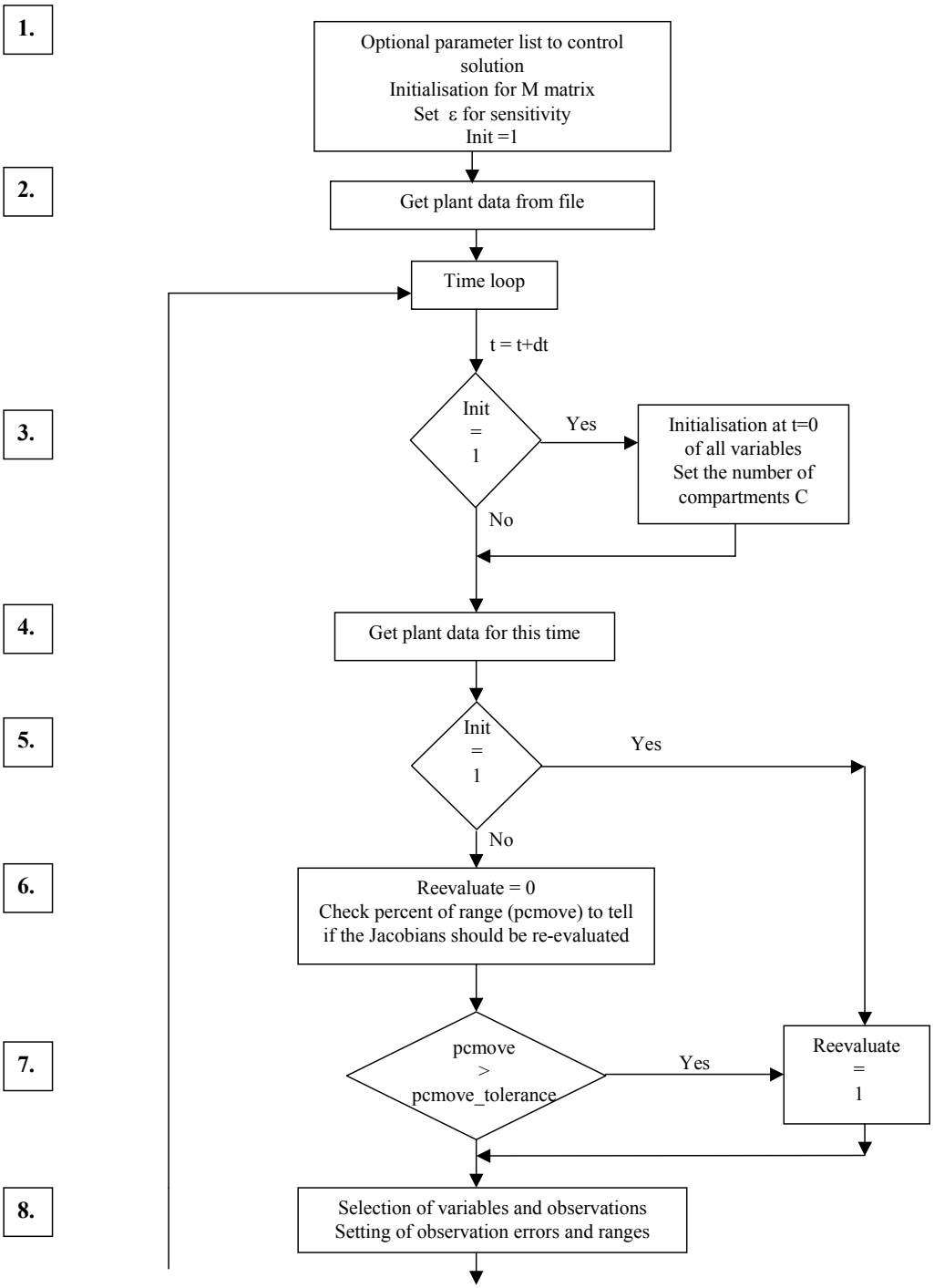
The initial conditions can be set so that x_0 is what we guess the parameter vector to be before we have seen the data, and M_0 is the initial covariance matrix, which reflects the confidence in this guess.

Equation 5.50 shows that the estimate x_{i+1} is obtained by adding a correction to the prediction of x_{i+1} based on x_i according to the model. The correction term for the model parameter vector is thus proportional to the prediction error (difference between the measured value of $\hat{w}_t$ and the prediction of $\hat{w}_t$) based on the previous estimate. The components of the Kalman filter gain matrix K_i are weighting factors that introduce an optimal correction into the integration cycle.

Notice that this form (equations 5.49 to 5.51) allows A and B to vary in time. This provides a way to handle non-linearities, since, as the process moves to a new operating point, elements of these matrices will change.

5.3 PROGRAM

MATLAB software was chosen to test the EKF algorithm. Indeed MATLAB is known for its efficiency concerning the matrix manipulation. The original MATLAB version of the extended Kalman filter was written by Mulholland (2001). The program is in Appendix C (for the six-compartment model). A table translates the symbols used in theory to their equivalents in the program in Appendix B. A simplified flow diagram illustrates the program in Figure 5-4.


1.
Optional parameter list to control solution
Initialisation for M matrix
Set ε for sensitivity
Init =1
2.
Get plant data from file
Time loop
t = t+dt
3.
Init = 1
Yes
Initialisation at t=0 of all variables
Set the number of compartments C
No
4.
Get plant data for this time
5.
Init = 1
Yes
No
6.
Reevaluate = 0
Check percent of range (pcmove) to tell if the Jacobians should be re-evaluated
7.
pcmove > pcmove_tolerance
Yes
Reevaluate = 1
8.
Selection of variables and observations
Setting of observation errors and ranges

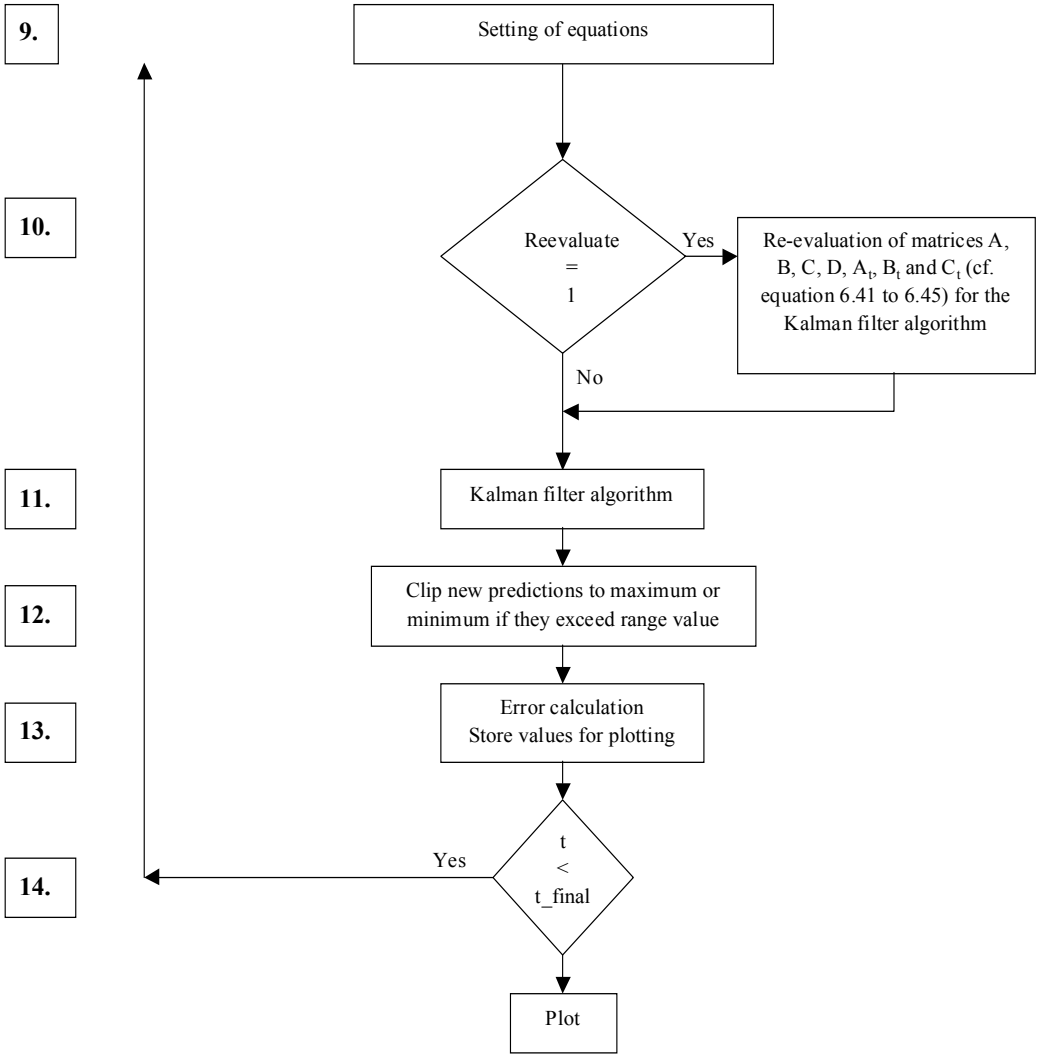


Figure 5-4: Simplified flow diagram of the program

The code allows easy alteration of compartment model structures.

5.3.1 Model initialisation (Step 1)

The operator sets the value of the optional parameters for the control solution. Equation 5.52 defines the measurement sensitivity ε:

$$\varepsilon = \frac{\text{Fractional error squared expected in the model}}{\text{Fractional error squared expected in the measurements}} \tag{5.52}$$

It is used to set the ration between the elements in the normalised Q and R matrices of the EKF, The initialisation of the matrix M (covariance matrix) is also done here (Equation 5.49). The parameter *Init* is set to 1 to allow an initialisation of all variables on the first iteration of the simulation.

5.3.2 Present operating conditions (Step 2)

The data are read from another file, which is presented as a data matrix with the time in the first column and each measurement in additional columns (inlet and outlet chlorine concentrations, inlet and outlet flow rates, level).

5.3.3 Variables initialisation (Step 3)

If it is the first iteration of the simulation, an initialisation of all the variables is done. The operator specifies the number of compartments *C*.

5.3.4 Data storage (Step 4)

For an easier reading of the program, each variable is stored from the data matrix (step 2) in a variable with a specific name.

5.3.5 Re-evaluation flag setting (Steps 5 to 7)

If it is the first iteration, the *Reevaluate* flag is directly set to 1, to force the calculation of the first Jacobians. After the first iteration, the *Reevaluate* factor is set to 0, and the program checks the percent of ranges moved since the last step to decide if the Jacobians must be re-evaluated or not.

5.3.6 Variables setting (Step 8)

Parameters such as the expected error in individual measurements are set.

5.3.7 Equations setting (Step 9)

The same is done for the equation system. Thus, it is possible to build all the necessary matrices for the extended Kalman filter from these two storage matrices. Moreover, only this part has to be changed between different models, allowing a clearer programming and time saving.

5.3.8 Jacobians re-evaluation (Step 10)

If needed, the Jacobians are re-evaluated.

5.3.9 Kalman filter algorithm (Steps 11 to 13)

The theory is translated into the MATLAB language. To facilitate the reading of the program, the Table in Appendix B translates the symbols used in the theory to their equivalents in the program. The calculated values are clipped to within constraints, and the error between the predicted outlet chlorine concentration and the data is calculated as well as the error concerning the ability of the system to match all observations.

5.3.10 Plotting

If the time loop is finished, the calculated values and the data are plotted.

5.4 RESULTS

5.4.1 Residence time distribution

All of the compartments in the different compartment models are defined by their water volumes. The water level changes but the cross sectional areas are fixed parameters. The effective cross sectional area of each compartment needs to be determined, so that the compartment models represent the real reservoir behaviour. For that, a tracer test simulation for each model has been undertaken with different cross sectional areas. The aim was to be closest to the tracer test simulation done with the CFD (Figure 4-18).

ε (measurement sensitivity) is set to 1 and M (covariance matrix) is initialised to 0.01. The data are fixed: the inflow is set equal to the outflow (=100 ML/d), and the height is set equal to 4 m. The kinetic factor k is set to 0 d^{-1} and thus the chlorine is considered as "the tracer". The inlet chlorine concentration is equal to 0 mg/L except between t = 2 days and t = 2.035 days, when it is equal to 2 mg/L to simulate an impulse test (Figure 5-5). The outlet chlorine concentration is considered as a non-observed variable and has to be calculated by the extended Kalman filter.

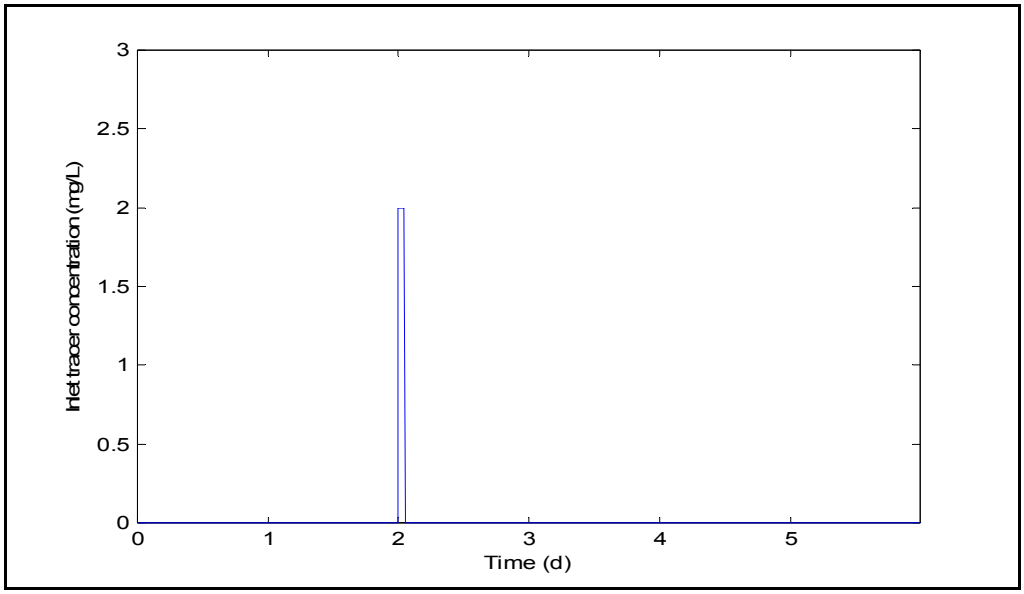


Figure 5-5: Impulse tracer test simulation for the inlet tracer concentration

5.4.1.1 Four-compartment model

If A_S is the cross sectional area of one section of the reservoir (both sections are equal), and A_1, A_2, A_3, and A_4 are the area of compartment 1, 2, 3 and 4 respectively (see figure 5-1), the following equations can be written, with γ equal to the ratio factor of A_1 to A_3, and A_2 to A_4.

$$A_S = A_1 + A_3 = A_2 + A_4 \tag{5.53}$$

$$A_1 = A_2 = \gamma A_S \quad \text{and} \quad A_3 = A_4 = (1-\gamma) A_S \tag{5.54}$$

Different values of γ have been tested. It is noted that γ is the fraction of the total volume which is "active" in mixing the through-flow, whereas the remaining volume only has inflow and outflow in the tidal sense, as the levels in compartments 3 and 4 (which are internally mixed) equilibrate to the levels in compartments 1 and 2 respectively. The best result is obtained with γ = 0.85 (Figure 5-6).

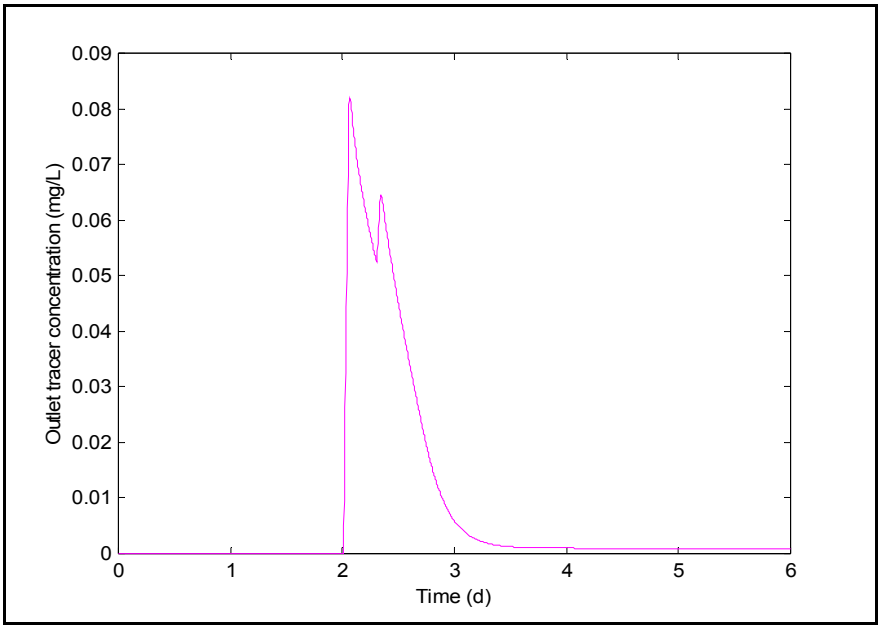


Figure 5-6: Impulse tracer test simulation with γ = 0.85 (four-compartment model)

In Figure 5-6, two peaks can be observed but the simulation does not match the FLUENT simulation closely (Figure 4-18). The first peak appears immediately after the impulse, due to the through flow compartments, which connect directly the inlet and the outlet. Consequently the model with the six compartments has been created (Section 5.1.2). The choice γ = 0.85 give the best results, and has been used below.

5.4.1.2 Six-compartment model

This model considers the entire reservoir, so the reservoir area will be considered as two times A_S (cross sectional area of one section of the reservoir). The cross sectional areas of the four

compartments approximating the plug flow are equal. The main compartment (compartment 2) must have the biggest area (according to the FLUENT simulation (Figure 4-18), and the first compartment must have a sufficient area to create a time delay (Figure 5-2).

The best results are obtained with:

$$A_1 = 0.2A_S \qquad A_2 = A_S \qquad A_3 = A_4 = A_5 = A_6 = 0.2A_S \quad (5.55)$$

and with α (fraction of the flow which is going to the re-circulation compartments) equal to 0.4 the flow coming from the main reservoir. The results are presenting in Figure 5-7.

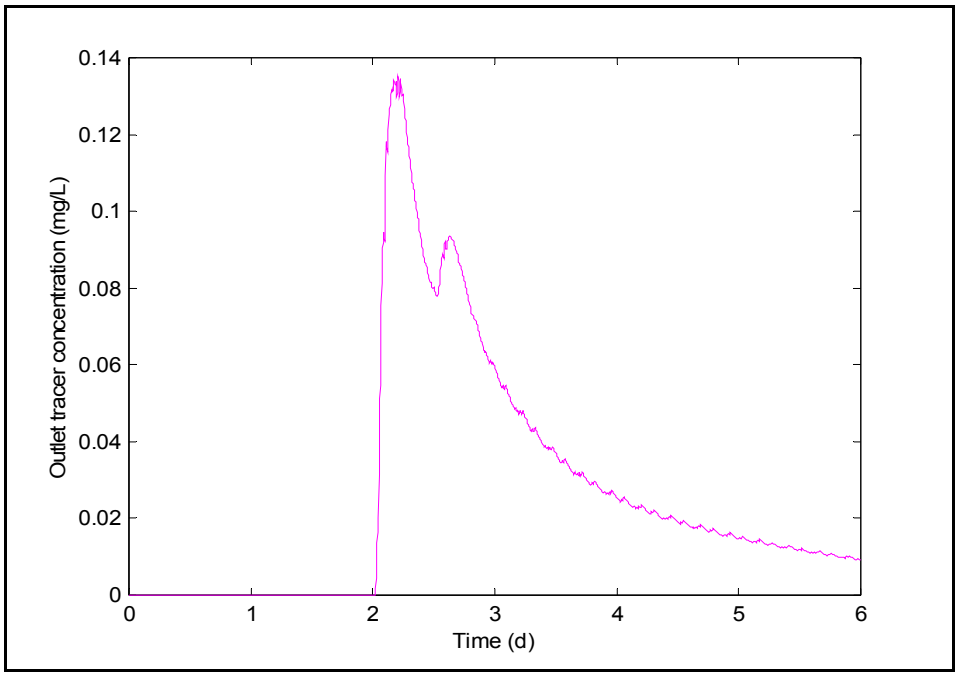


Figure 5-7: Impulse tracer test simulation (six-compartment model)

This simulation is closer to the FLUENT simulation (Figure 4-18). Consequently, this model is expecting to give better results. The noise after the second peak is due to the re-circulation.

5.5 IDENTIFICATION MODE

After all the physical variables of the mathematical system have been fixed, the EKF can be used to identify the kinetic factor *k*. The model is being supplied with actual plant measurement data varying in time, and the *k* variations that are found, are necessary to cause the model to predict the measured chlorine variations at the outlet, within the context of each model. Compared to water quality variations, relatively fast variations can occur as a result of water level variations (splashing at inlet, water path around baffles, etc). All of the data (inlet and outlet flow rates, inlet and outlet chlorine concentrations, and water level, measured on the plant over the period indicated below) are set as observed variables, and *k* has to be calculated as it

varies through this time period. k is initialised to $1d^{-1}$, which seems to be in the range of probable values. ε (measurement sensitivity) is kept equal to 1, the initial value of M (covariance matrix) is equal to 0.01.

The results are presented for the three models; the data period is from 10/10/01 to 12/10/01. In this period the level undergoes a lot of variation but the inlet and outlet chlorine concentrations are steadier (Figure 5-8).

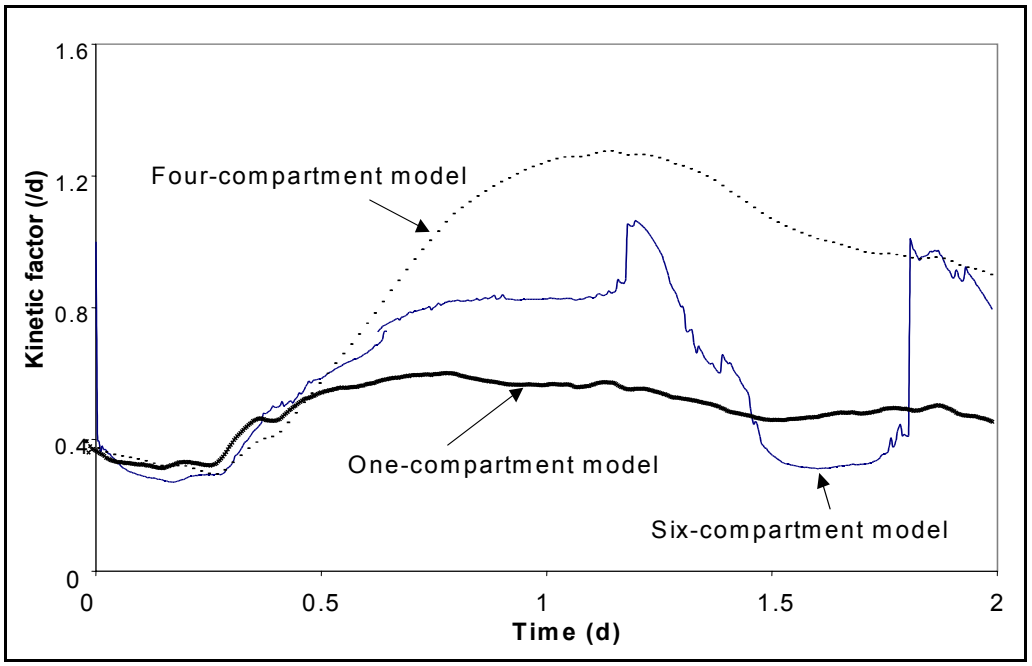


Figure 5-8: Kinetic factor value for the three models (October data)

To explain the difference between the three curves, it is important to know that the kinetic factor k is directly linked to the volume. The total volume for the three models is the same but it is not shared out in the same way, and hence the water experiences different residence times, resulting in the kinetic factors being different. This also explains why the kinetic factor is not equal to 1.5 d^{-1} as in the FLUENT simulation.

It can be seen that the one-compartment model (which is the simplest one) has a kinetic factor, which does not vary a lot. On the contrary, the kinetic factor for the six-compartment model, which is the most complicated, varies a lot. Figure 5-9 illustrates a calculation of the kinetic factor for another period of time (from 15/03/02 to the 17/03/02), which presents the data with different characteristics: the level is almost stable but the inlet and outlet chlorine concentration varies a lot.

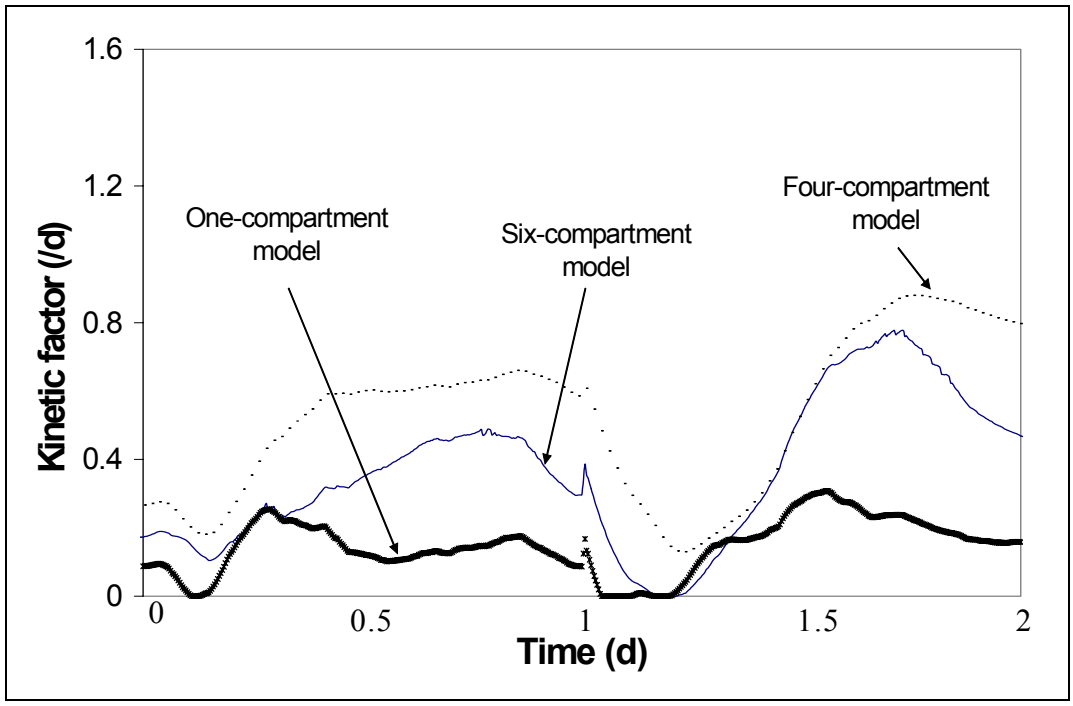


Figure 5-9: Kinetic factor for the three models (March data)

For this set of data, the calculated kinetic factors seem to follow the same trend, which confirm the fact that the kinetic factor is linked to the compartment model. Indeed, when the level is almost constant (constant volume), the kinetic factor undergoes the same variations, and the order is similar to the set of data in October: the kinetic factor curve for the four-compartment model is above the curve for the six-compartment model, which is in turn above the curve for the one-compartment model.

However the value range of each kinetic factor for the March data set is below the one calculated in October, which means that irrespective of the types of model, the kinetic value changes depending on the period of the year. This is probably due to the quality of the water.

5.6 FORWARD MODELLING

The forward model predicts the outlet concentration of chlorine, using the measurement of inlet and outlet flow rates, the inlet concentration of chlorine, and the level as observed values (Figure 5-10). For each model, a constant value for the kinetic factor (obtained from the identification model solution) is fixed as an observed variable, with a small observation error.

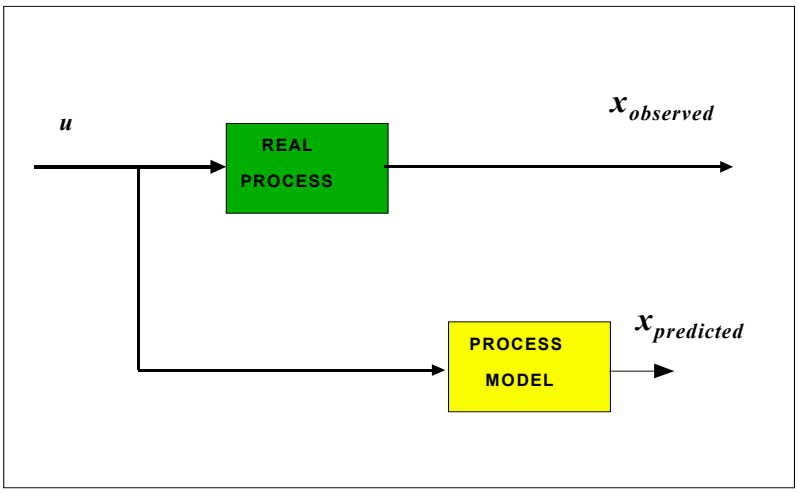


Figure 5-10: Real-time prediction using the same inputs as the process

For each model u is the vector of observed data (inlet and outlet flow rates, level, inlet chlorine concentration, kinetic factor) and x is the outlet chlorine concentration. The real process represents the plant and the process model is the Kalman filter. So the observed and predicted outlet data can be compared. If the model is good the predicted data should follow the observed data.

5.6.1 Tuning of the extended Kalman filter

5.6.1.1 The ε factor (measurement sensitivity)

This factor is the ratio between Q and R terms to force the EKF to follow more or less the observations (Equation 5.52). For each compartment model, five different values are studied for ε: 0.0001, 0.1, 1, 10, 1000 for the March data set. The values of the kinetic factor k are for the one-compartment model; the six-compartment model and the four-compartment model are 0.2 d^{-1}, 0.45 d^{-1} and 0.55 d^{-1} respectively.

- **Six-compartment model**

For this model, just ε equal to 0.1, 1, and 10 give results. For 0.0001 and 1000, the computation fails (Figure 5-11).

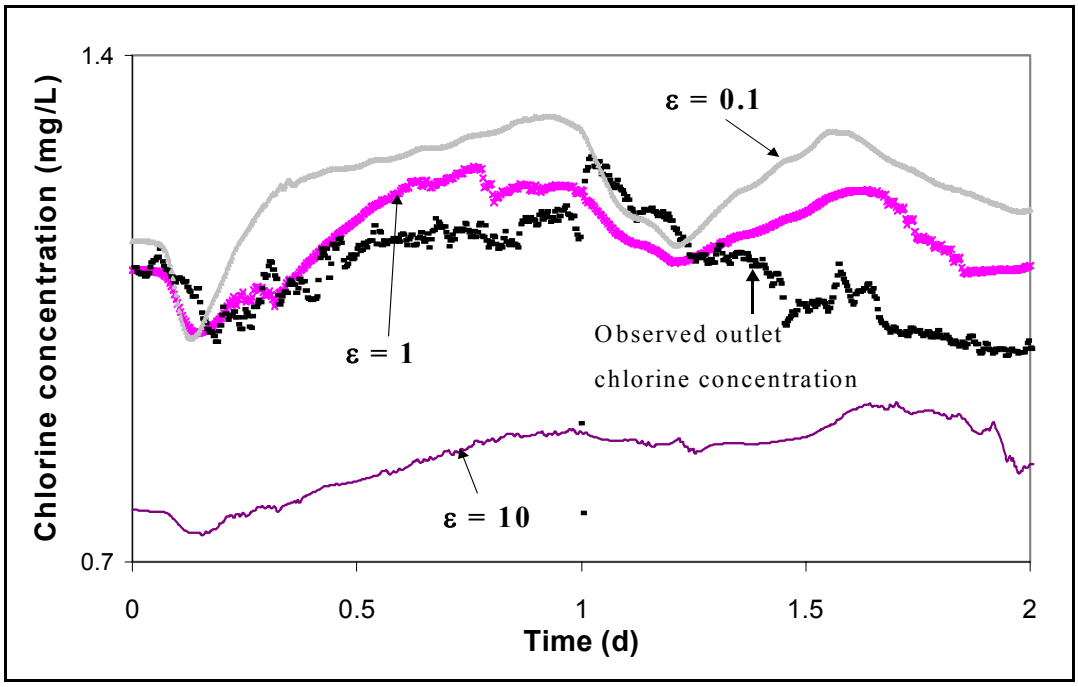


Figure 5-11: Observed and predicted outlet chlorine concentration with different values of the ε factor for the six-compartment model (k = 0.2 d^{-1})

The simulation with ε (measurement sensitivity) equal to 1 gives good results, which means that the fractional error squared expected in the model must be of the same order as the fractional error squared expected in the measurement. If one fractional error is greater than the other one, the predicted outlet chlorine concentration does not follow the trend of the observed outlet chlorine concentration as well.

- **Four-compartment model**

For this model, the simulation with ε (sensitivity measurement) values equal to 10 and 1000 gives exactly the same results (Figure 5-12).

It is noted that the four-compartment is simpler than the six-compartment model, so it converges more quickly (its computational time is smaller than the six-compartment model computational time) and it is more stable than the six-compartment model. It gives good results with a measurement sensitivity factor ε equal to 1, but the closest predicted outlet chlorine concentration is with a measurement sensitivity factor ε equal to 10.

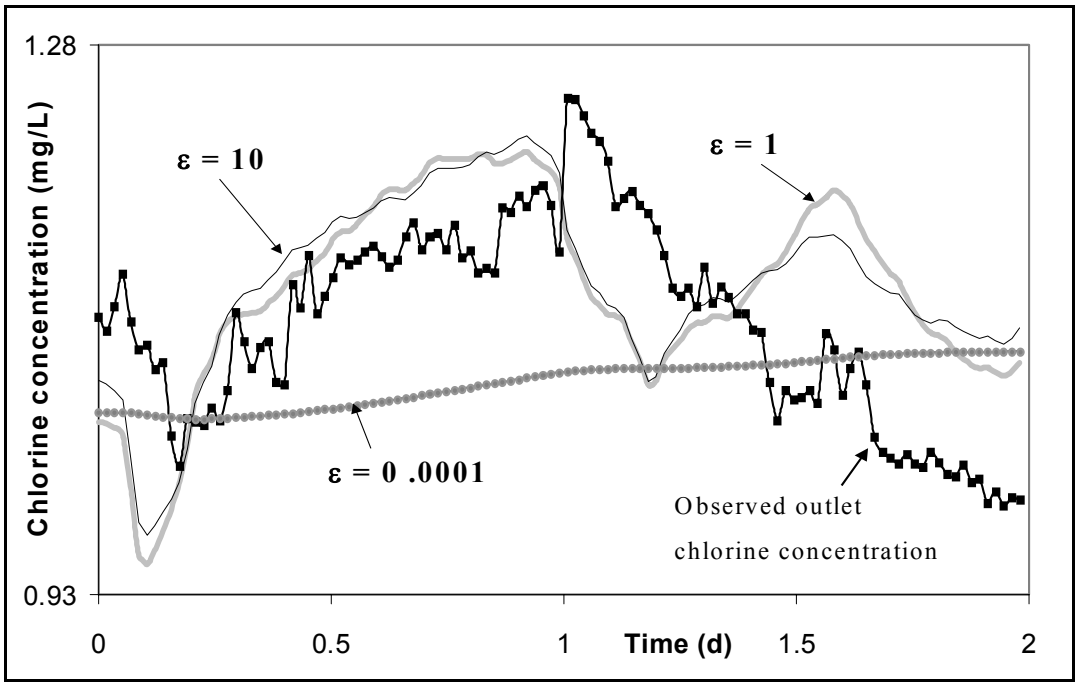


Figure 5-12: Observed and predicted outlet chlorine concentration with different values of ε factor for the four-compartment model (k = 0.45 d^{-1})

- **One-compartment model**

For this compartment model as well, the values for ε factor between 0.001 and 10^{10} give exactly the same results, so the EKF is very stable, which was expected with the one-compartment model (Figure 5-13).

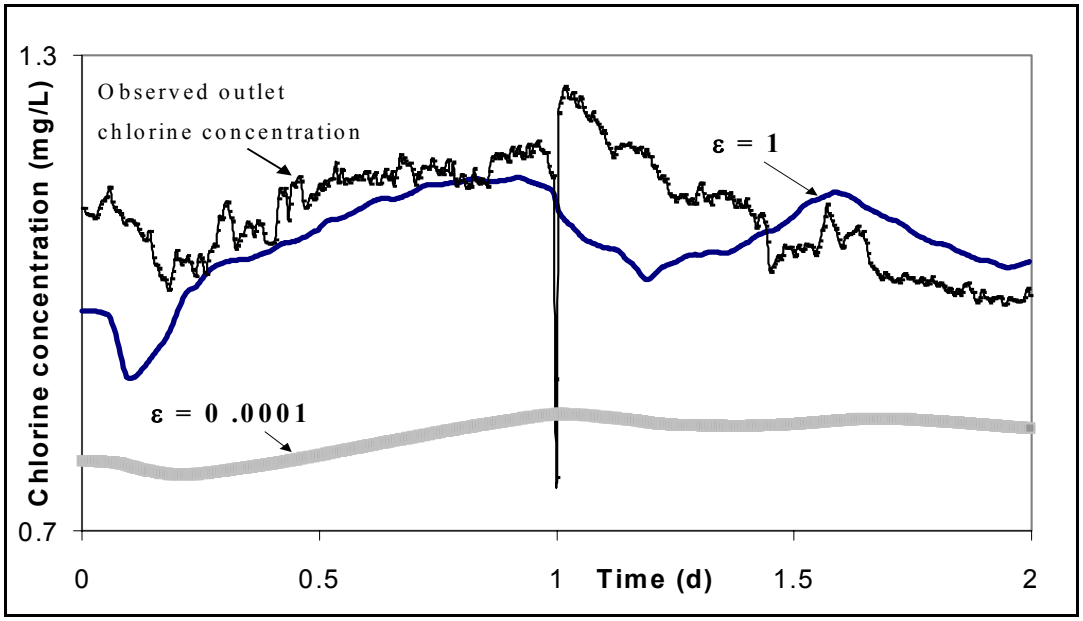


Figure 5-13: Observed and predicted outlet chlorine concentration with different values of ε factor for the one-compartment model (k = 0.55 d^{-1})

The simulation with the one-compartment model gives good results, but not as good as the two other models above.

For the following work, the ε factor (measurement sensitivity) will be taken equal to 1, because the six-compartment model gives good results for this value, though it does become unstable with a higher value. Since the 6-compartment model is the most accurate, the work below continues with it, to illustrate the tuning of the EKF.

5.6.1.2 Initialisation of the covariance matrix *M*

By experience, a small value is set for the initialisation of the filter covariance matrix *M* (0.01), thus this forces the filter to follow the observed variables. Figure 5-14 illustrates the results for the six-compartment model with two values for the initialisation of the matrix *M*: 0.01 and 1.

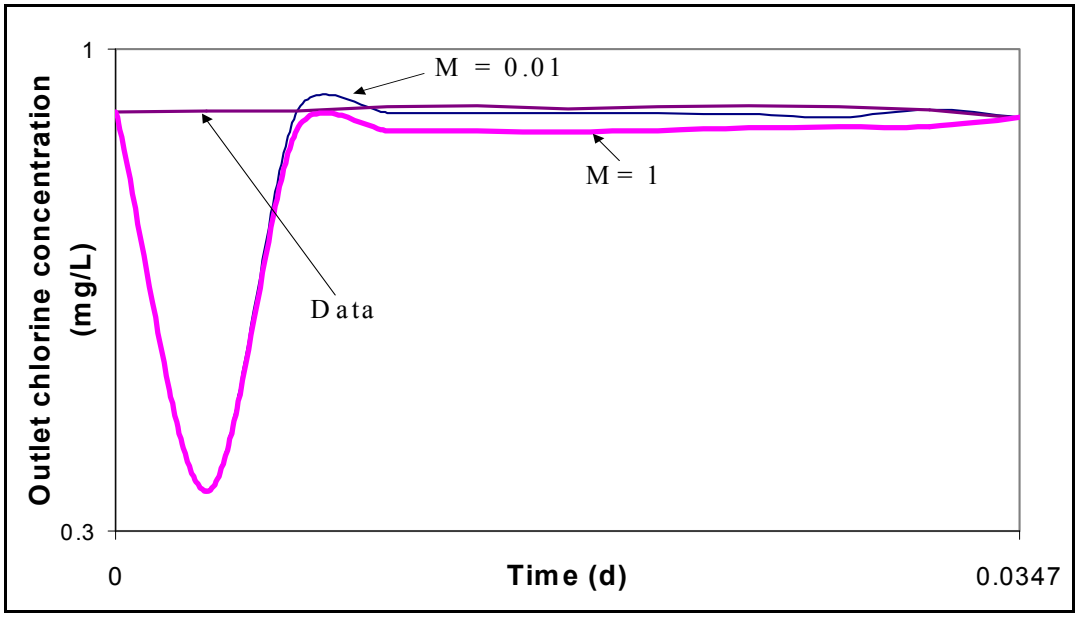


Figure 5-14: Predicted outlet chlorine concentration with different initialisation values for the covariance matrix *M* (six-compartment model)

It is seen that the predicted outlet chlorine concentration curve for the initialisation of the covariance matrix *M* equal to 0.01 converges faster to the observed outlet chlorine concentration curve than for the covariance matrix *M* initialised to 1. Thus, the EKF is tuned, with a ε factor (measurement sensitivity) equal to 1 and the initialisation value of the matrix *M* equal to 0.01.

5.6.2 Comparison amongst three models

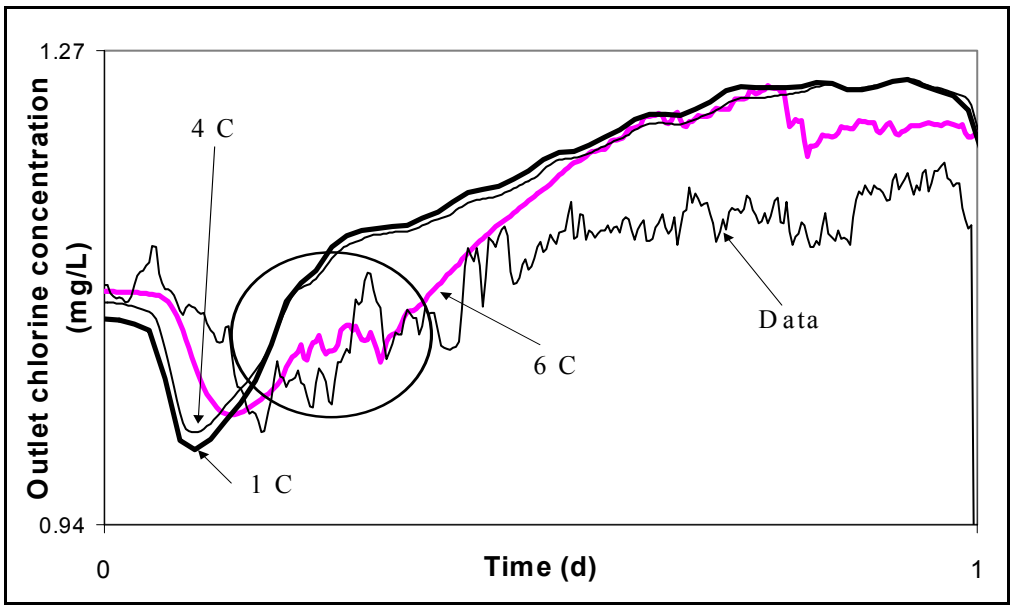


Figure 5-15: Comparison of the predicted outlet chlorine concentration between the three models

The three models have the ability to predict quite closely the outlet chlorine concentration (Figure 5-15). The six-compartment model (C = 6) is the best one, considering that it predicts the detailed variation of the observed outlet chlorine concentration (circled in Figure 5-15). On the other hand, because it is the simplest one, the one-compartment model is more robust than the others. The one-compartment algorithm will be used on-line. This model has been chosen for its simplicity and its ability to adequately predict the outlet chlorine concentration.

CHAPTER 6

ADAPTIVE CONTROL

In this Chapter, the Dynamic Matrix Control theory is applied to the compartment models in order to control the outlet chlorine concentration of the reservoir. The tuning of the controller and the results of the closed-loop are presented for the three compartment models.

6.1 DYNAMIC MATRIX CONTROL THEORY

6.1.1 Definition

Model Predictive Control refers to a class of control algorithms in which a dynamic model with its associated uncertainties is used to predict and optimise process performance. Control design methods based on the MPC concept have found wide acceptance in industrial applications because of their ability to handle process interactions and unusual dynamic responses, and because they do not necessarily demand a rigorous model derived from first principles.

The DMC model algorithm is currently one of the most popular and widely used MPC algorithms, because it is simple, intuitive and allows the formulation of prediction vector in a natural way. It is based on a linearised step response model, the *convolution model*, for prediction of the effect of possible control actions.

6.1.2 Theory

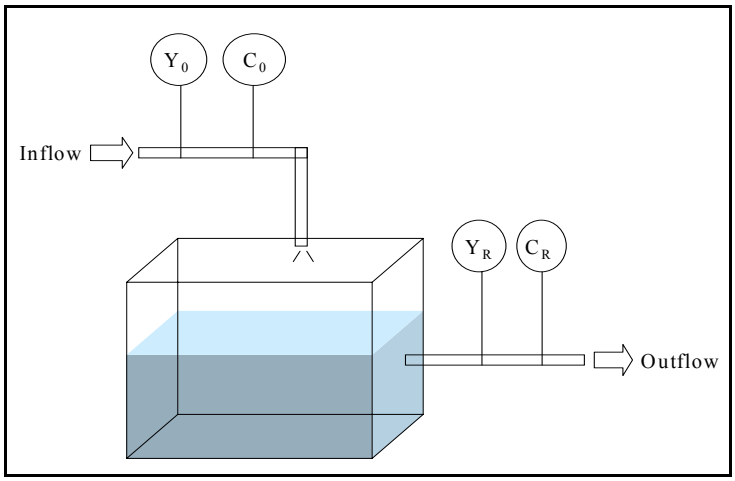


Figure 6-1: Chlorine contact tank with the outlet microbe concentration and the outlet chlorine concentration determined by the inlet microbe concentration and the inlet chlorine concentration

Figure 6-1 illustrates an example of a 2-input, 2-output system: the chlorine contact tank in which inlet microbe concentration (Y_0) and inlet chlorine concentration (C_0) cause variations in the outlet microbe concentration (Y_R) and outlet chlorine concentration (C_R). Considering that the system is steady, if a step is made in C_0, two separate responses for C_R and Y_R are expected. Likewise, distinct responses for C_R and Y_R for a step in Y_0 would be expected. Figure 6-2 shows graphically the responses of C_R and Y_R (from their original steady values) for unit positive steps in C_0 and Y_0.

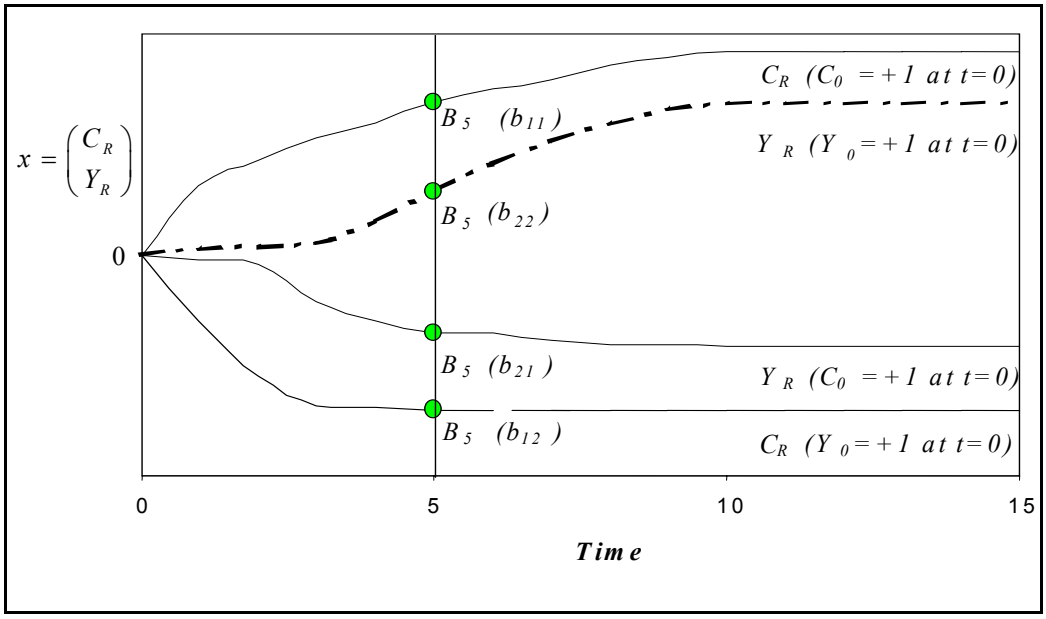


Figure 6-2: Step responses for a 2-input 2-output system

Mulholland (2001) explained that for the input vector m (C_0, Y_0), if it is now considered not just one step but a series of control vector moves Δm_1, Δm_2, ..., Δm_M, over a sequence of M time steps and if the system is linear, the resultant sequence in x (C_R, Y_R) over P intervals by shifting, scaling and superposing the above step responses can be built:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_M \\ x_{M+1} \\ \vdots \\ x_P \end{pmatrix} = \begin{bmatrix} B_1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ B_2 & B_1 & 0 & 0 & 0 & \cdots & 0 \\ B_3 & B_2 & B_1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ B_M & B_{M-1} & \cdots & B_1 & 0 & \cdots & 0 \\ B_M & B_M & B_{M-1} & \cdots & B_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ B_M & B_M & B_M & B_M & B_{M-1} & \cdots & B_1 \end{bmatrix} \begin{pmatrix} \Delta m_1 \\ \Delta m_2 \\ \Delta m_3 \\ \vdots \\ \Delta m_M \\ \Delta m_{M+1} \\ \vdots \\ \Delta m_P \end{pmatrix} \quad \begin{array}{l} \text{(Steady-state response} \\ \text{achieved } M \text{ time intervals} \\ \text{ahead with } M < P \text{)} \end{array} \tag{6.1}$$

This represents the convolution model for future outputs as $x = B\Delta m$, where the "matrix of matrices" B is generally known as the "Dynamic Matrix". Then Mulholland (2001) defines the $P \times M$ matrices from step response coefficients. B_0 is the *offset* matrix of the coefficients and B_{OL} is the matrix of the *open loop* response coefficients.

$$B_{OL} = \begin{bmatrix} B_M & B_M & B_{M-1} & B_{M-2} & B_{M-3} & \cdots & B_3 & B_2 \\ B_M & B_M & B_M & B_M & B_M & \cdots & \cdots & B_3 \\ B_M & B_M & B_M & B_M & B_M & \cdots & \cdots & \vdots \\ B_M & B_M & B_M & B_M & B_M & \cdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & B_{M-1} \\ B_M & B_M & B_M & B_M & B_M & \cdots & B_M & B_M \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ B_M & B_M & B_M & B_M & B_M & \cdots & B_M & B_M \end{bmatrix}$$

$$B_0 = \begin{bmatrix} B_M & B_{M-1} & B_{M-2} & B_{M-3} & \cdots & B_2 & B_1 \\ B_M & B_{M-1} & B_{M-2} & B_{M-3} & \cdots & B_2 & B_1 \\ B_M & B_{M-1} & B_{M-2} & B_{M-3} & \cdots & B_2 & B_1 \\ B_M & B_{M-1} & B_{M-2} & B_{M-3} & \cdots & B_2 & B_1 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ B_M & B_{M-1} & B_{M-2} & B_{M-3} & \cdots & B_2 & B_1 \end{bmatrix}$$

and the present measurements (P) and past inputs (M):

$$x_{OMEAS} = \begin{pmatrix} x_{OMEAS} \\ x_{OMEAS} \\ x_{OMEAS} \\ x_{OMEAS} \\ \vdots \\ x_{OMEAS} \end{pmatrix} \quad \text{and} \quad \Delta m_{PAST} = \begin{pmatrix} \Delta m_{-M+1} \\ \Delta m_{-M+2} \\ \Delta m_{-M+3} \\ \Delta m_{-M+4} \\ \\ \Delta m_0 \end{pmatrix}$$

then the "open-loop" response, corrected for present model offset, is

$$x_{OL} = x_{OMEAS} + \left[B_{OL} - B_0 \right] \Delta m_{PAST} \tag{6.2}$$

and by including the contribution of the future control input steps Δm, the "closed loop" response up to the P-step horizon is obtained :

$$x_{CL} = x_{OL} + B\,\Delta m \tag{6.3}$$

The future open-loop response x_{OL} based on past inputs and the present output is computed on each step. Thus the control problem to reach the desired trajectory x_{CL} amounts to finding suitable Δm as in Figure 6-3.

A constrained multivariable Linear Dynamic Matrix Controller (LDMC), based on the linear programming solution of Chang and Seborg (1983), and the formulation of Morshedi *et al.* (1985), has been developed as follows.

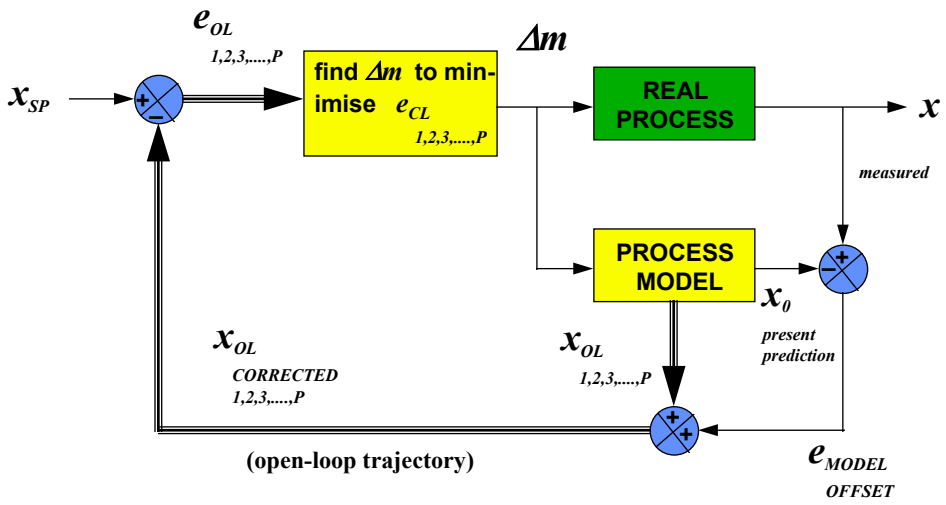


Figure 6-3: Model Predictive Control configuration (Mulholland, 2001)

If x_{SP} is defined to contain a sequence of set-points for the outputs up to the time horizon P steps ahead, so that the open loop error may be calculated in advance as $x_{OL} - x_{SP}$, the closed-loop error for a control move sequence Δm will be, using equation (6-1):

$$e_{CL} = x_{CL} - x_{SP} = e_{OL} + B\Delta m \tag{6.4}$$

Only a limited sequence of N moves (Δm^*) is generally optimised ($N << P$). Equivalently it can be set $\Delta m_k = 0$ for $k > N$, or alternately replacing B with the non-square $P \times N$ matrix:

$$A = \begin{bmatrix} B_1 & & & & 0 \\ B_2 & B_1 & & & \\ \vdots & \vdots & \ddots & & \\ B_N & B_{N-1} & & \ddots & B_1 \\ \vdots & \vdots & & & \vdots \\ B_M & B_M & & & B_{M-N+1} \\ \vdots & \vdots & & & \\ B_M & B_M & \cdots & \cdots & B_M \end{bmatrix}$$

Then

$$e_{CL} = e_{OL} + A\Delta m^* \tag{6.5}$$

A quadratic objective function is defined, dependent only on the strategy Δm^*.

$$\begin{aligned} J(\Delta m^*) &= (e_{CL})^T W (e_{CL}) + (\Delta m^*)^T L (\Delta m^*) \\ &= (e_{OL} + A\,\Delta m^*)^T W (x_{OL} - x_{SP} + A\,\Delta m^*) + (\Delta m^*)^T L (\Delta m^*) \end{aligned} \tag{6.6}$$

If J is minimised with respect to Δm^*, it is found an optimal sequence of control moves, Δm^*, which attain minimum deviation from the set-point trajectory up to the time horizon P, for minimum control move effort. The weights in the matrices W and Λ, generally diagonal, establish the extents to which deviations of either parameter are penalised. Higher values in W than Λ will generally be associated with higher "gains". The values in Λ cause *"move suppression"*. It is easily shown that differentiation of J with respect to the elements of Δm^*, and setting the result to the zero vector, yields the *unbounded quadratic optimum* control move strategy

$$\Delta m_{UQO} = -\,[A^T W A + L]^{-1} A^T W\, e_{OL} \tag{6.7}$$

By adding the successive moves, the sequence of actual control settings is obtained:

$$m = L\,\Delta m^* + m_{INIT} \tag{6.8}$$

$$L = \begin{bmatrix} I & 0 & 0 & 0 & 0 & \cdots & 0 \\ I & I & 0 & 0 & 0 & \cdots & 0 \\ I & I & I & 0 & 0 & \cdots & 0 \\ I & I & I & I & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ I & I & I & I & I & \cdots & I \end{bmatrix}$$

As noted by Mulholland (2001), a method which will search for the minimum of J within defined constraints for both the inputs m and the outputs x requires *Quadratic Programming*, and is quite computation-intensive. In a less demanding approximation, Linear Dynamic Matrix Control (LDMC), a combination of control moves getting as close as possible to Δm_{UQO}, and staying within the constraints is sought by Morshedi *et al* (1985). This re-definition of the problem allows using *Linear Programming* to handle the constraints. Although it does not ensure the *quadratic optimum*, it is expected to be close (and identical within the constraints).

Finally, the optimal Δm solution is found, whether constrained or otherwise, including optimal values for the limited sequence of steps Δm_1, Δm_2, ... ,Δm_N. However the first step Δm is the only one actually implemented, before the entire optimisation process is repeated on the next time-step. The optimisation effect of more than one step can provoke overshooting, with subsequent steps correcting the steady-state response.

6.2 APPLICATION TO OUTLET CHLORINE CONCENTRATION CONTROL

In this work, the control model for the chlorine contact reservoir is single-input, single-output, where only the inlet chlorine concentration is varied to reach the desired outlet chlorine concentration. Actually, only the unbounded optimal input solution (equation 6.7) was computed. Sometimes, where the optimum control move Δm lay *outside* of the allowed range, it was simply "clipped" back to the maximum or minimum value.

An adaptive DMC technique is proposed, in which the dynamic matrix A, as well as B_{OL} and B_0, are updated on each time-step, thus taking into account the system non-linearity. The model simulates the chlorine decay process in real time with the same flows and level used on the plant, using the collected data.

Following Lacave (2001), the model is used to produce updated step responses in real time. P separate solutions are maintained in parallel with the main real-time representation of the process, all based on an offset input signal, continuously updating the local step response. These solutions have the same time-varying inputs as the actual process model, except that the manipulated variable (C_0) is given a fixed offset (in this case, +0.1 mg/L). At each control step, one of the solutions (in sequence) is reset to the present modelled output. Between this time and the next "resetting" of this particular solution, its output curve diverge gradually from the model output curve, on account of the +0.1 mg/L offset in input. The various stages of deviation of each solution are used to rebuild a local step-response. The resultant step-response is then set into the DMC, which is used for a linear convolution model prediction of the future output. The

linearity of the model eases a direct calculation of the required control actions to track the desired set-point curve optimally. The continuous adjustment of the dynamic matrix guarantees adaptation with regard to non-linearity in the original complex process model (Figure 6-4).

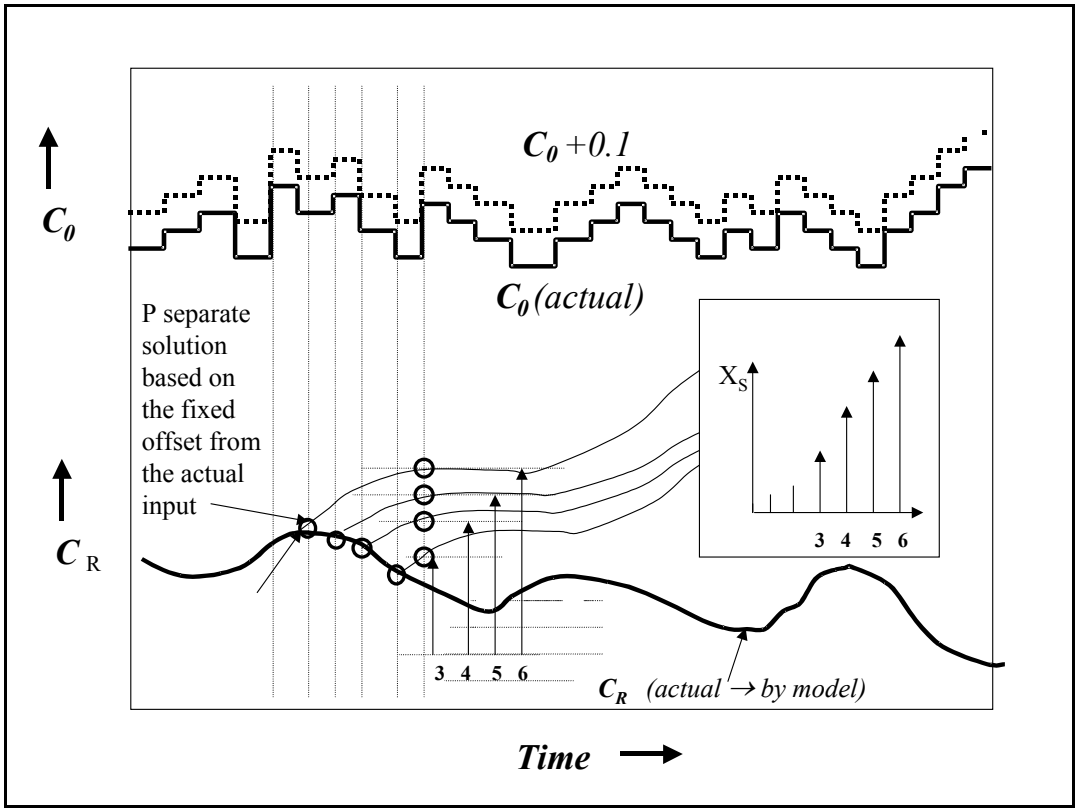


Figure 6-4: Continuously updated step response from parallel solutions of real-time model

6.3 PROGRAMMING

The controller module is written at the end of the extended Kalman filter program (see Appendix C). The table (Appendix B) translates the symbols used in the theory presented in section 6.1.2 to their equivalents in the program. Figure 6-5 is a simplified flow diagram of the program.

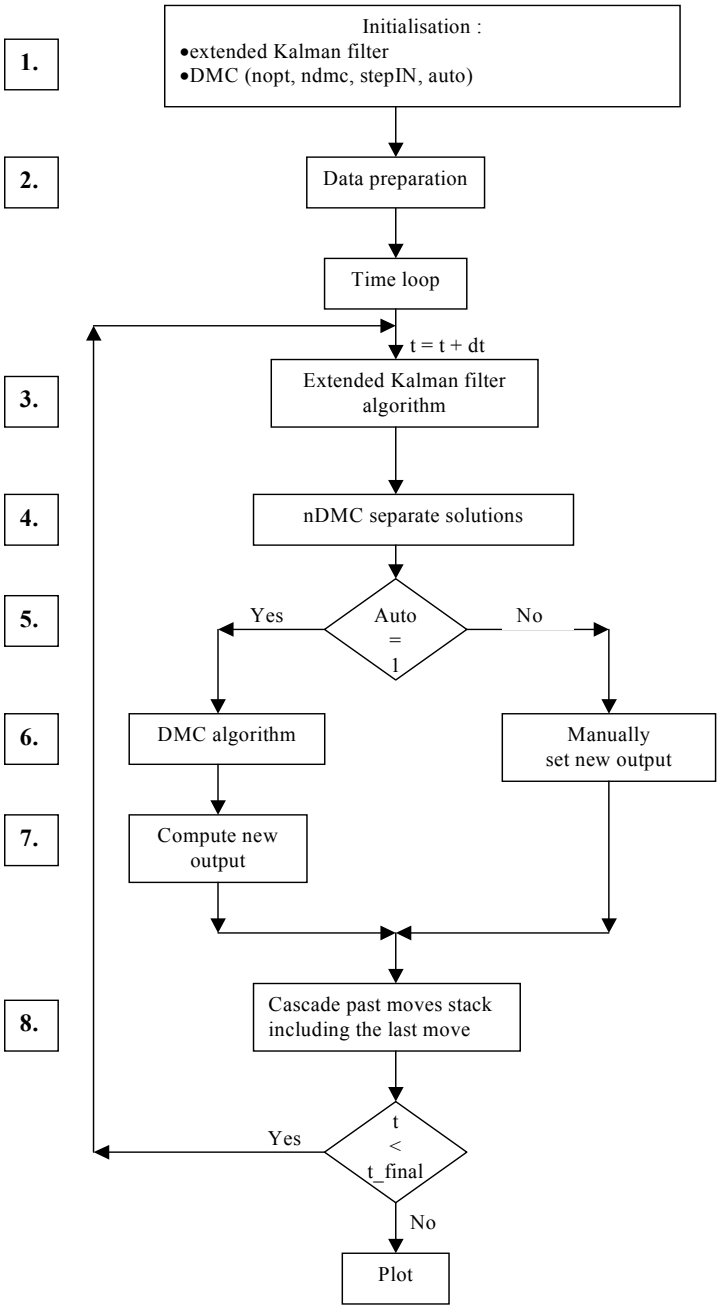


Figure 6-5: Simplified flow diagram of the program including the EKF and DMC algorithms

6.3.1 Initialisation (Step 1)

The DMC algorithm needs values for the number of optimisation steps to the horizon *P* (cf. Page 6-2 to 6-4)). The small fixed offset added to the inlet chlorine concentration (cf. Pages 6-6 and 6-7) is set to + 0.1 mg/L and the number of the control moves is set to 1.

6.3.2 Steps 2 and 3

The data preparation and EKF algorithm are explained in the previous chapter (Pages 5-12 and 5-13).

6.3.3 DMC step responses (Step 4)

P separate solutions of the model are solved simultaneously. These all have an input offset by + 0.1 mg/L from the actual input. Each solution output is reset in turn to a solution without the offset. Thus the sequence of solution represents successive deviations to an input step of + 0.1 mg/L.

In these tests, the DMC is based on the same model as that used to simulate the process (four, six and one-compartment models. The three controllers were compared, though, of course, they were controlling three different models

λ and the *W* matrices were adjusted to obtain best controller performance (Equation 6.6).

6.3.4 Control (Step 5 to 8)

The DMC needs Δm_{PAST} vector to be updated at each step. This vector contains history of what previous control moves, allowing the DMC to calculate the open loop trajectory. If the DMC is disabled, Δm_{PAST} vector is updated in the same way, according to any manual settings of the inlet chlorine concentration, to enable a correct open-loop prediction once the DMC controller is enabled. The DMC is developed in the MATLAB language (cf. Appendix B and C).

6.4 RESULTS AND DISCUSSION

The off-line control tests were done by controlling the extended Kalman filter model. In the first part of the program, the extended Kalman filter algorithm predicts the outlet chlorine concentration using the input vector (inlet chlorine concentration). Then, in the second part, the DMC algorithm forces the predicted outlet chlorine concentration to reach the set-point by calculating a new inlet chlorine concentration, which is considered in the following step as a new input for the extended Kalman filter algorithm. The control will be considered for each of the compartment models: each controller will control its own specific model.

6.4.1 Tuning of the DMC

Different values are studied for *W* (the penalty weight on squared deviation from the set-point of the outlet chlorine concentration), with the λ factor (which is the penalty weight on the control move of the inlet chlorine concentration) nominally fixed to the 1, since in the case of just 1-input, 1-output, it is only the ratio W/λ that matters. The number of optimisation steps to the

"steady-state" horizon is P=20 with an interval of 0.0035d giving an horizon of 1.7h, though full responses take up to 4h with the reservoir volumes and flows that apply here. For the first day, the DMC is disabled, and the observed large fluctuations in the inlet chlorine concentration are those occurring on the plant, arising largely from filter-bed charging and discharging. Then, the DMC is enabled with the set-point for the outlet chlorine concentration set to 1, and thereafter the inlet chlorine setting is that provided by the controller.

6.4.1.1 Test with W matrix weights equal to 1

The six-compartment controller does not give as good results, because of its increased complexity. Indeed, we try to control models with different complexity, so the six-compartment model will be the hardest one to control. However this does not mean that the six-compartment control algorithm will be the worst on the plant, because this model might be the best representation of the plant. The two other compartment models give similar results (Figure 6-6).

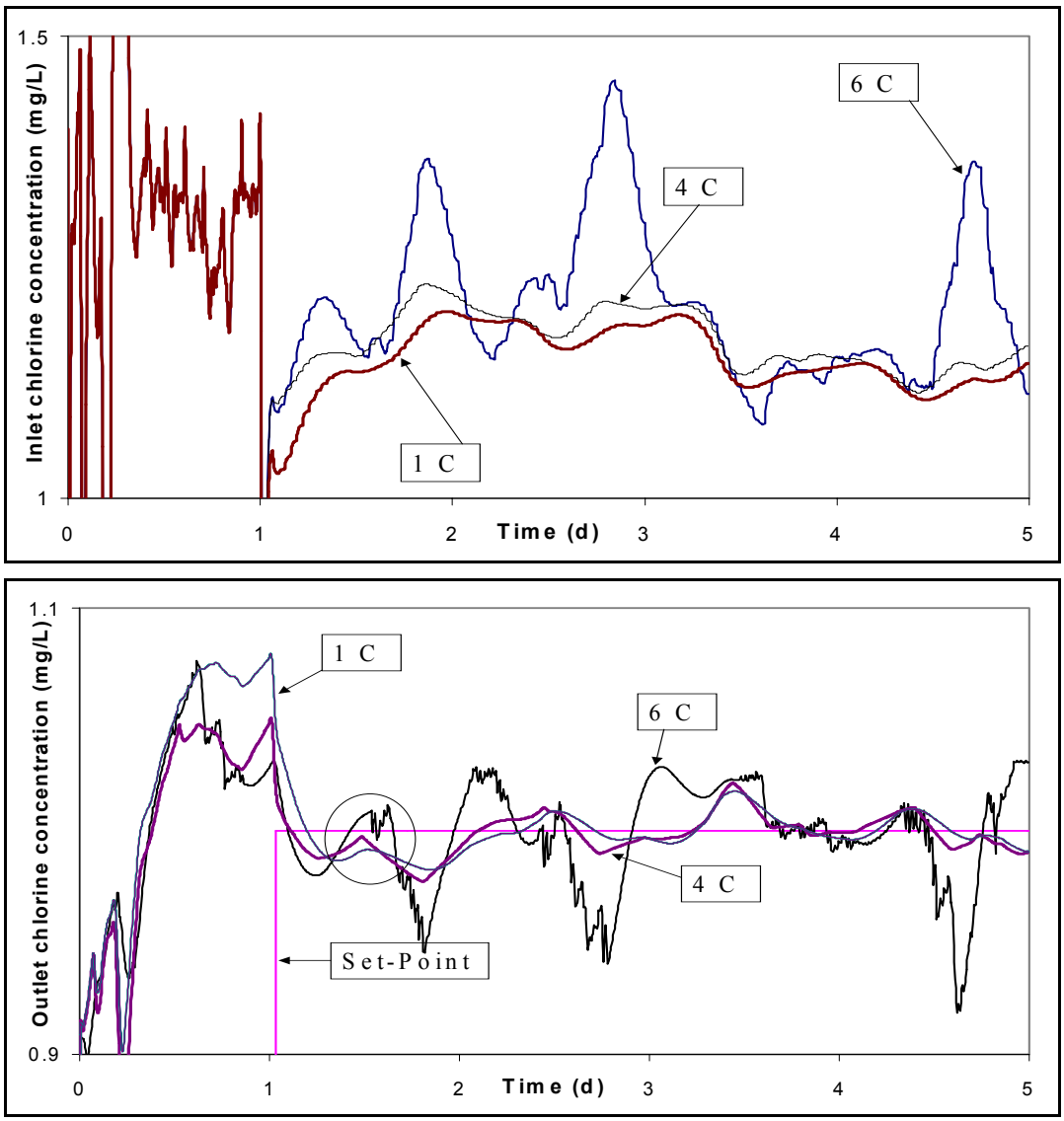


Figure 6-6: Comparison between the three models (*W*=1)

6.4.1.2 Test with *W* matrix weights equal to 100

By increasing the value of the *W* matrix weights, stronger moves are expected but the stability may reduce. In Figure 6-7 (W=100) it can be noticed that, at the beginning of the control, the six-compartment model gives smoother results and the outlet chlorine concentration curve fluctuates less than with *W* matrix equal to 1 (circled).

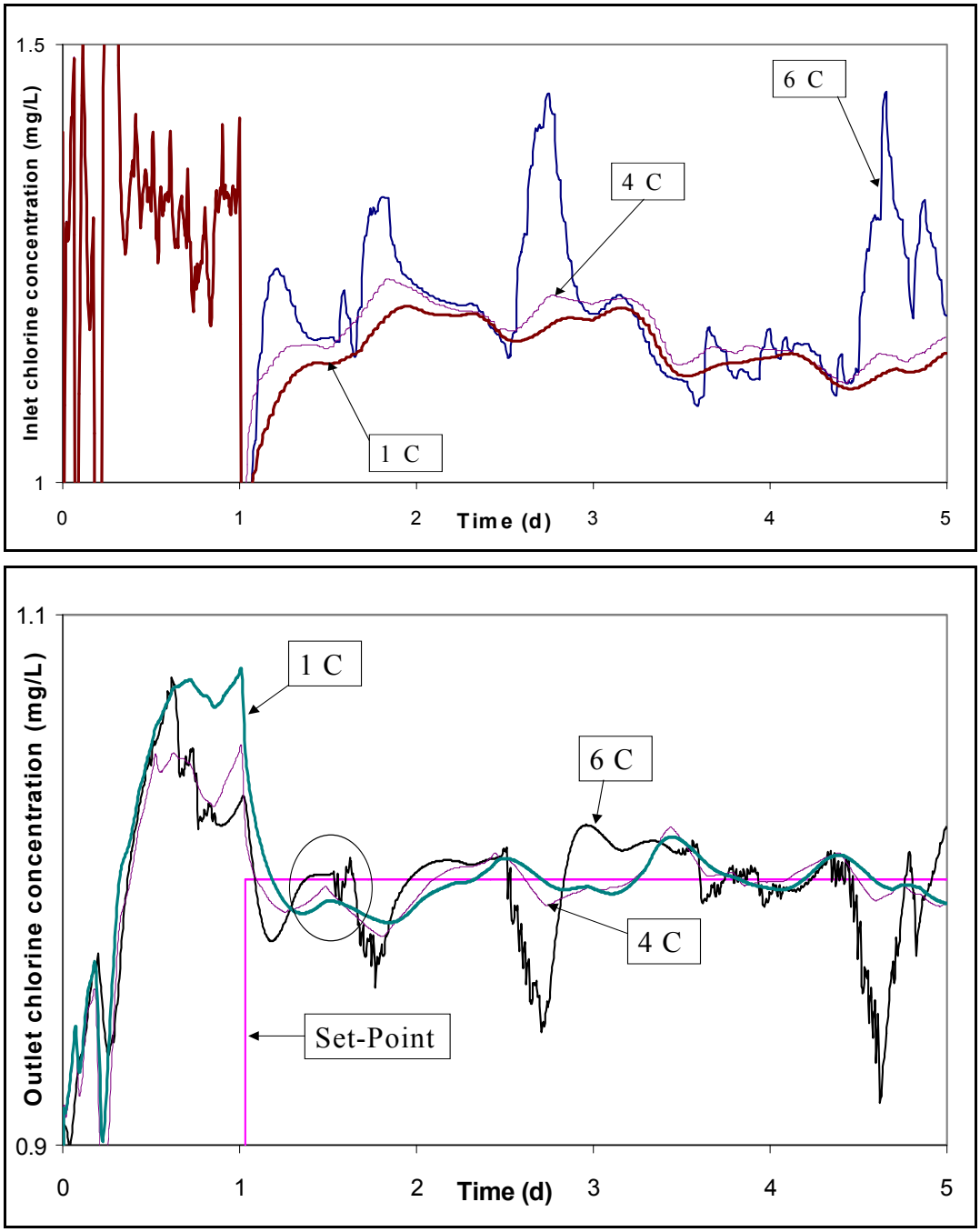


Figure 6-7: Comparison between the three models (*W*=100)

6.4.1.3 Test with *W* matrix weights equal to 10 000

The results given with *W*=10 000 are similar to those given with *W*= 100. If the penalty weight is increased above 10 000, the closed loop for the six-compartment model becomes oscillatory and less stable, eventually becoming a limit cycle with bang-bang control action (Figure 6-8).

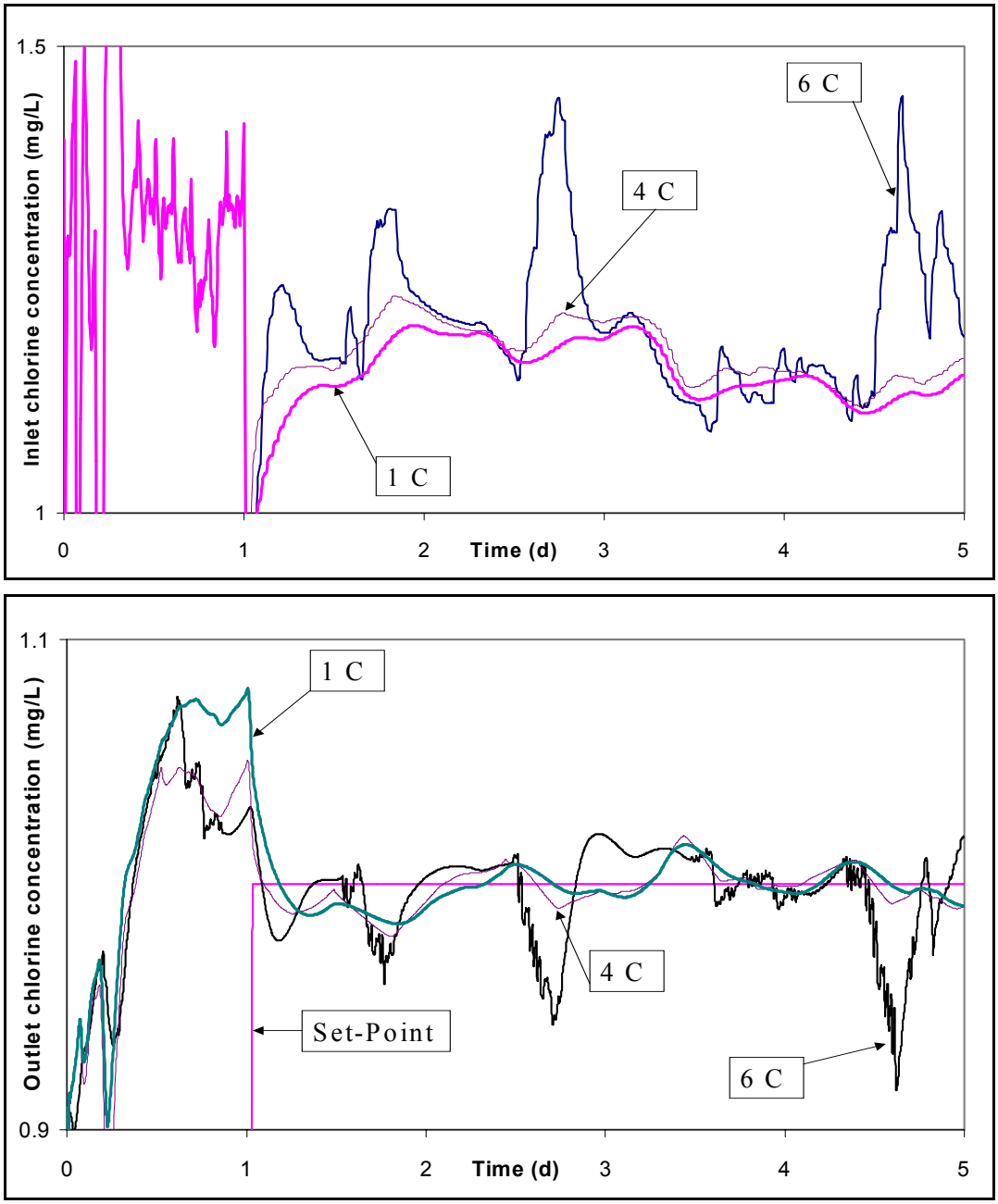


Figure 6-8: Comparison between the three models (W= 10 000)

6.4.2 Optimisation horizon (P)

The Figures 6-9, 6-10 and 6-11 illustrate for the three models (one, four, six respectively) control with a short (P = 10) and a long (P = 20) optimisation steps to the horizon. The time interval is equal to 0.0035 d. This means that the optimal horizons are respectively equal to 0.035 days (0.84h) and 0.07 days (1.76h) (10 x 0.0035 and 20 x 0.0035).

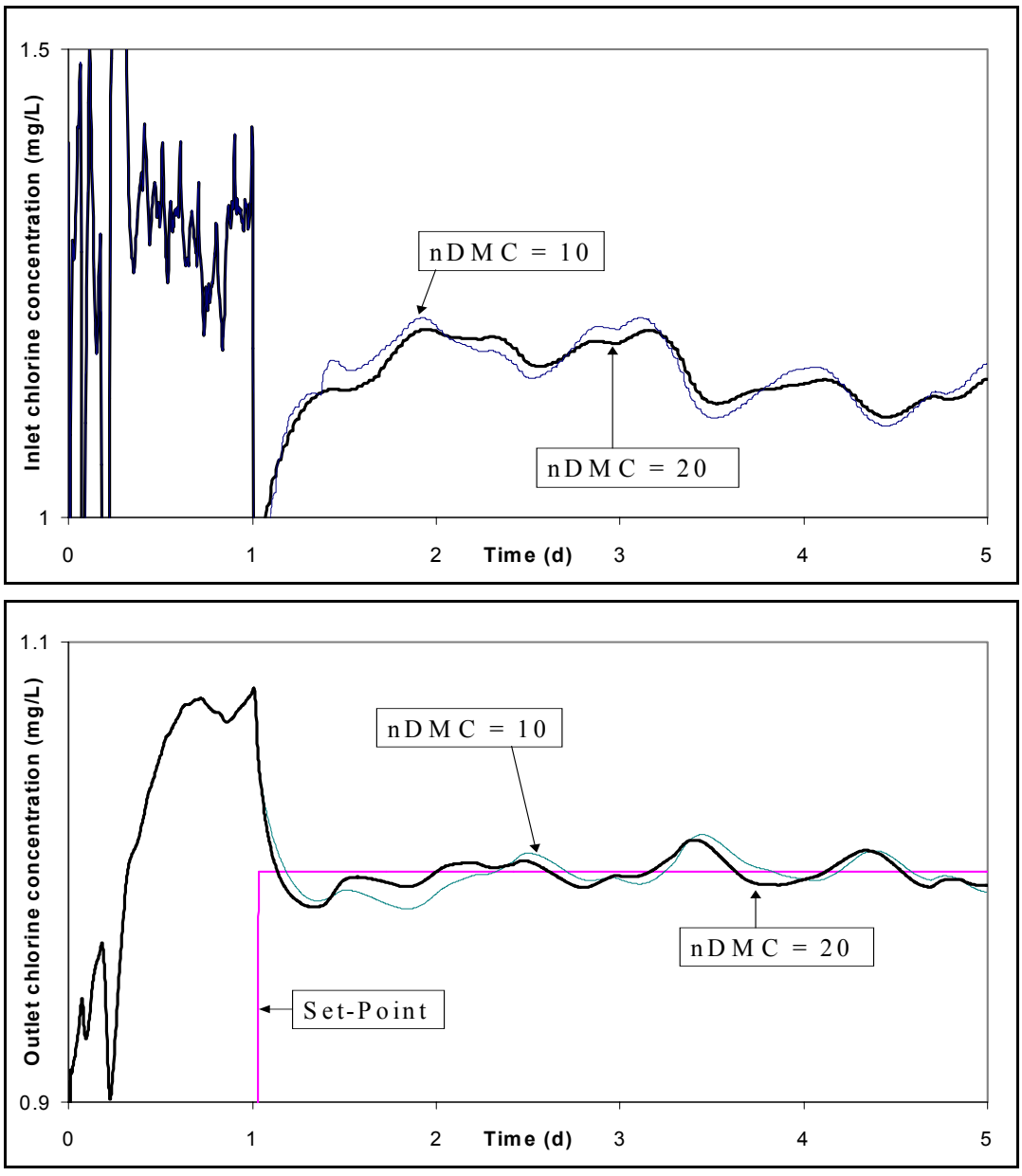


Figure 6-9: Control of one-compartment model with 0.035 d and 0.07 d optimisation horizons

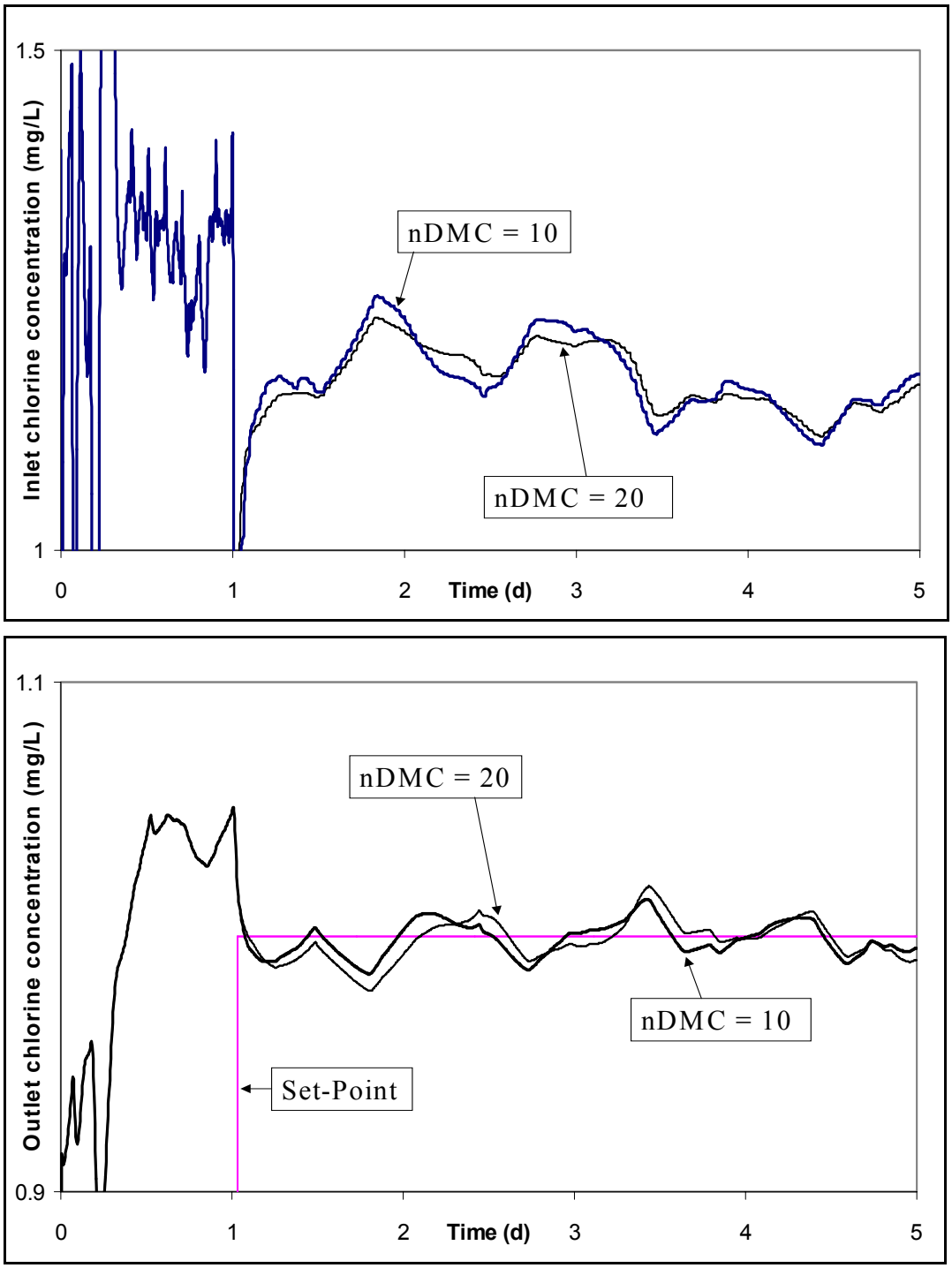


Figure 6-10: Control of four-compartment model with 0.035 d and 0.07 d optimisation horizons

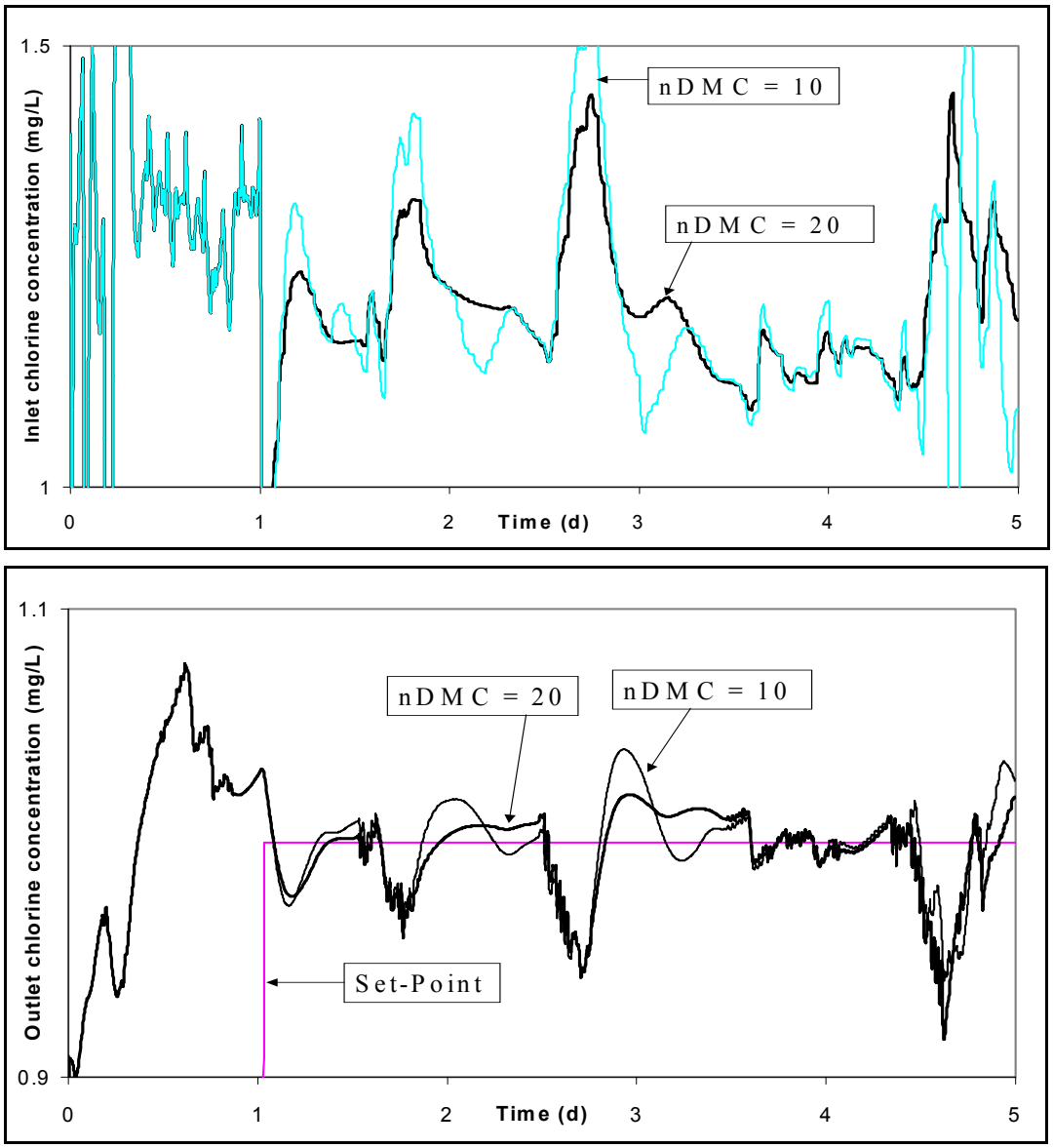


Figure 6-11: Control of six-compartment model with 0.035 d and 0.07 d optimisation horizons

For the six-compartment model particularly, it is seen that the control with 10 optimisation steps to the horizon is worse than the control with 20 optimisation steps to the horizon. With more than 20, there is no improvement in the control to warrant the increased computation.

The one-compartment model gives an adequate open-loop simulation in the previous chapter (Figure 5-13), and as the control gives good results, this model has been chosen to be implemented on-line. It has thus not been considered worthwhile attempting to implement the more accurate 6-compartment model, with its inherent computational demands and risks. Moreover, the on-line implementation within the SCADA system needs something simple and robust.

CHAPTER 7

ON-LINE IMPLEMENTATION

Isermann (1982) explained that adaptive control systems adapt their behaviour to the changing properties of controlled processes and their signals. In the feedback adaptation (closed loop adaptation, Figure 7-1) information on the process behaviour is gained by measuring process input and output signals, from which parameters determining the behaviour can be identified. Then, based on this information, the controller can be calculated and adapted. A second feedback path results leading to a closed loop action with the signal flow path: control loop signals-adaptation algorithm-controller-control-loop signals.

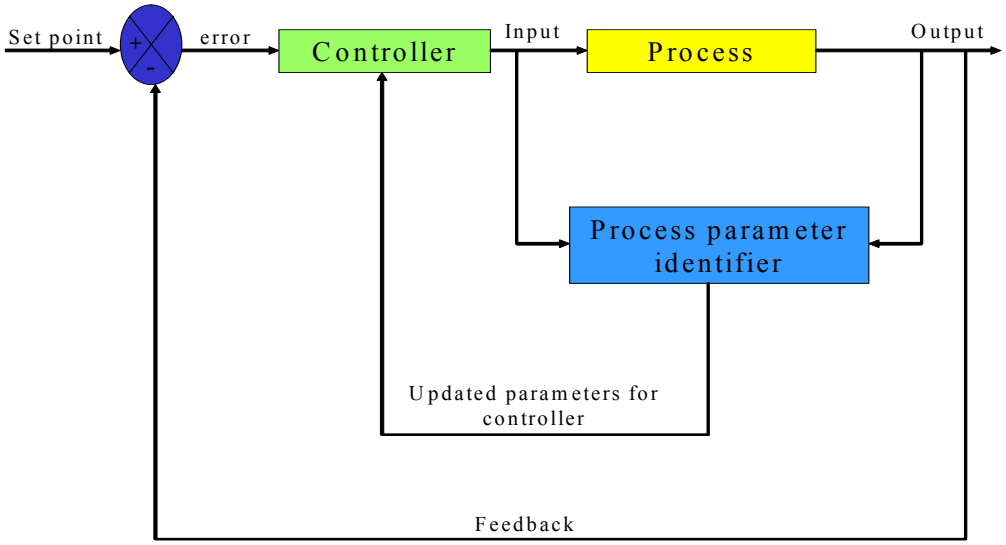


Figure 7-1: Adaptive control structure

7.1 SIMPLIFIED ONE-COMPARTMENT MODEL

The one-compartment model has been chosen to be implemented on-line. However, some modifications have been made. It was shown in section 5.5 that the kinetic factor k was not constant but depended on the volume inside the reservoir and on the period of the year. The extended Kalman filter is simplified and used to find the kinetic factor k as described in the following equations (Equation 7.1 to 7.4), using the inlet and outlet chlorine concentration, the level and the inlet flow rate of the reservoir.

Equation 5.36 can be re-written in this form:

$$\frac{dC(t)}{dt} = \frac{1}{\tau} C_0(t) - \left(\frac{1}{\tau} + k \right) C(t) \quad \text{with } \tau = \frac{\mathrm{A} \times \mathrm{h}}{\mathrm{F}_0} \tag{7.1}$$

where C_0 = inlet chlorine concentration (mg/L)

C = outlet chlorine concentration (mg/L)

t = time (d)

k = kinetic factor (d^{-1})

A = area of the reservoir (m^2)

h = water level (m)

F_0 = inlet flow rate of the reservoir ($m^3.d^{-1}$)

τ = theoretical residence time (d)

If the first order decay is not included, Equation 7.1 can be integrated:

$$C(t + \Delta t) = \left(e^{A^* \Delta t} \right) C(t) + [1 - e^{A^* \Delta t}] A^{-1} B C_0(t) \quad \text{with } A = \frac{1}{\tau} \quad \text{and } B = -\frac{1}{\tau} \tag{7.2}$$

Now, including the first order decay with a simple Euler integration, Equation 7.3 is obtained from Equation 7.2:

$$C(t + \Delta t) = A_* C(t) + B_* C_o(t) - k(t) C_t(t) \Delta t \quad \text{with } A_* = e^{A \Delta t} \quad \text{and } B_* = e^{A \Delta t} - 1 \tag{7.3}$$

Thus, the main equation of the Kalman filter (Equation 5.50) is obtained:

$$k(t + \Delta t) = k(t) + K \left[A_* C(t) + B_* C_0(t) - k(t) C(t) \Delta t - x(t + \Delta t) \right] \tag{7.4}$$

Then, in the DMC algorithm, the matrices B, B_{OL}, and B_0 (Equation 6.1 and 6.2) are dependent on the kinetic factor k, which is updated each time the extended Kalman filter runs, allowing for better control of the outlet chlorine concentration. The optimisation steps to the horizon for the DMC is set to 40 points spaced at 3 minute intervals (2h horizon). For the 2-4 hour response times observed under the typical reservoir volumes and flows that apply here, this was a compromise bearing in mind the computational load.

The prediction error matrix Q of the Kalman filter algorithm is directly linked to the observation error matrix R (Equation 5.46, 5.47 and 5.52):

$$Q = \varepsilon^* R \tag{7.5}$$

where ε is the measurement sensitivity, which will be termed as well as the gain of the extended Kalman filter.

The same is done for the penalty weight on the squared deviation from the set-point to the outlet chlorine concentration matrix *W* and the penalty weight on the control move of the inlet chlorine concentration matrix λ:

$$W = \beta * \lambda \quad (7.6)$$

where β is thus the gain of the DMC.

The MATLAB test program (Section 6.3) is translated into VISUAL BASIC, which is the mathematical language read by the Script Agent of ADROIT (Appendix E). A table in Appendix D gives the symbols used in the theory for their equivalent in the Visual Basic program (Appendix E).

The ADROIT Script Agent can be used in the off-line mode or in the on-line mode. In the off-line mode, the plant is simulated by equations using the measured plant inputs to predict the outlet chlorine concentration (if the DMC controller is enabled, it replaces input plant chlorine concentration with the DMC output). In the on-line mode, the observed outlet chlorine concentration $C_m(t)$ is filtered (to remove rapid measurement noise) using the following relation:

$$C_F(t+\Delta t) = \gamma C_m(t)+(1-\gamma)C_F(t) \quad (7.7)$$

where C_F = filtered outlet chlorine concentration (mg/L)

C_m = observed outlet chlorine concentration (mg/L)

t = time (d)

γ = smoothing factor

This smoothed chlorine concentration is then used as feedback to the DMC, and, of course, the DMC output is output to the plant if the DMC is enabled. Figure 7-2 is a simplified flow diagram of the ADROIT program. The table in Appendix D translates the symbols used in the theory (Section 6.3 and 7.1)

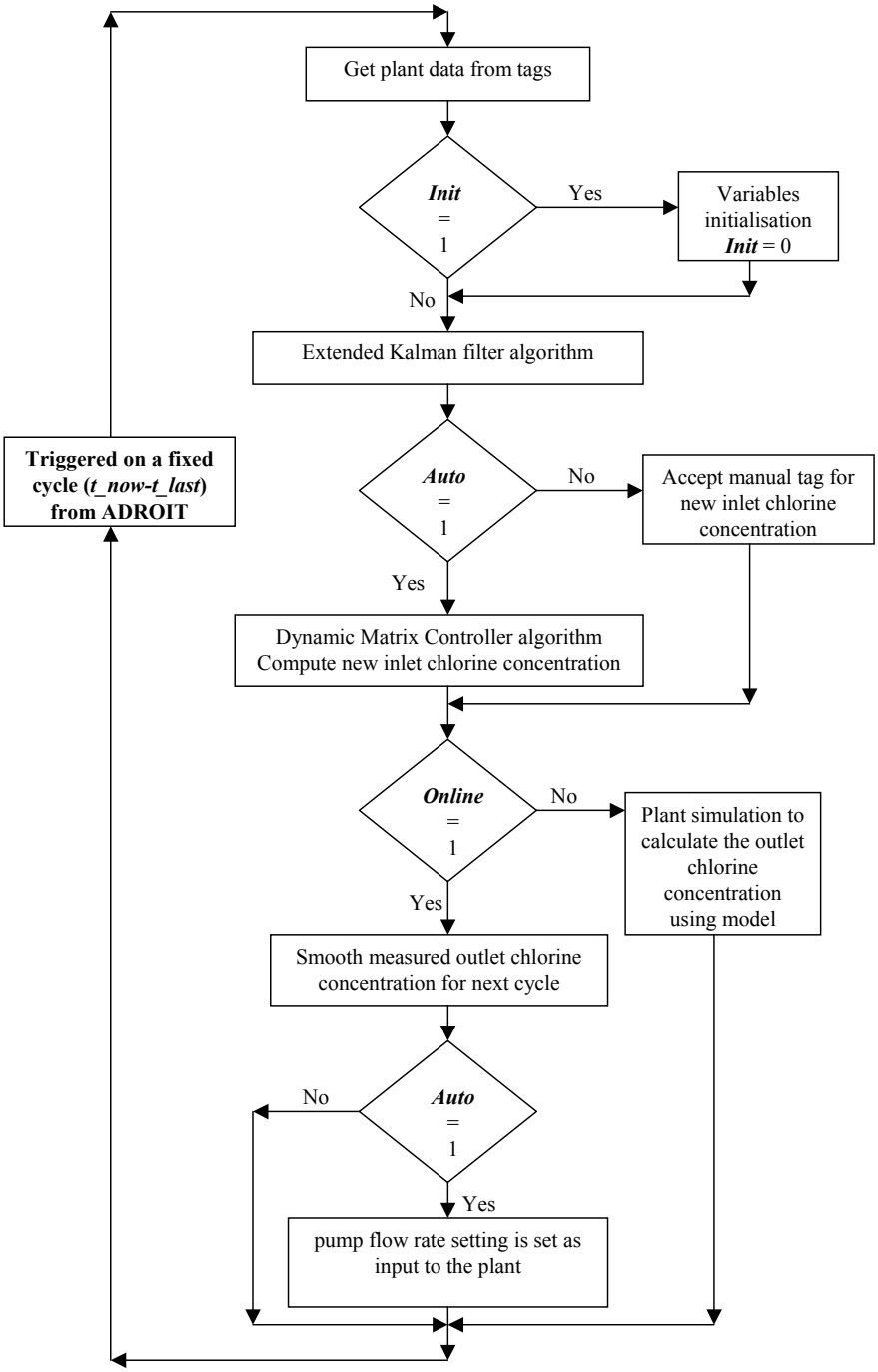


Figure 7-2: Simplified flow diagram of the ADROIT program

7.2 RESULTS

7.2.1 Adroit presentation

All the data in Adroit are called *tags* and stored in different categories (analogue, digital, script, text, etc...) in the Configuration Agent. In the Script Agent, it is possible to import the tags directly to the program, to modify them, and then to set them back into the Configuration Agent. Thus, the inlet and outlet chlorine concentration, the inlet flow rate and the level of the reservoir data are directly imported in the program, as well as the time elapsed since midnight in seconds. The time t_p between two runs of the program is set in to the Script Agent. The program is protected from wrap-around at midnight.

As it is impossible to store variables in the program itself, and some of them need to be permanently updated accounting for the previous moves (as Δm_{PAST} for example, Equation 6.2), these variables are defined as tags. It is possible to obtain and to relay them to the Configuration Agent between two runs of the program, thus allowing them to be stored. Figure 7-3 shows the User Interface of Adroit.

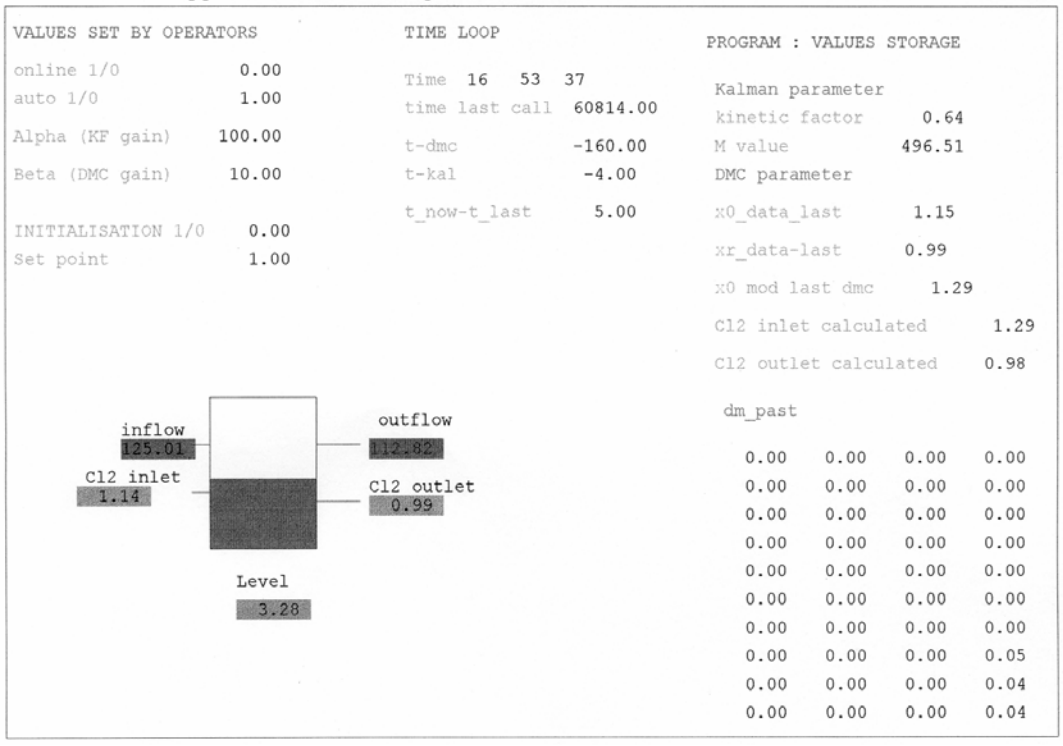


Figure 7-3: User Interface screen of ADROIT

For the off-line study, the inlet chlorine concentration value is set to 1.6 mg/L, the outlet chlorine concentration value is set to 1.2 mg/L, the level is set to 4 m and the inlet flow rate to 130 ML/d. The kinetic factor k is arbitrarily initialised to 0.65 d^{-1}. This over-specification will lead to a new kinetic factor. The smoothing factor γ, to remove the feedback outlet chlorine measurement noise, is set to 0.1.

7.2.2 Off-line mode

In preparation for the on-line controller commissioning and to experiment with tuning parameters, an off-line test was done (*online* factor is set to 0)

7.2.2.1 Time loop

In the on-line mode, the Adroit program is called every 5 seconds, to give a reasonably fast response on the screen to operator changes of the settings such as set-point or tuning parameters. Within this context the EKF counts down from 20×5 to execute on a 100-second interval, wilst the DMC counts down asynchronously from 36×5 to execute on a 180-second interval. The on-board simulation model (which replaces the actual plant in this section 7.2.2), on the other hand, executes on every 5-second call.

A *speed up* factor has been created to obtain faster response in the off-line mode. The internal time in the DMC and the extended Kalman filter are multiplied by this factor, thus a faster response is obtained. This factor has been set equal to 20 in tests.

7.2.2.2 Extended Kalman filter only

To begin, the DMC is disabled (*auto* is set to 0) and so the Kalman filter must calculate the right kinetic factor k, for the fixed inlet and outlet chlorine concentration, the level and the inlet flow rate. Depending on the value of the Kalman gain ε, the steady state is obtained more or less rapidly (Figure 7-4 and 7-5).

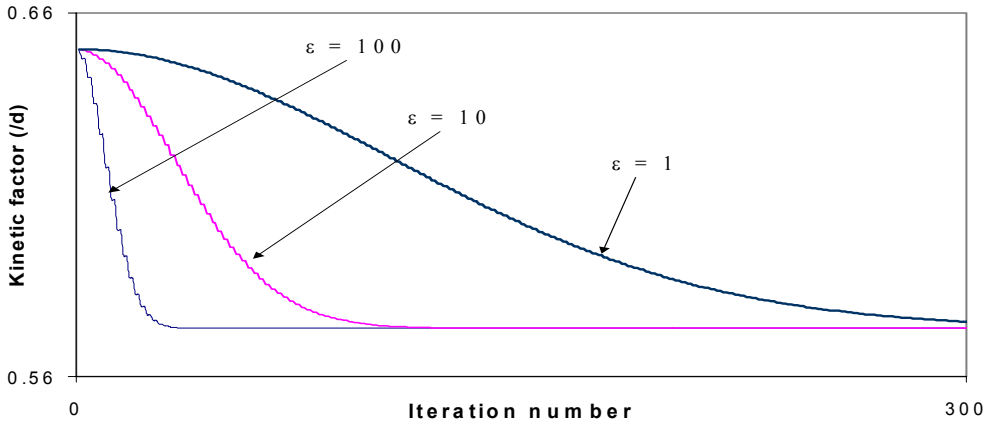


Figure 7-4: Estimation of the kinetic factor *k* with different values of the EKF

gain (ε= 1, 10, 100)

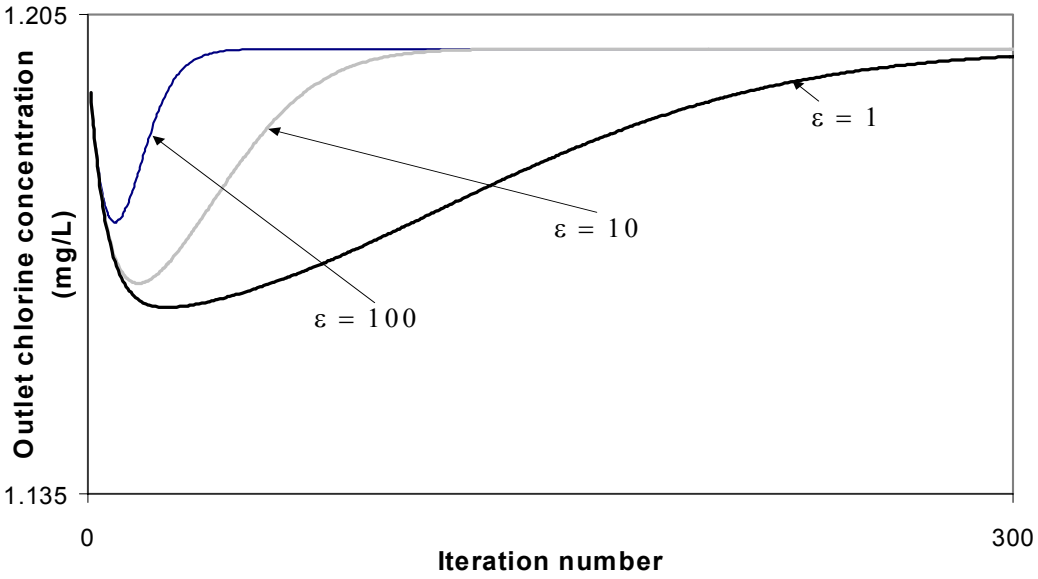


Figure 7-5: Predicted outlet chlorine concentration with different values of EKF gain (ε=1, 10, 100)

The iteration number is the number of times that the extended Kalman filter is run, meaning that the program runs 600 times (=2x300). At the beginning, the kinetic factor is initialised with a high value, so the calculated outlet chlorine concentration lies below the observed outlet chlorine concentration, then the extended Kalman filter finds the right value so that the calculated outlet chlorine concentration can be equal to the observed outlet chlorine concentration. The best result is obtained with the extended Kalman filter gain ε equal to 100. Above this value, the extended Kalman filter becomes unstable. For this simulation, the kinetic factor obtained is equal to 0.57 d^{-1}.

The Kalman filter gain ε is fixed to 100 and the inflow rate is doubled (from 130 ML/d to 260 ML/d), and the value of the kinetic factor k obtained is 1.14 d^{-1}, which was the expected result (twice the kinetic factor obtained with an inflow rate equal to 130 ML/d).

7.2.2.3 Extended Kalman filter alone then together with Dynamic Matrix Controller

After having obtained the steady state with the Kalman filter (300 iterations), the DMC is triggered with a set-point fixed to 1 mg/L at the 350th iteration. Different values for the DMC gain factor β are studied, the results are shown after the 300th iteration (run number of the extended Kalman filter) (Figure 7-6).

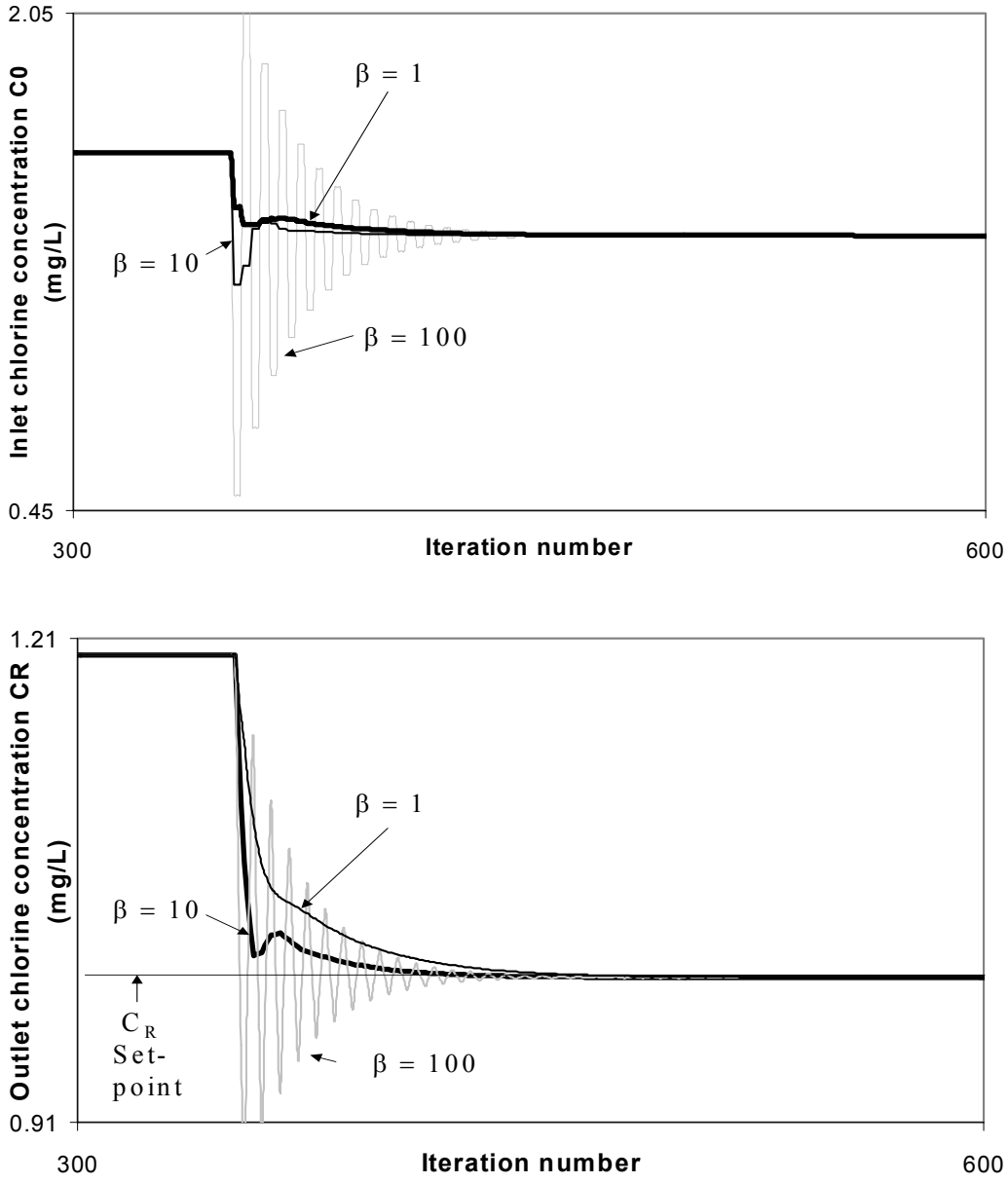


Figure 7-6: Predicted chlorine concentration with different values of the DMC gain (β = 1, 10, 100)

The best result is obtained with a DMC gain factor β equal to 10. Indeed, the results with the DMC gain β equal to 1 are smoother than the results obtained with β equal to 10 but the set-point is obtained with a greater number of iterations. It can be seen that for a large DMC gain factor, it becomes unstable.

7.2.2.4 Extended Kalman filter together with Dynamic Matrix Controller

For this study, the DMC with an outlet chlorine concentration set-point equal to 1 mg/L is triggered from the beginning. The DMC gain β is set equal to 1 for the first test and then equal

to 10 (Figure 7-7). The extended Kalman filter gain ε is set to 100 (Figure 7-8 illustrates the estimated kinetic factor, as ε is fixed, only one curve is obtained).

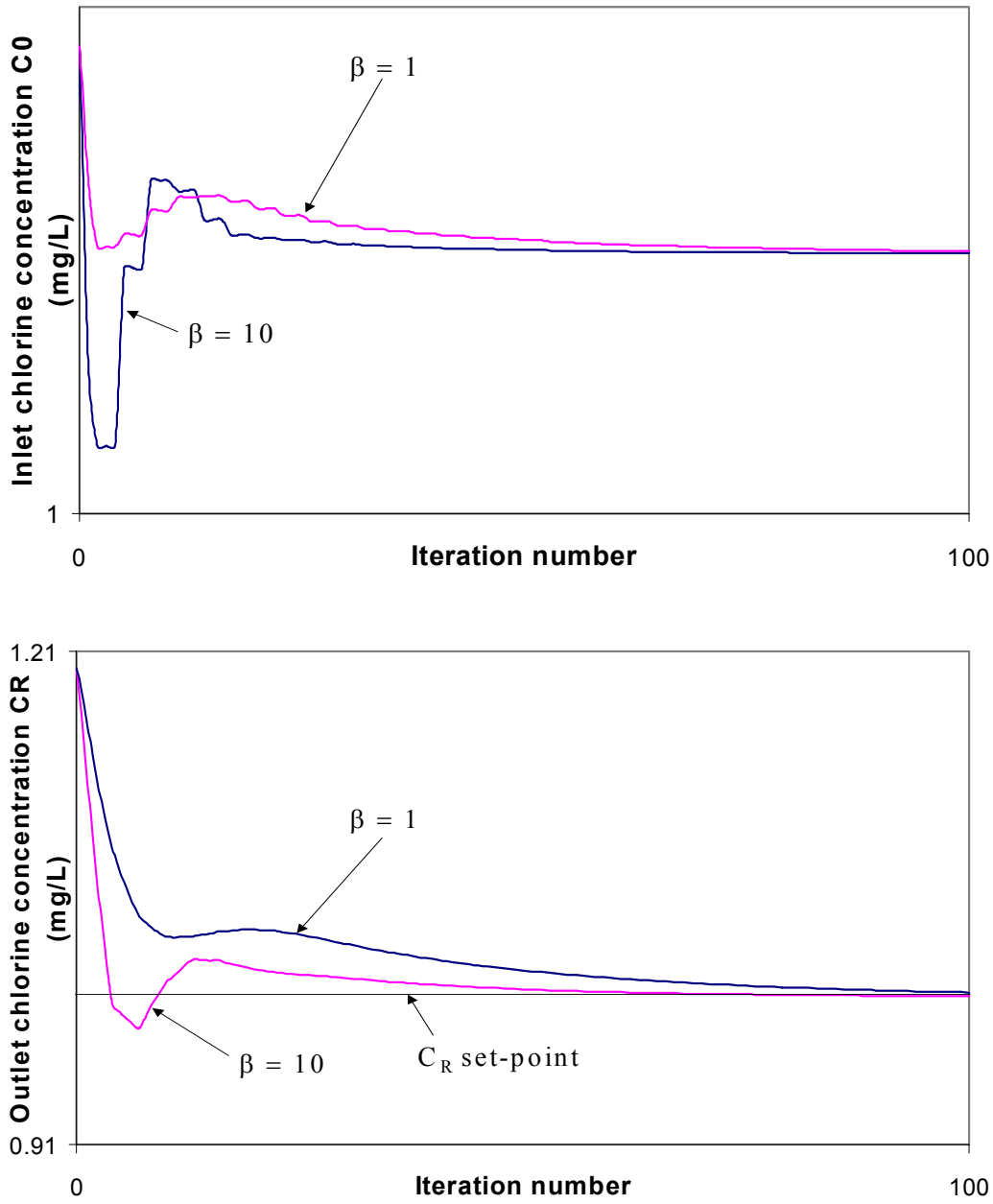


Figure 7-7: Predicted outlet chlorine concentration with different values of the DMC gain (β=1 and 10)

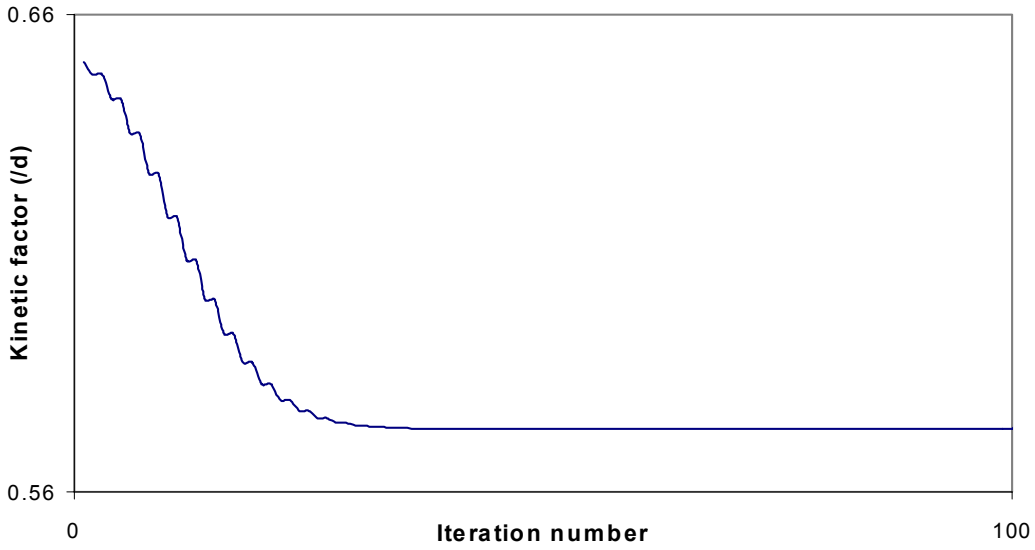


Figure 7-8: Estimation of the kinetic factor *k* (ε= 100)

With a DMC β gain equal to 10, the trajectory approaches the steady state rapidly. Though the kinetic factor requires some time to converge to the fixed k value, this variation does not degrade the DMC performance significantly.

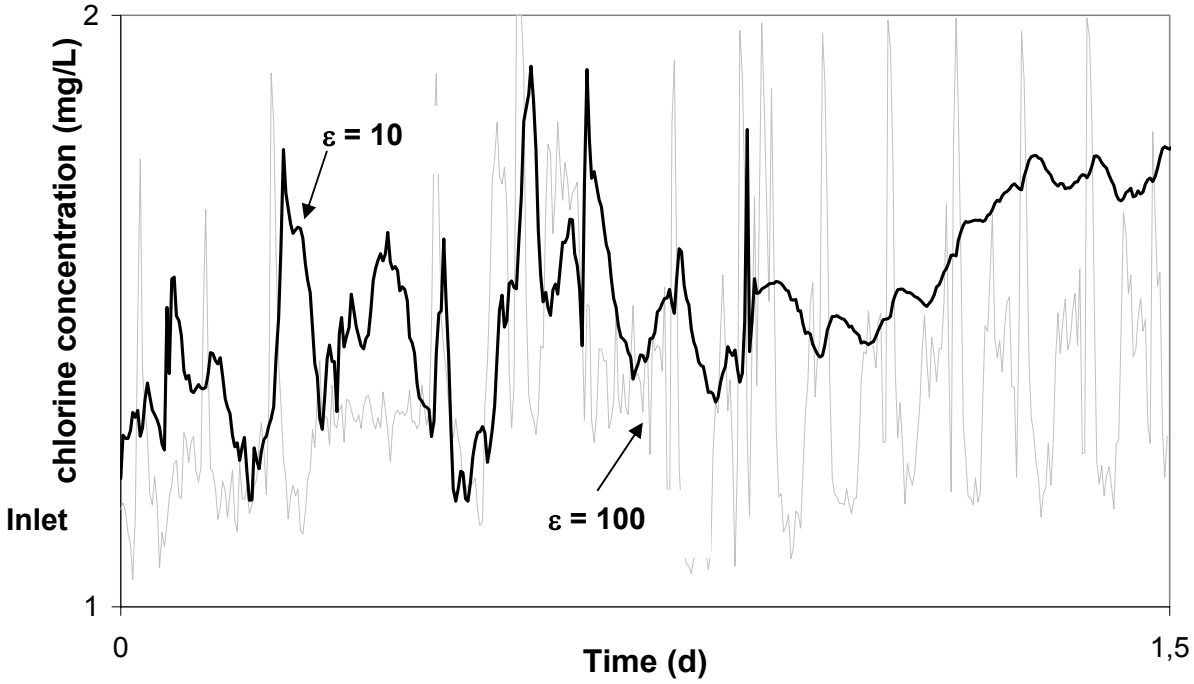


Figure 7-9: Comparison of the predicted inlet chlorine concentration with two different values of the EKF gain ε (10 for the dark curve and 100 for the light curve)

7.2.3 On-line mode

7.2.3.1 Tuning of the extended Kalman filter

Firstly, the extended Kalman filter gain ε was set to 100 and the Dynamic Matrix Controller gain β was set to 10, as determined in the off-line mode. However, the system was responding too fast (Fig. 7-9). To obtain a smoother signal, the extended filter gain ε has been decreased to 10. The Dynamic Matrix Controller gain β was kept set to 10.

7.2.3.2 Hypochlorite pump for inlet chlorine

The operators controlled the chlorine concentration in the inlet flow based on their experience. They did this by varying the pump flow rate of NaOCl (sodium hypochlorite or "hypo"), which is converted to a percentage of pump scale (Table 7-1 shows typical observed behaviour). Depending on the water inflow rate, and on the inlet chlorine concentration wanted, they set different values for the hypo pump flow rate.

Table 7-1: Pump flow rate estimation for different inlet chlorine concentrations and different water inflow rates

Inflow rate (ML/d)	Inlet chlorine concentration (mg/L)	Pump (%)
110	1.4	5
110	1.6	6.5
140	1.4	7

In order to apply the required inlet chlorine concentration calculated by the DMC algorithm, an open-loop predictor was added that sets the hypo pump flow rate to achieve the specified inlet chlorine concentration, which in this case is the manipulated variable of the DMC. It was found impossible to use the feedback measurement of the actual chlorine inlet concentration, owing to large swings in this value as water is drawn to, or discharged from, the filter beds (See Fig. 7-9).

Based on Table 7-1, the following relations have been used:

$$\text{Predicted Inlet Chlorine Concentration} = \frac{(6.15 \times \text{Pump Setting}) + 95.5}{(\text{Water Flow} + 2)} \tag{7.8}$$

$$\text{Required Pump Setting} = \frac{(\text{Required Inlet Chlorine Concentration}) \times (\text{Water Flow} + 2) - 95.5}{6.15} \tag{7.9}$$

The offset of the Water Flow by +2 is merely a protection against division-by-zero.

7.2.3.3 Results

The first part of the following graph has no automatic control, as is evidenced by the occasional stepping of the hypo pump setting by the operator. Then the DMC controller (incorporating the adaptive adjustment of the identified chlorine loss rate constant from the Kalman filter) is switched on at the 80th hour (shown by the "auto line ", which goes from zero to one). The chlorine set point is fixed to 0.95 mg/L.

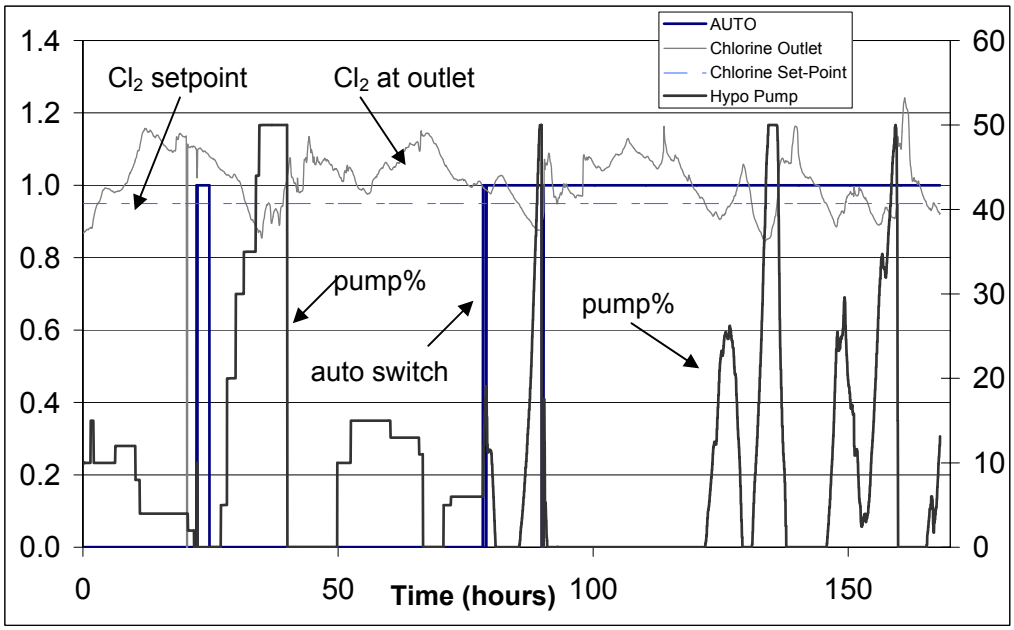


Figure 7-10: Comparison of outlet chlorine control performance by manual operation (up to 80 hours) and by adaptive Dynamic Matrix Controller (after 80 hours)

The automatic control is acceptable, perhaps a little better than the occasional adjustments of the operators. In fact there is some confidence in the new controller amongst the operators. However, when time permits it is intended to improve the tuning of this controller using a longer optimisation horizon (4 hours instead of the present 2 hours), and to increase the move suppression (lower β). This will reduce the gain of the controller, which appears to be overshooting. Bearing in mind that this version of the DMC controller does not treat constraints optimally (it simply "clips" hypo pump settings at maximum and minimum values), one

problem is that the hypo pump cannot be set lower than 0% (whereas hypo is still being added at 0%). On the plot, the main deviations from the set-point are all caused by excessive control action - each peak comes from an excessive hypo pump adjustment - making the controlled variable it a bit oscillatory. Other issues such as appropriate tuning of the Kalman filter, for a desirable rate of variation of the predicted chlorine decay rate constant, should also be examined further.

CHAPTER 8

CONCLUSIONS

This study has focused on the prediction and real-time control of the outlet chlorine concentration for the chlorine contact reservoir at Umgeni Water Wiggins Waterworks, Durban.

In order to control the outlet chlorine concentration, it was necessary to gain an understanding of the physical performance of the reservoir. A Computational Fluid Dynamics study revealed complex behaviour of the reservoir. In particular, it indicated preferential pathways for the flow through the two sections of the reservoir. These create stagnant volumes in which the chlorine concentration is especially depleted. The baffles inside the two sections of the reservoir are probably not well located, and a change of the geometry of the two sections may improve the situation. While Computational Fluid Dynamics can aid understanding of the physical processes of the reservoir, the model is too detailed and computationally intensive to be applicable for real time control. A simplified model is therefore required. The interpretation of a simulated tracer test done using Computational Fluid Dynamics led to the successful creation of compartment models, namely the six-compartment model and the four-compartment model. In order to implement the model on-line, however, the reservoir has also been considered as a single compartment (one-compartment model). This allowed a robust and simple implementation for only a small reduction in prediction accuracy.

As the mathematical models are non-linear with a mix of differential and algebraic equations, they are solved for simulation purposes within an extended Kalman filter. The algorithm is used in two different ways. Initially, the compartment models are able to infer the chlorine decay constant, given the real-time measured observations around the reservoir, including inlet and outlet chlorine concentrations. The effects of the large disturbances in the inlet chlorine measurement, due to filter-washing, are reduced by using a long response time for this Kalman filter. Next, as discussed below, the adapted model, with the continuously updated chlorine decay rate constant, forms the basis of the predictive controller. It was found that the approximation of the chlorine decay kinetics as a first order decay model does not greatly influence the quality of the results, when compared to plant data.

Dynamic Matrix Control (DMC), one of the most popular techniques of Model Predictive Control (MPC), was chosen for controlling the outlet chlorine concentration by manipulating

the inlet chlorine concentration, by means of the sodium hypochlorite dosing pump. DMC is based on a linear convolution model using step-responses, and therefore does not require rigorous derivation from first principles.

The one-compartment model has been simplified for its implementation on-line. In preparation for the commissioning of the on-line control algorithm, and to experiment with the tuning of parameters, preliminary closed loop off-line tests were designed to determine robustness and controller performance. In order to apply the algorithm on-line using the inlet chlorine concentration requested by the DMC, a feedforward controller has been created to manipulate the NaOCl pump flow rate to achieve the inlet chlorine setting.

Acceptable performance of the DMC controller for the outlet chlorine concentration has been achieved, to the extent that there is some confidence in it by experienced operators. It is recommended that the optimisation horizon be increased from 2 to 4 hours, and that the controller gain β be reduced in controlled tests. This should reduce the oscillations observed in the controlled variable, the chlorine composition at the reservoir outlet.

This on-line implementation of advanced control in a water treatment process has shown that there is potential for further development in this area, especially in a case like this where long time constants, temporal changes in behaviour, and a multivariate dependency, conspire to make intuitive operator interventions less effective.

REFERENCES

Åström, K.J. and Wittenmark, B. (1995). Adaptive control. *2nd Ed., Addison-Wesley Publishing Company,* New York, USA.

American Water Works Association. (1999). Water quality and treatment: a handbook for community water supply. *4th Ed., McGraw-Hill,* New York, USA.

Balakrishnan, A.V. (1984). Kalman filtering theory. *Optimization software, inc.,* New York, USA.

Catlin, D.E. (1980). Estimation, control, and the discrete Kalman filter. *Applied Mathematical Sciences, **71***.

Chang, T.S. and Seborg, D.E. (1983). A linear programming approach for multivariable feedback control with inequality constraints. *Int.J.Control, **37**: 583-597.*

Chataigner, O., Gatel, D., Bonne, P. and Cavard, J. (1999). Better disinfection through optimised clearwells. *Proceedings 1999 AWWA annual conference,* Chicago, USA.

Chui, C.K. and Chen, G. (1987). Kalman filtering with real time applications. *Springer-Verlag, second edition.*

Clarke, D.W., Mohtadi, C. and Tuffs, P.S. (1987). Generalized predictive control - Part I. The basic algorithm. *Automatica, **23**: No 2, 137-148.*

Cutler, C.R. and Ramaker, B.L. (1980). Dynamic Matrix Control - a computer control algorithm. *AICHE National Meeting, Houston, Texas.*

Desjardins, R. (1975). Le traitement des eaux. *2nd Ed.,Ecole polytechnique de Montreal,* Montreal, Quebec.

Ducluzaux, B. (1999). Measurements uninterrupted and in situ of the tracers by sensors. Application to the ion iodide. *Actes Neuvième Rencontre d'Octobre, Paris, France.*

Environment Quebec (2002). Accessed on 6 October 2001 from URL, http://www.menv.gouv.qc.ca/index-en.htm

Fang Hua, West, J.R., Barker, R.A. and Forster, C.F. (1998). Modelling of chlorine decay in municipal water supplies. *Water Research, **33**, No 12, 2735-2746.*

Faust, S.D. and Aly, O.M. (1999). Chemistry of water treatment. *2nd Ed.,CRC Press.*

Garcìa, C.E., Prett, D.M. and Morari, M. (1989). Model Predictive Control: theory and practice - a survey. *Automatica, **25**, No. 3, 335-348.*

Guiamba, I.R.F. (2001). Adaptive Dynamic Matrix Control for a Multivariable Training Plant. *MscEng Thesis,* University of Natal, Durban, South Africa.

Isermann, R. (1982). Parameter Adaptive Control Algorithms – A tutorial. *Automatica, **18**, No. 5, 513-528.*

Johnson, P., Graham, N.J.D., and Wilson, M. (1997). Predictive chlorine dosing: a new paradigm. *J. CIWEM, December, **11**.*

Kandasamy, P. and Govender, M. (2002). Chlorine decays kinetics in potable water reservoir. *Laboratory project report.* School of Chemical Engineering, University of Natal, Durban, South Africa.

Keystone Aniline Corporation (2002). Accessed on 6 October 2001 from URL, http://www.keystonecorporation.com

Lacave, B. (2002). Modelling and control of a co-current sugar dryer. *MscEng Thesis,* University of Natal, Durban, South Africa.

Laubush, E.J (1955). Chlorine, its manufacture, properties and use. *Reinhold Publishing Corporation.*

Leclerc, J.P and Grevillot, G. (1998). Le traçage numérique: prédiction de la distribution des temps de séjour (DTS) par les outils de la mécanique des fluides numériques. *Traceurs et méthodes de traçage, **12**.*

Le Grange, L. (1999). Flo++ User Manual. Potchefstroom, South Africa.

Le Lann, M.V., Cabassud, M., and Casamatta, G. (1995). Adaptive Model Predictive Control. *Methods of Model Based Process Control, 427-457. Kluwer Academic Publishers.*Netherlands

Levenspiel, O. (1999)**.** Chemical reaction engineering. *3rd Ed., Wiley.*

Ljung, L. (1999). System identification. Theory for user. *2nd Ed.,Prentice Hall PTR*

Lyonnaise des eaux (2002). Accessed on 5 March 2001 from URL, http://www.lyonnaisedeseaux.fr

Molinari, J. (1976). Intéractions avec le milieu et développements récents dans l'emploi des traceurs artificiels. *La houille blanche, No. 3/4,* Grenoble, France.

Morshedi, A.M., Cutler, C.R. and Shrovaniek, T.A. (1985). Optimal solution of Dynamic Matrix Control with Linear Programming Techniques (LDMC). *Proc. Am. Control Conf., 199-208.* Boston, Massachusetts

Mulholland, M. and Prosser, J.A. (1997). Constrained Linear Dynamic matrix control of a distillation column. *SA Inst. Ch. Eng. 8th Nat. Meeting,* Cape town, South Africa.

National Sanitation Foundation International. *3475 Plymouth rd, Ann Arbor, U.S.A.*

Ogunnaike, B.A. and Ray W.H. (1994). Process Dynamics, Modelling and Control. *Oxford University Press.*

Powell, J.C., Hallam, N.B., West, J.R., Forster, C.F., and Simms, J. (1999). Factors which control bulk chlorine decay rates. *Water research, **34**.*

Qin, S.J. and Badgwell, T.A. (1997). An overview of Industrial Model Predictive Control Technology. *Fifth International Conference on Chemical Process Control, 232-256.*

Sérodes, J.B., Rodriguez, M.J. and Ponton, A. (2000). Chlorocast©: a methodology for developing decision-making tools for chlorine disinfection control. *Environmental Modelling and Software, **16**, issue 1, 2001.*

Umgeni Water (2002) Accessed on 6 June 2001 from URL,

http://www.umgeni.co.za

van der Walt, J.J. (2002). Is a chlorine contact tank really that simple? A CFD investigation into the hydraulics and kinetics of chlorine decay. *WISA conference 2002,* Durban, South Africa.

Viljoen, O.J., Haarhoff, J. and Joubert, J.H.B. (1997). The prediction for chlorine decay from potable water in pipeline systems*. WRC Report No 704/1/97.* Pretoria, South Africa.

World Health Organisation (1984). Guidelines for drinking water quality, No.1: Recommendations. *2nd Ed., Geneva.*

APPENDIX A

EXTENDED KALMAN FILTER FORMULATION

Presented in this appendix, is the extended Kalman filter formulation (Mulholland, 2001) for solution of differential and algebraic equation systems. The first step provides a linearisation of the system using a Taylor series expansion, and then the Kalman filter is used for state estimation.

Consider the system of first order differential and algebraic equations:

$$\begin{aligned} \frac{d\,\boldsymbol{y}}{d\,t} &= \boldsymbol{f}\left(\boldsymbol{y},\boldsymbol{z}\right) \\ \boldsymbol{0} &= \boldsymbol{g}\left(\boldsymbol{y},\boldsymbol{z}\right) \end{aligned} \tag{A.1}$$

where **y** is a vector of state variables and **z** a vector of algebraic variables.

Defining the Jacobians:

$$\boldsymbol{A}=\boldsymbol{J}_{fy}=\begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \cdots & \frac{\partial f_1}{\partial y_N} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} & \cdots & \frac{\partial f_2}{\partial y_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial y_1} & \frac{\partial f_N}{\partial y_2} & \cdots & \frac{\partial f_N}{\partial y_N} \end{bmatrix} \quad \boldsymbol{B}=\boldsymbol{J}_{fz}=\begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \frac{\partial f_1}{\partial z_2} & \cdots & \frac{\partial f_1}{\partial z_M} \\ \frac{\partial f_2}{\partial z_1} & \frac{\partial f_2}{\partial z_2} & \cdots & \frac{\partial f_2}{\partial z_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial z_1} & \frac{\partial f_N}{\partial z_2} & \cdots & \frac{\partial f_N}{\partial z_M} \end{bmatrix}$$

$$\boldsymbol{C}=\boldsymbol{J}_{gy}=\begin{bmatrix} \frac{\partial g_1}{\partial y_1} & \frac{\partial g_1}{\partial y_2} & \cdots & \frac{\partial g_1}{\partial y_N} \\ \frac{\partial g_2}{\partial y_1} & \frac{\partial g_2}{\partial y_2} & \cdots & \frac{\partial g_2}{\partial y_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_N}{\partial y_1} & \frac{\partial g_N}{\partial y_2} & \cdots & \frac{\partial g_N}{\partial y_N} \end{bmatrix} \quad \boldsymbol{D}=\boldsymbol{J}_{gz}=\begin{bmatrix} \frac{\partial g_1}{\partial z_1} & \frac{\partial g_1}{\partial z_2} & \cdots & \frac{\partial g_1}{\partial z_M} \\ \frac{\partial g_2}{\partial z_1} & \frac{\partial g_2}{\partial z_2} & \cdots & \frac{\partial g_2}{\partial z_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_N}{\partial z_1} & \frac{\partial g_N}{\partial z_2} & \cdots & \frac{\partial g_N}{\partial z_M} \end{bmatrix} \tag{A.2}$$

linearise the right hand sides about $(\mathbf{y}_0, \mathbf{z}_0)$ to obtain the augmented system:

$$\begin{aligned} \begin{pmatrix} \dot{\boldsymbol{y}} \\ \hline \boldsymbol{0} \end{pmatrix} &= \begin{pmatrix} \boldsymbol{f}\left(\boldsymbol{y}_0,\boldsymbol{z}_0\right) \\ \hline \boldsymbol{g}\left(\boldsymbol{y}_0,\boldsymbol{z}_0\right) \end{pmatrix} + \left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{B} \\ \hline \boldsymbol{C} & \boldsymbol{D} \end{array}\right] \begin{pmatrix} \left(\boldsymbol{y}-\boldsymbol{y}_0\right) \\ \hline \left(\boldsymbol{z}-\boldsymbol{z}_0\right) \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{F}_0 \\ \hline \boldsymbol{G}_0 \end{pmatrix} + \left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{B} \\ \hline \boldsymbol{C} & \boldsymbol{D} \end{array}\right] \begin{pmatrix} \boldsymbol{y} \\ \hline \boldsymbol{z} \end{pmatrix} \end{aligned} \tag{A.3}$$

where

$$\begin{pmatrix} \boldsymbol{F_0} \\ \hline \boldsymbol{G_0} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}(\boldsymbol{y}_0, \boldsymbol{z}_0) - \boldsymbol{A}\boldsymbol{y_0} - \boldsymbol{B}\boldsymbol{z_0} \\ \hline \boldsymbol{g}(\boldsymbol{y}_0, \boldsymbol{z}_0) - \boldsymbol{C}\boldsymbol{y_0} - \boldsymbol{D}\boldsymbol{z_0} \end{pmatrix} \tag{A.4}$$

To allow for the possibility that some of the **z** elements might be free, overspecify the behaviour by suggesting that **z** will move towards some observed value $\mathbf{z_0}$.

$$\dot{\boldsymbol{z}} = \tfrac{1}{\tau}\left(\boldsymbol{z}_0 - \boldsymbol{z}\right) \tag{A.5}$$

so that Equation A.3 becomes

$$\begin{pmatrix} \dot{\boldsymbol{y}} \\ \hline \dot{\boldsymbol{z}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{F_0} \\ \hline \boldsymbol{H_0} \end{pmatrix} + \left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{B} \\ \hline \boldsymbol{0} & \boldsymbol{E} \end{array}\right] \begin{pmatrix} \boldsymbol{y} \\ \hline \boldsymbol{z} \end{pmatrix} \tag{A.6}$$

with

$$H_0 = \tfrac{1}{\tau} z_0$$
$$E = -\tfrac{1}{\tau} I$$

an additional requirement is also defined from Equation A.3 as

$$\left[\boldsymbol{C} \mid \boldsymbol{D}\right] \begin{pmatrix} \boldsymbol{y} \\ \hline \boldsymbol{z} \end{pmatrix} = -\boldsymbol{G_0} \tag{A.7}$$

To handle that the possibility that the states **y** may also be observed, augment the above equation as follows:

$$\left[\begin{array}{c|c} \boldsymbol{I} & \boldsymbol{0} \\ \hline \boldsymbol{C} & \boldsymbol{D} \end{array}\right] \begin{pmatrix} \boldsymbol{y} \\ \hline \boldsymbol{z} \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}_0 \\ \hline -\boldsymbol{G_0} \end{pmatrix} \tag{A.8}$$

Letting $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{y} \\ \hline \boldsymbol{z} \end{pmatrix}$, $\boldsymbol{\Phi} = \left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{B} \\ \hline \boldsymbol{0} & \boldsymbol{E} \end{array}\right]$ and $\boldsymbol{u}_0 = \begin{pmatrix} \boldsymbol{F_0} \\ \hline \boldsymbol{H_0} \end{pmatrix}$, integrate from 0 to t keeping $\boldsymbol{u_0}$ fixed:

$$\boldsymbol{x_t} = \boldsymbol{e^{\Phi t} x_0} + \left[\boldsymbol{e^{\Phi t} - I}\right]\boldsymbol{\Phi^{-1}}\,\boldsymbol{u_0} \tag{A.9}$$

The same integration can be applied to the interval $\boldsymbol{t \rightarrow t + \Delta t}$, giving

$$\boldsymbol{x_{t+\Delta t}} = \boldsymbol{A_t x_t} + \boldsymbol{B_t u_t} \tag{A.10}$$

with $\boldsymbol{A_t} = \boldsymbol{e^{\Phi \Delta t}}$

$$\boldsymbol{B_t} = \left[\boldsymbol{e^{\Phi \Delta t} - I}\right]\boldsymbol{\Phi^{-1}}$$

and setting $\boldsymbol{C_t} = \left[\begin{array}{c|c} \boldsymbol{L} & \boldsymbol{0} \\ \hline \boldsymbol{C} & \boldsymbol{D} \end{array}\right]$, $\boldsymbol{w_t} = \begin{pmatrix} \boldsymbol{y_0} \\ \hline \boldsymbol{-G_t} \end{pmatrix}$, with $\boldsymbol{L}$ selecting observed states, we also require

$$\boldsymbol{C_t x_t = w_t} \tag{A.11}$$

Representing the equivalent set of measurements by $\hat{\mathbf{w}}_\mathbf{t}$, the Kalman filter is configured as follows:

$$\begin{aligned} \boldsymbol{K_t} &= \boldsymbol{M_t\, C_t^T \left[C_t\, M_t\, C_t^T + R\right]^{-1}} \\ \boldsymbol{x_{t+\Delta t}} &= \boldsymbol{A_t\, x_t + B_t\, u_t + K_t \left[\hat{w}_t - C_t\, x_t\right]} \\ \boldsymbol{M_{t+\Delta t}} &= \boldsymbol{A_t \left[I - K_t\, C_t\right] M_t\, A_t^T + Q} \end{aligned} \tag{A.12}$$

where the covariance matrix is initialised with $\mathbf{M_0}$ small and $\mathbf{Q}$ and $\mathbf{R}$ the expected error covariance matrices for the model and the measurements respectively.

APPENDIX B

TABLE B-1: INTERPRETATION OF VARIABLES IN MATLAB PROGRAM

Time	t
Inlet flow rate data	F0_data
Outlet flow rate data	FR_data
Level data	H_data
Inlet chlorine concentration data	x0_data
Outlet chlorine concentration data	xR_data
Chlorine concentration at the mixing point	xM
Water surface area	AF
Observed variable subscript	o
Inlet subscript	0
Outlet subscript	R
Kinetic factor	k1
Flow from i to j	Fij
Number of compartments	C
Outlet chlorine concentration set-point	xr_sp
Initialisation factor	Init
$\frac{\partial}{\partial \boldsymbol{y}}$ or $\frac{\partial}{\partial \boldsymbol{z}}$ (Jacobians)	Jy or Jz
Initialisation of the covariance matrix M	Mt_start
Covariance matrix M	Mt
Measurement sensitivity factor ε	QR100
Calculation of the Jacobians	Reevaluate
Percent of range of the Jacobians	pcmove
Maximum percent of range of the Jacobians allowing	pcmove_tolerance

Small offset	stepIN
Automatic (DMC is trigerred)	auto
Control moves number	nopt
Optimisation steps to the horizon (P)	nDMC
Δm	B
Δm_{UQO}	dmopt
Δm_{PAST}	dmpast
e_{OL}	eol
B_{OL}	Bol
B_0	B0
W	WW
Λ	Lam

APPENDIX C

MATLAB PROGRAM

```
%*****************************************************************
%                                                                *
%        SIX-COMPARTMENT MODEL (Extended Kalman Filter Solver)   *
%                 with DYNAMIC MATRIX CONTROL                    *
%                                                                *
%                                                                *
%*****************************************************************

close all;
clear all;
%*****************************************************************
%                                                                *
%                    INITIALISATION                              *
%                                                                *
%*****************************************************************

% FOR CONTROL

% Optional parameter list to control solution
pcmove_tol=10;     % max percentage move for any one variable
                   % before reevaluation of Jacobians
minrange=0.001;    % min allowed ranging for Q & R setting
tol = 1e-15;       % Tolerance for matrix exponential convergence
SM = 1e-8;         % Small value to protect against div-by-zero
SMM = 1e-12;       % Small value to weed matrix Mt and matrix K
Mt_START=0.01;     % Initialisation of Mt matrix
nFirstKrecalc = 1; % Number of initial recalculations of K
                   % to get some convergence
Kint = 10;         % No. of steps between re-evaluation of  K
Kcomp = 10;        % No. of compulsory initial re-evaluations of K
QR100 =1;          % Q & R scaling term
FirstK=1;          % Flag
fast_response_factor=2.0;
STARTATLAST = 0;
CONSTRAINTSFROMLAST = 0;
LOADLASTMT = 0;
```

```
% Common Error Factors for Tuning
% multiplies by relative "yo" errors above to get observation errors
% for observed variables only
yo_err_factor =  .5;

% multiplies by relative "y" errors above to set unknown 'z' model    %
errors
yu_err_factor =   5;

% multiplies by relative "f" errors below to get equation errors
f_err_factor  =  .1;

% multiplies elements of R matrix before squaring
Rfactor       =   1;

% ------------------------------------------------------------------------
% FOR DMC
dtDMC=0.03;   % [days]
nDMC=20;
stepIN=0.1;   % shift in input to reveal response
ipointer=1;   % initial pointer position
icount=0;
resp=zeros(nDMC,1);
tlast_step=0;
auto=0;       % initially on manual
nopt=1;
xR_sp         =   0.8;  % initialization set point

% ------------------------------------------------------------------------
% GET PLANT DATA FROM FILE
data_020402;         % That open the file with the data
nstep=size(DATA,1);


% ------------------------------------------------------------------------
% TIME INTERVAL
t  = 0;
t_final=DATA(nstep,1);           % [days]
dtav=t_final/nstep;              % [days]
Tau = fast_response_factor*dtav;


% ------------------------------------------------------------------------
% MAXIMUM POSSIBLE SIZE
nfmax =600;   % number of equations
nymax =800;   % total number of variables
f = zeros(nfmax,1);
ff = zeros(nfmax,5);
y=zeros(nymax,1);
y_lastJ=zeros(nymax,1);
yy=zeros(nymax,6);
yo = zeros(nymax,1);
```

```
ylimflag = zeros(nymax,4); % flags (a: ny count b:1/2=lower/upper
                           % c:limit value d: reference index)
for i=1:nymax
   ylimflag(i,1)=i;        % marker
   ylimflag(i,2)=0;        % no limit
   ylimflag(i,3)=-99;      % no limit value
   ylimflag(i,4)=0;        % no reference index
end

% ------------------------------------------------------------------
% PLOTTING INFORMATION
dtplot = dtav;
p=floor(t_final/dtplot);
tp=zeros(p,1);
tlastplot=0;
iplot=0;



%!!!!!!!!!!!!!!!!!!      Main time loop !!!!!!!!!!!!!!!!!!!!!!!!!!!!


INIT = 1;  % Initialise on First Pass
nKint=0;
nKcomp=0;
upperclip=0;
idata=0;
t=0;

while t<t_final      %t_final

   [t t_final]

   tlast=t;
   idata=idata+1;
   t=DATA(idata,1);
   dt=t-tlast;

   % ----------------------TO BE SET BY USER BELOW-------------------
   % Set present observations on each step

   if INIT
      % number of compartments
      C=6;  % number of compartments
      M=1;  % only 1 species
      % fixed parameter arrays
      AF=zeros(C,1);       % Compartment surface area   [m2]

      % observation arrays
      xo=zeros(C,1); ho=zeros(C,1);

      % working arrays
      x=zeros(C,1); h=zeros(C,1);

      % constraint arrays
      xc=zeros(C,1); hc=zeros(C,1);

      % plotting arrays  (pREDICTIONS)
      xP=zeros(p,C); hP=zeros(p,C);
      k1P=zeros(p,1); alphaP=zeros(p,1); F0P=zeros(p,1);
      FRP=zeros(p,1); x0P=zeros(p,1); xRP=zeros(p,1);xMP=zeros(p,1);
```

```
   F1MP=zeros(p,1); FM2P=zeros(p,1);
   F2SP=zeros(p,1);F34P=zeros(p,1);
   hMP=zeros(p,1);hRP=zeros(p,1);F45P=zeros(p,1);F56P=zeros(p,1);
   F6MP=zeros(p,1);

   % plotting arrays  (OBSERVATIONS)
   xoP=zeros(p,C); hoP=zeros(p,C);
   k1oP=zeros(p,1); alphaoP=zeros(p,1);
   F0oP=zeros(p,1); FRoP=zeros(p,1); x0oP=zeros(p,1);
   xRoP=zeros(p,1);xMoP=zeros(p,1); F1MoP=zeros(p,1); FM2oP=zeros(p,1);
   F2SoP=zeros(p,1);F34oP=zeros(p,1); hMoP=zeros(p,1);hRoP=zeros(p,1);
   F45oP=zeros(p,1);F56oP=zeros(p,1);F6MoP=zeros(p,1);

   % flow resistance factor for reservoir interconnection i/j
   %[(Ml/day)/(m)^0.5]
   beta1M = 100;
   beta2S = 100;
   beta34 = 100;
   beta45 = 100;
   beta56 = 100;
   beta6M = 100;

   % Values for upper constraints
   k1c      =8;                      % [/day]
   F0c      =300;                       % max for F0
   FRc      =300;
   x0c      =3.0;                       % chlorine in total inflow [ppm]
   xRc      =2.0;                       % chlorine in total outflow [ppm]
   hMc      =8;
   hRc      =8;
   xMc      =2;
   for i=1:C
      hc(i)    =8.0;                    % Level  [m]
      xc(i)    =2.0;                    % Chlorine level in outflow [ppm]
   end

end

% FOR MEASURED PLANT DATA ================================

F0_data  =DATA(idata,2);      % Total flow inlet
FR_data  =DATA(idata,3);      % Total flow outlet
H_data   =DATA(idata,4);      % Level
x0_data  =DATA(idata,5);      % Total concentration inlet
xR_data  =DATA(idata,6);      % Total concentration outlet

%=========================================================

% Volume surface area [Ml/m]
AF(1) = 0.2*(116.0*76.5)/1e3;   % First compartment
AF(2) = 1*(116.0*76.5)/1e3;     % Second compartment
AF(3) = 0.2*(116.0*76.5)/1e3;   % part of the plug flow volume
AF(4) = 0.2*(116.0*76.5)/1e3;   % part of the plug flow volume
AF(5) = 0.2*(116.0*76.5)/1e3;   % part of the plug flow volume
AF(6) = 0.2*(116.0*76.5)/1e3;   % part of the plug flow volume


% Initial & Observed Values

k1o      =0.2;% [/day]
alphao   =0.6;
```

```
F0o      =F0_data;
FRo      =FR_data;
if ((~auto)|(tlast==0))
   x0o   =x0_data;      % chlorine in total inflow [ppm]
   x0_set=x0_data;      % for first switch to control
end
if auto
   x0o   =x0_set;       % if it is under control, use the last set value
end
xRo      =xR_data;      % chlorine in total outflow [ppm]
xMo      =0;            % chlorine in the mising point [ppm]
hMo      =H_data;
hRo      =H_data;
F1Mo     =0;
FM2o     =0;
F34o     =0;
F45o     =0;
F56o     =0;
F6Mo     =0;
F2So     =alphao*FR_data;
for i=1:C
xo(i)    =xR_data;    % Chlorine level in flow between reservoir
end

for i=1:C
   ho(i)    =H_data;     % Level  [m]
end

%*****************************************************************
%                                                                *
%                    EXTENDED KALMAN FILTER                      *
%                                                                *
%*****************************************************************

% Check percent of ranges moved since last step
% to tell if must re-evaluate Jacobians


if INIT
   REEVALUATE=1;
else
   REEVALUATE=0;
   for j=1:ny
      pcmove=100*abs((y(j)-y_lastJ(j))/(yy(j,6)-yy(j,5)));
      if pcmove>pcmove_tol
         y_lastJ=y;
         REEVALUATE=1;   % if any one move greater than tolerance
         break;
      end
   end
end



pertfr = 0.001;

n_evals = 1+REEVALUATE*nymax; % breaks out of loop at 1+nys

% EVALUATE FUNCTIONS & their JACOBIAN
for ne=1:n_evals
   if (ne>1)
```

```
   % perturb
   jj=ne-1;

   dyjj = pertfr*(yy(jy(jj),6)-yy(jy(jj),5));
   if dyjj==0 halt; end;
   y(jy(jj))=y(jy(jj))+ dyjj;
end
INI=2;


% ----------------------TO BE SET BY USER BELOW-----------------


for reload=1:(INIT+1)
   if STARTATLAST&INIT
      INI=INI-1;
      if INI==0
         y=yL;
      end
   else
      INI=INIT;
   end;
   n=0;
% Selection of Variables & Observations, Setting of Observation
% Errors and Ranges
% [Observation Errors are as Standard Deviation (-ve=absolute;
% +ve=% of Initial)]
% STATES                              r s o er% min  max         init
for i=1:C n=n+1; if INI yy(n,:)=[ 1 1 1  1  0.0 hc(i)]; y(n)=ho(i);
end; h(i) =y(n); yo(n)=ho(i);  end; % [m]

for i=1:C n=n+1; if INI yy(n,:)=[ 2 1 0  1  0.0 xc(i)]; y(n)=xo(i);
end; x(i) =y(n); yo(n)=xo(i);  end; % [m]

ns=n;  % No of states

% OTHER VARIABLES
n=n+1;           if INI yy(n,:)=[ 3 1 0  .01  0.0  hMc ]; y(n)=hMo;
end; hM    =y(n); yo(n)=hMo;        % [m]

n=n+1;           if INI yy(n,:)=[ 4 1 1  1  0.0  x0c ]; y(n)=x0o;
end; x0    =y(n); yo(n)=x0o;        % [ppm]

n=n+1;           if INI yy(n,:)=[ 5 1 0  2 0.0  xRc ]; y(n)=xRo;
end; xR    =y(n); yo(n)=xRo;        % [ppm]

n=n+1;           if INI yy(n,:)=[ 6 1 0  2  0.0  xMc ]; y(n)=xMo;
end; xM    =y(n); yo(n)=xMo;        % [ppm]

n=n+1;           if INI yy(n,:)=[ 7 1 1  1  0.0  F0c ]; y(n)=F0o;
end; F0    =y(n); yo(n)=F0o;        % [Ml/day]

n=n+1;           if INI yy(n,:)=[ 8 1 1  1 0.0  FRc ]; y(n)=FRo;  end;
FR    =y(n); yo(n)=FRo;        % [Ml/day]

n=n+1;           if INI yy(n,:)=[ 9 1 0  1 -150  300]; y(n)=F1Mo;
end; F1M   =y(n); yo(n)=F1Mo;       % [Ml/day]

n=n+1;           if INI yy(n,:)=[10 1 0  1  -150 300]; y(n)=FM2o;
end; FM2   =y(n); yo(n)=FM2o;       % [Ml/day]
```

```
n=n+1;           if INI yy(n,:)=[11 1 0 0.5 -150  300]; y(n)=F2So;
end; F2S   =y(n); yo(n)=F2So;       % [Ml/day]

n=n+1;           if INI yy(n,:)=[12 1 0  1 -150  300]; y(n)=F34o;
end; F34   =y(n); yo(n)=F34o;       % [Ml/day]

n=n+1;           if INI yy(n,:)=[13 1 0  1 -150  300]; y(n)=F45o;
end; F45   =y(n); yo(n)=F45o;       % [Ml/day]

n=n+1;           if INI yy(n,:)=[14 1 0  1 -150  300]; y(n)=F56o;
end; F56   =y(n); yo(n)=F56o;       % [Ml/day]

n=n+1;           if INI yy(n,:)=[15 1 0  1 -150  300]; y(n)=F6Mo;
end; F6M   =y(n); yo(n)=F6Mo;       % [Ml/day]

n=n+1;           if INI yy(n,:)=[16 1 1  1 0.0  1.0 ]; y(n)=alphao;
end; alpha =y(n); yo(n)=alphao;     % [-]

n=n+1;           if INI yy(n,:)=[17 1 1  1 0   k1c ]; y(n)=k1o;  end;
k1    =y(n); yo(n)=k1o;        % [/day]

n=n+1;           if INI yy(n,:)=[18 1 1  1  0.0   k2c ]; y(n)=k2o;
end; k2    =y(n); yo(n)=k2o;        % [/(ppm.day)]

n=n+1;           if INI yy(n,:)=[19 1 0  .01  0.0 hRc ]; y(n)=hRo;
end; hR    =y(n); yo(n)=hRo;        % [m]

end


% STORE COUNT
ny=n;  % No of variables

n=0;
% Selection of Functions, Setting of Functions
%                                  sel err ref
% STATE EQUATIONS
n=n+1; ff(n,:)=[ 1 1 1 1 0]; f(n)=(F0-F1M)/AF(1);     % dh/dt [m/day]
n=n+1; ff(n,:)=[ 1 1 1 2 0]; f(n)=(F1M+F6M-F2S)/AF(2);
n=n+1; ff(n,:)=[ 1 1 1 3 0]; f(n)=((1-alpha)*F2S-F34)/AF(3);
n=n+1; ff(n,:)=[ 1 1 1 4 0]; f(n)=(F34-F45)/AF(4);
n=n+1; ff(n,:)=[ 1 1 1 5 0]; f(n)=(F45-F56)/AF(5);
n=n+1; ff(n,:)=[ 1 1 1 6 0]; f(n)=(F56-F6M)/AF(6);

n=n+1; ff(n,:)=[ 1 1 2 1 0]; f(n)=(F0*x0-F1M*x(1)-
                                                -
(k1+k2*x(1))*x(1)*AF(1)*h(1)-x(1)*(F0-F1M))/(AF(1)*h(1)+SM);% dx/dt

n=n+1; ff(n,:)=[ 1 1 2 2 0]; f(n)=((F1M+F6M)*xM-F2S*x(2)
 -(k1+k2*x(2))*x(2)*AF(2)*h(2)-x(2)*(F1M+F6M-F2S))/(AF(2)*h(2)+SM);

n=n+1; ff(n,:)=[ 1 1 2 3 0]; f(n)=((1-alpha)*F2S*x(2)-F34*x(3)-
(k1+k2*x(3))*x(3)*AF(3)*h(3)-x(3)*((1-alpha)*F2S-
F34))/(AF(3)*h(3)+SM);

n=n+1; ff(n,:)=[ 1 1 2 4 0]; f(n)=(F34*x(3)-F45*x(4)-
(k1+k2*x(4))*x(4)*AF(4)*h(4)-x(4)*(F34-F45))/(AF(4)*h(4)+SM);

n=n+1; ff(n,:)=[ 1 1 2 5 0]; f(n)=(F45*x(4)-F56*x(5)-
(k1+k2*x(5))*x(5)*AF(5)*h(5)-x(5)*(F45-F56))/(AF(5)*h(5)+SM);
```

```
n=n+1; ff(n,:)=[ 1 1 2 6 0]; f(n)=(F56*x(5)-F6M*x(6)-
(k1+k2*x(6))*x(6)*AF(6)*h(6)-x(6)*(F56-F6M))/(AF(6)*h(6)+SM);


% OTHER VARIABLES
n=n+1; ff(n,:)=[ 1 1  9 1 0]; f(n)=F1M - beta1M*sign(h(1)-
h(2))*sqrt(abs(h(1)-h(2)));%  [Ml/day]


n=n+1; ff(n,:)=[ 1 1 12 1 0]; f(n)=F34 - beta34*sign(h(3)-
h(4))*sqrt(abs(h(3)-h(4)));%  [Ml/day]

n=n+1; ff(n,:)=[ 1 1 13 1 0]; f(n)=F45 - beta45*sign(h(4)-
h(5))*sqrt(abs(h(4)-h(5)));%  [Ml/day]

n=n+1; ff(n,:)=[ 1 1 14 1 0]; f(n)=F56 - beta56*sign(h(5)-
h(6))*sqrt(abs(h(5)-  h(6)));%  [Ml/day]

n=n+1; ff(n,:)=[ 1 1 15 1 0]; f(n)=F6M - beta6M*sign(h(6)-
h(2))*sqrt(abs(h(6)-  h(2)));%  [Ml/day]

n=n+1; ff(n,:)=[ 1 1  8 1 0]; f(n)=FR  - alpha*F2S;

n=n+1; ff(n,:)=[ 1 1  5 1 0]; f(n)=xR  - x(2);

 n=n+1; ff(n,:)=[ 1 1  6 1 0]; f(n)=xM  -
(F1M*x(1)+F6M*x(6))/(F1M+F6M+SM);       %  [ppm]



% ----------------------TO BE SET BY USER ABOVE-----------------------


% STORE COUNT
nf=n;  % No of equations

if INIT

   INIT = 0;

   % Selections to be included in solution
   nfs  = sum(ff(1:nf,1));             % Number of selected functions
   nys  = sum(yy(1:ny,2));             % Number of selected variables
   nss  = sum(yy(1:ns,2));          % Number of selected states only
   if nf>ns
      nos  = sum(ff(ns+1:nf,1));  % Will include all selected
                                  % equations plus states if observed
   else
      nos = 0;
   end
   noss = 0;
   for i=1:ns
      if ((yy(i,2)==1) & (yy(i,3)==1))
         nos  = nos + 1;   % Number of selected & observed variables
         noss = noss + 1;  % Number of observed states
      end
   end

   R=sparse(nos,nos);
   Q=sparse(nys,nys);
   Jy=sparse(nys,nys);
```

```
if noss>0
   Lt=sparse(noss,nss);  % Selection matrix for observed states
end
if LOADLASTMT
   load LASTMT;                    % Mt from last run
else
   Mt=Mt_START*speye(nys,nys);     % Initialise filter covariance
                                   % matrix (sparse)
end
M0=sparse(nys,nys);                % zeros
K=sparse(nys,nos);                 % Set up Kalman gain matrix
us=zeros(nys,1);

% make lookup tables
jy=zeros(nys,1);
i=0;
for j=1:ny
   if yy(j,2)==1
      i=i+1;
      jy(i)=j;
   end
end
jf=zeros(nfs,1);
i=0;
for j=1:nf
   if ff(j,1)==1
      i=i+1;
      jf(i)=j;
   end
end
% lookup table for yy parameters
luyy=zeros(ny,C*M,6);
ilast=0;
for j=1:ny
   i=yy(j,1);
   if ilast~=i
      ilast=i;
      icounter=1;
   else
      icounter=icounter+1;
   end
   luyy(i,icounter,:)=yy(j,:);
end

% R Matrix
i=0;
for j=1:ns
   if ((yy(j,2)==1)&(yy(j,3)==1))
      i=i+1;
      if yy(j,4)<=0
         R(i,i)=(Rfactor*yy(j,4)*yo_err_factor)^2;
      else
         R(i,i)= (Rfactor*yy(j,4)*yo_err_factor*max(yy(j,6)-
         yy(j,5),minrange)/QR100)^2;
      end
   end
end
if nf>ns
   for j=ns+1:nf
      if ff(j,1)==1
         i=i+1;
```

```
                    kk=ff(j,3);  % lookup indices
                    if ff(j,5)==0
                       icounter=ff(j,4);  % i index only
                    else
                       icounter=(ff(j,4)-1)*M+ff(j,5);  % i & j indices
                    end
                    if ff(j,2)>0
                       R(i,i)=
                       (Rfactor*f_err_factor*ff(j,2)*max(luyy(kk,icounter,6)-
                       luyy(kk,icounter,5),minrange)/QR100)^2;
                    else
                       R(i,i)= (Rfactor*f_err_factor*ff(j,2))^2;
                    end
                 end
              end
           end

           % Q Matrix
           i=0;
           for j=1:ns
              if ff(j,1)==1
                 i=i+1;
                 kk=ff(j,3);  % lookup index
                 if ff(j,5)==0
                    icounter=ff(j,4);  % i index only
                 else
                    icounter=(ff(j,4)-1)*M+ff(j,5);  % i & j indices
                 end
                 if ff(j,2)>0
                    Q(i,i)= (f_err_factor*ff(j,2)*max(luyy(kk,icounter,6)-
                    luyy(kk,icounter,5),minrange)/QR100)^2;
                 else
                    Q(i,i)= (f_err_factor*ff(j,2))^2;
                 end
              end
           end
           if ny>ns
              for j=ns+1:ny
                 if yy(j,2)==1
                    if yy(j,3)==1  % observed ?
                       factor=yo_err_factor;
                    else
                       factor=yu_err_factor;
                    end
                    i=i+1;
                    % Q Matrix
                    if yy(j,4)<=0
                       Q(i,i)=(yy(j,4)*factor)^2;
                    else
                       Q(i,i)= (yy(j,4)*factor*max(yy(j,6)-
yy(j,5),minrange)/QR100)^2;
                    end
                 end
              end
           end

           % Check
           if i~=nys
              printf('\n\n #### ERROR #### Must Select Same Equations as
                       States ! \n\n');
              halt;
```

```
      end

    fs  = zeros(nfs,1);    % selected function values
      ys  = zeros(nys,1);     % selected variables
      ysR = zeros(nys,nDMC);  % to keep present states (Cl2 only) for
                              % each solution
      ws  = zeros(nos,1);     % selected & observed variables
      zos = zeros(nys-nss,1); % partly observed, partly previous values

   end

   if ne==1
      f0=f;
   else
      for i=1:nfs
         df=f(jf(i))-f0(jf(i));
         Jy(i,jj)=df/dyjj;
      end
      y(jy(jj))=y(jy(jj))- dyjj;
   end
   if ne==(1+nys)
      break;       % break out of the n_evals loop
   end
end
f = f0;   % back to original position
% load fs selection from f
for i=1:nfs
   fs(i)=f(jf(i));
end

% Fill out yo vector
for i=1:ny
   if (yy(i,3)~=1)
      yo(i)=y(i);
   end
end

% SOLVE

% Set zos,ys and first part of ws
io=0;
for i=1:nss
   ys(i)=y(jy(i));
   if yy(jy(i),3)==1
      io=io+1;
      ws(io)=yo(jy(i));
      Lt(io,i)=1;   % Selection Matrix
   end
end
if nys>nss
   for i=nss+1:nys
      % #### ys(i)=y(jy(i));
      zos(i-nss)=yo(jy(i));
   end
end

if REEVALUATE
   AA = Jy(1:nss,1:nss);
   BB = Jy(1:nss,nss+1:nys);
   CC = Jy(nss+1:nfs,1:nss);
   DD = Jy(nss+1:nfs,nss+1:nys);
```

```
   Et = -speye(nys-nss,nys-nss)/Tau;
   P = sparse(nys,nys);
   P(1:nss,1:nss) = AA;
   P(1:nss,nss+1:nys) = BB;
   P(nss+1:nys,nss+1:nys) = Et;

   % for singular P use series to find "expmPdt_IdivP" = [expm(P*dt)-
   %I]*P^-1
   change=99;
   expmPdt_IdivP=dt*speye(nys,nys);
   changemat=dt*speye(nys,nys);
   Pdt=P*dt;
   n=1;
   while change>tol
      n=n+1;
      changemat=(changemat*Pdt)/n;
      change=sum(sum(abs(changemat)));    % makes a 1-by-n vector with
                                          % the sum of the columns as
                                          % its entries
      expmPdt_IdivP=expmPdt_IdivP+changemat;
      if n> 1000
         n
         break;
      end
   end;

   % Now integrate using matrix exponential
   At=expmPdt_IdivP*P+speye(nys,nys);
   Bt=expmPdt_IdivP;

   % Observation Matrix
   Ct=sparse(nos,nys);
   if noss>0
      Ct(1:noss,1:nss)=Lt;
   end
   Ct(noss+1:nos,1:nss)=CC;
   Ct(noss+1:nos,nss+1:nys)=DD;
end;

% Working Vectors
Ft = fs(1:nss) - AA*ys(1:nss) - BB*ys(nss+1:nys);
Gt = fs(nss+1:nfs) - CC*ys(1:nss) - DD*ys(nss+1:nys);
Ht = zos/Tau;

% Augmented System
us(1:nss) = Ft;
us(nss+1:nys) = Ht;

% Load rest of Observation Vector ws (first part loaded above)
for i=noss+1:nos
   ws(i)=-Gt(i-noss);
end

% KALMAN FILTER
nKint=nKint+1;
nKcomp=nKcomp+1;
if (nKint>Kint)|(nKcomp<=Kcomp)
   nKint=1;
end
if nKint==1     % re-evaluate Kalman 'K' every 'Kint' steps only
```

```
      if FirstK==1
         repeat=nFirstKrecalc;  % to get initial convergence
         FirstK=0;
      else
         repeat=1;
      end;
      for nKrecalc=1:repeat
         % K = Mt*Ct'*inv(Ct*Mt*Ct'+ R);
         CMCR=Ct*Mt*Ct'+ R;
         CM=Ct*Mt;
         K=(CMCR\CM)';
         Mt = At*(speye(nys,nys) - K*Ct)*Mt*At' + Q;
         MP=Mt-SMM*spones(Mt);
         MP=max(M0,MP);      % chop off low positives
         MN=Mt+SMM*spones(Mt);
         MN=min(MN,M0);      % chop off small negatives
         Mt=MP+MN;
         [nzmax(K) nnz(K) nzmax(Mt) nnz(Mt)]
      end
   end

   yslast=ys;

   % Actually operate the filter...
   ys = At*ys + Bt*us + K*(ws-Ct*ys);

   % clipping
   upperclip=0;
   for j=1:nys
      if ys(j)<yy(jy(j),5)
         ys(j)=yy(jy(j),5);          % low clip
         ylimflag(jy(j),2)=1;
         ylimflag(jy(j),3)=yy(jy(j),5); % low limit value
         ylimflag(jy(j),4)=yy(jy(j),1); % reference index
      else
         if ys(j)>yy(jy(j),6)
            ys(j)=yy(jy(j),6);       % high clip
            ylimflag(jy(j),2)=2;
            upperclip=99;
            ylimflag(jy(j),3)=yy(jy(j),6); % low limit value
            ylimflag(jy(j),4)=yy(jy(j),1); % reference index
         else
            ylimflag(jy(j),2)=0;
            ylimflag(jy(j),3)=-99;  % no limit effective
            ylimflag(jy(j),4)=0;    % no reference index
         end
      end
   end

   err2=0;
   err2_deriv=0;
   ntotvar=0;
   ntotderiv=0;
   i=0;
   for j=1:nss
      if yy(jy(j),3)==1
         i=i+1;
         err2 = err2 + ((ys(j)-yo(jy(j)))*yo_err_factor)^2/R(i,i);  %
observed states
         ntotvar=ntotvar+1;
      end
```

```
end
for j=nss+1:nys
   if yy(jy(j),3)==1
      err2 = err2 + ((ys(j)-yo(jy(j)))*yo_err_factor)^2/Q(j,j);
      % errors in observed 'z'
      ntotvar=ntotvar+1;
   end
end
i=noss;
ii=0;
for j=1:nf
   if j<=ns
      if ff(j,1)==1     % selected equation / state
         ii=ii+1;
         err2_deriv = err2_deriv + (f(j)*f_err_factor)^2/Q(ii,ii);
         % derivatives
         ntotderiv=ntotderiv+1;
      end
   else
      if ff(j,1)==1     % selected equation
         i=i+1;
         err2 = err2 + (f(j)*f_err_factor)^2/R(i,i);
         % compensate for the factor in RR
         ntotvar=ntotvar+1;
      end
   end
end

wt_ob_er = sqrt(err2/ntotvar); % to see how well it is doing
wt_deriv_er = dt*sqrt(err2_deriv/ntotderiv); % to see how unsteady the
                                             % process should be compared
                                             %  with actual

% load back to full vector
for i=1:nys
      y(jy(i))=ys(i);
end

%****************************************************************
%                                                               *
%                    STORE FOR PLOTTING                         *
%                                                               *
%****************************************************************

if (t-tlastplot)>=(0.9*dtplot)
   iplot=iplot+1;
   tp(iplot)=t-dt; % NOTE: these are one step out, thus dt subtracted
   tlastplot = t;
   % PREDICTIONS
   for i=1:C
      hP(iplot,i)    =h(i);    % Level in the differents compartments [m]
      xP(iplot,i)    =x(i);    % Chlorine level in compartment [ppm]
   end
   xRP(iplot)        =xR;      % chlorine in total outflow [ppm]
   xMP(iplot)        =xM;      % chlorine in total outflow [ppm]
   x0P(iplot)        =x0;      % chlorine in total inflow [ppm]
   F0P(iplot)        =F0;      % total flow to reservoir [Ml/day]
   FRP(iplot)        =FR;      % total flow from reservoir [Ml/day]
   hMP(iplot)        =hM;      % level at the mixing point
   alphaP(iplot)     =alpha;   % split fraction for total reservoir feed
   F1MP(iplot)       =F1M;
```

```
   FM2P(iplot)        =FM2;
   F2SP(iplot)        =F2S;
   F34P(iplot)        =F34;
   F45P(iplot)        =F45;
   F56P(iplot)        =F56;
   F6MP(iplot)        =F6M;
   k1P(iplot)         =k1;           % 1st order rate constant [/day]

   % OBSERVATIONS
   for i=1:C
      hoP(iplot,i)    =ho(i);
      xoP(iplot,i)    =xo(i);
   end
   xRoP(iplot)        =xRo;
   xMoP(iplot)        =xMo;
   x0oP(iplot)        =x0o;
   F0oP(iplot)        =F0o;
   FRoP(iplot)        =FRo;
   hMoP(iplot)        =hMo;
   alphaoP(iplot)     =alphao;
   F1MoP(iplot)       =F1Mo;
   FM2oP(iplot)       =FM2o;
   F2SoP(iplot)       =F2So;
   F34oP(iplot)       =F34o;
   F45oP(iplot)       =F45o;
   F56oP(iplot)       =F56o;
   F6MoP(iplot)       =F6Mo;
   k1oP(iplot)        =k1o;
   wtd_obs_err(iplot)=wt_ob_er;
   wtd_deriv_err(iplot)=wt_deriv_er;
   xR_spP(iplot)=xR_sp;

   % save chlorine caLculation

   results(iplot,1)=tp(iplot);
   results(iplot,2)=x0P(iplot);
   results(iplot,3)=xR_spP(iplot);
   results(iplot,4)=xRP(iplot);

end

%*****************************************************************
%                                                                *
%                    DYNAMIC MATRIX CONTROL                      *
%                                                                *
%*****************************************************************

% Also step all offset solutions to find step response for DMC control
if tlast==0   % must initialise solutions
   ysR=zeros(nys,nDMC);
   for k=1:nDMC
      ysR(:,k)=yslast;   % initialise all at last ys
   end
end
for k=1:nDMC
   %  need to find 'f' again
      n=0;
   for i=1:C n=n+1;  h(i)  =ysR(n,k); end; % [m]
   for i=1:C n=n+1;  x(i)  =ysR(n,k); end; % [m]
   n=n+1;            hM    =ysR(n,k);      % [m]
   n=n+1;            x0    =ysR(n,k);      % [ppm]
```

```
n=n+1;              xR    =ysR(n,k);       % [ppm]
n=n+1;              xM    =ysR(n,k);       % [ppm]
n=n+1;              F0    =ysR(n,k);       % [Ml/day]
n=n+1;              FR    =ysR(n,k);       % [Ml/day]
n=n+1;              F1M   =ysR(n,k);       % [Ml/day]
n=n+1;              FM2   =ysR(n,k);       % [Ml/day]
n=n+1;              F2S   =ysR(n,k);       % [Ml/day]
n=n+1;              F34   =ysR(n,k);       % [Ml/day]
n=n+1;              F45   =ysR(n,k);       % [Ml/day]
n=n+1;              F56   =ysR(n,k);       % [Ml/day]
n=n+1;              F6M   =ysR(n,k);       % [Ml/day]
n=n+1;              alpha =ysR(n,k);       % [-]
n=n+1;              k1    =ysR(n,k);       % [/day]
n=n+1;              k2    =ysR(n,k);       % [/(ppm.day)]
n=n+1;              hR    =ysR(n,k);       % [m]
n=0;
n=n+1; f(n)=(F0-F1M)/AF(1);;               % dh/dt [m/day]
n=n+1; f(n)=(F1M+F6M-F2S)/AF(2);           % dh/dt [m/day]
n=n+1; f(n)=((1-alpha)*F2S-F34)/AF(3);     % dh/dt [m/day]
n=n+1; f(n)=(F34-F45)/AF(4);               % dh/dt [m/day]
n=n+1; f(n)=(F45-F56)/AF(5);               % dh/dt [m/day]
n=n+1; f(n)=(F56-F6M)/AF(6);               % dh/dt [m/day]



n=n+1; f(n)=(F0*x0-F1M*x(1)...
       -(k1+k2*x(1))*x(1)*AF(1)*h(1)-x(1)*(F0-F1M))/(AF(1)*h(1)+SM);

n=n+1; f(n)=((F1M+F6M)*xM-F2S*x(2)...
       -(k1+k2*x(2))*x(2)*AF(2)*h(2)-x(2)*(F1M+F6M-
      F2S))/(AF(2)*h(2)+SM);

n=n+1; f(n)=((1-alpha)*F2S*x(2)-F34*x(3)...
       -(k1+k2*x(3))*x(3)*AF(3)*h(3)-x(3)*((1-alpha)*F2S-
         F34))/(AF(3)*h(3)+SM);

n=n+1; f(n)=(F34*x(3)-F45*x(4)...
       -(k1+k2*x(4))*x(4)*AF(4)*h(4)-x(4)*(F34-F45))/(AF(4)*h(4)+SM);

n=n+1; f(n)=(F45*x(4)-F56*x(5)...
       -(k1+k2*x(5))*x(5)*AF(5)*h(5)-x(5)*(F45-F56))/(AF(5)*h(5)+SM);

n=n+1; f(n)=(F56*x(5)-F6M*x(6)...
       -(k1+k2*x(6))*x(6)*AF(6)*h(6)-x(6)*(F56-F6M))/(AF(6)*h(6)+SM);

% OTHER VARIABLES
n=n+1; f(n)=F1M - beta1M*sign(h(1)-h(2))*sqrt(abs(h(1)-h(2)));

n=n+1; f(n)=F34 - beta34*sign(h(3)-h(4))*sqrt(abs(h(3)-h(4)));

n=n+1; f(n)=F45 - beta45*sign(h(4)-h(5))*sqrt(abs(h(4)-h(5)));

n=n+1; f(n)=F56 - beta56*sign(h(5)-h(6))*sqrt(abs(h(5)-  h(6)));

n=n+1; f(n)=F6M - beta6M*sign(h(6)- h(2))*sqrt(abs(h(6)-  h(2)));

n=n+1; f(n)=FR  - alpha*F2S;

n=n+1; f(n)=xR  - x(2);

n=n+1; f(n)=xM  - (F1M*x(1)+F6M*x(6))/(F1M+F6M+SM);
```

```
      for i=1:nfs
           fs(i)=f(jf(i));
      end



   Ft = fs(1:nss) - AA*ysR(1:nss,k) - BB*ysR(nss+1:nys,k);
   Gt = fs(nss+1:nfs) - CC*ysR(1:nss,k) - DD*ysR(nss+1:nys,k);
   Ht = zos/Tau;
   % offset the input
   Ht(2) = (zos(2)+stepIN)/Tau;    %BEWARE - THIS INPUT POSITION COULD
                                   %CHANGE !!!!!!
   % Augmented System
   us(1:nss) = Ft;
   us(nss+1:nys) = Ht;
   % Load rest of Observation Vector ws (first part loaded above)
   for i=noss+1:nos
      ws(i)=-Gt(i-noss);
   end
   ysR(:,k) = At*ysR(:,k) + Bt*us + K*(ws-Ct*ysR(:,k)); % step it
end


% Step response at the bigger interval
if (t-tlast_step)>=dtDMC
   icount=icount+1;
   tlast_step=t;
   % Now get the step response by comparison
   for k=1:nDMC
      ipos=ipointer-k+1;  % ipointer will be on the youngest point
      if ipos<1
         ipos=ipos+nDMC;   % wrap
      end
      resp(k)=(ysR(2*C+3,ipos)-ys(2*C+3))/stepIN;  % unit step response :
                                      % Beware: THIS POSITION IS EXIT Cl2
   end
   ipointer=ipointer+1;  % this will be the oldest
   if ipointer>nDMC
      ipointer=1;
   end
   % Push this one down to reference trajectory
   ysR(:,ipointer)=ys;
   % Initialise DMC
   if icount==1   % first call
      DM=zeros(nDMC,nDMC);         % Dynamic Matrix (use a simple square
                                   % system)
      DMol=zeros(nDMC,nDMC);       % Openloop Matrix
      DM0=zeros(nDMC,nDMC);        % Measurement Offset Matrix
      % Initialise counter for Control Time Steps
      ncount=0;
      % Initilaise vector of previous control moves
      dmpast=zeros(nDMC,1);
      % Set up Tuning Matrices WW & Lam
      WW=zeros(nDMC,nDMC);
      for i=1:nDMC
         WW(i,i)=10000;
      end
      Lam=1;
      % Limits for Output
      mmax=4.0;
```

```
   mmin=0.2;
   x0o_last=x0o;
end

% Is Closed-loop Control Required ? ---------------------------------
if t>1
    auto=1;
end
if auto & (icount>=nDMC)    % must also have filled step response
    if t>0
        xR_sp=0.8;
    end
    if t>1   % [days]
        xR_sp=1;
     end

     % now find new value for x0_sp
    % DYNAMIC MATRIX CONTROL ALGORITHM (BELOW)**********************
    % Make Dynamic Matrix DM
    for i=1:nDMC
       for j=1:i
          jj=i-j+1;
          DM(i,j)=resp(jj);
       end
    end
    % Make Openloop Matrix DMol & Offset Measurement Matrix DM0
    for i=1:nDMC
       for j=1:nDMC
          jj=min(nDMC,nDMC+i-j+1);
          DMol(i,j)=DM(nDMC,nDMC+1-jj);    % pick off backwards along
                                           % bottom line of DM
          jjj=nDMC-j+1;
          DM0(i,j)=DM(nDMC,j);
       end
    end
    % Present value of controlled variable
    xR=ys(2*C+3);  % Beware: THIS POSITION IS EXIT C12

    % Openloop error trajectory
    eol=ones(nDMC,1)*(xR-xR_sp)+(DMol-DM0)*dmpast;

    % only one move, so only 1st col of DM
    DMs=DM(:,nopt);

    % only do least squares part, not constrained search
    dmopt=-inv(DMs'*WW*DMs+Lam)*DMs'*WW*eol;
    mpresent=x0o;
    mnew=mpresent+dmopt;
    % Clip externally to limits

    mnew=min(mmax,max(mmin,mnew));
    dmused=mnew-mpresent;
    x0_set=mnew;
 else
    dmused=x0o-x0o_last;
 end
 % Update past moves vector
 for i=1:(nDMC-1)
    dmpast(i)=dmpast(i+1);
 end
 dmpast(nDMC)=dmused;  % newest move at bottom of vector
```

```
        x0o_last=x0o;
    end
    % end of DMC control ------------------------------------------------
end

% Store the last working variables
k1L=k1;                  % [/day]
k2L=k2;                  % [/(day.ppm)]
alphaL=alpha;
F0L=F0;
FRL=FR;
x0L=x0;                   % chlorine in total inflow [ppm]
xRL=xR;                   % chlorine in total outflow [ppm]
xML=xM;                   % chlorine in total outflow [ppm]
hML=hM;
hRL=hR;
F1ML=F1M;
F2SL=F2S;
FM2L=FM2;
F34L=F34;
F45L=F45;
F56L=F56;
F6ML=F6M;
hL=h;                    % Level  [m]
xL=x;                    % Chlorine level in outflow [ppm]
yL=y;
yoL=yo;
save LASTVAR k1L k2L alphaL F0L FRL x0L xRL xML hML hRL F1ML F2SL FM2L F34L
F45L F56L F6ML hL xL yL yoL;
save LASTMT Mt;
save LASTLIM ylimflag;

% ----------------------TO BE SET BY USER BELOW----------------------------

% Draw graphs
%plot(tp(1:p),x0oP(1:p),'rx',tp(1:p),x0P(1:p),'b:',tp(1:p),xRoP(1:p),'b:',tp
(1:p), xRP(1:p),'m',tp(1:p),xR_spP(1:p),'yx');
%legend('x0','x0P','xRo','xRP','xr_setpoint');

p=iplot;

figure(1);
plot(tp(1:p),x0P(1:p),'b:',tp(1:p),xR_spP(1:p),'yx',tp(1:p), xRP(1:p),'m')
axis([0 tp(iplot) 0.6 1.6]);
xlabel('time (day)')
ylabel('chlorine concentration mg/L')
%title('24/09/01 to 30/09/01, with kfit = 0.003*F-0.025*L+0.3*C+50*exp(1/T)-
51.9, temp = 20 C')
figure(2);
plot(tp(1:p),F0P(1:p),'r-',tp(1:p),FRP(1:p),'b-
',tp(1:p),F0oP(1:p),'ro',tp(1:p),FRoP(1:p),'bo');
legend('F0','FR','F0o','FRo');
axis([0 tp(iplot) 0 240]);
xlabel('time (day)')
ylabel('Ml/d')

figure(3);
plot(tp(1:p),hP(1:p,1),'r-
',tp(1:p),hoP(1:p,1),'ro',tp(1:p),xP(1:p,1),'r:',tp(1:p),xoP(1:p,1),'rx',tp(
1:p),hRP(1:p),'m--');
legend('h1','ho1','x1','xo1','hR');
```

```
axis([0 tp(iplot) -10 8]);
xlabel('time (day)')
ylabel('level (m)')

figure(4);
plot(tp(1:p),hP(1:p,2),'g-',tp(1:p),hoP(1:p,2),'g--
',tp(1:p),xP(1:p,2),'g:',tp(1:p),xoP(1:p,2),'gx',tp(1:p),hRP(1:p),'m--');
legend('h2','ho2','x2','xo2','hR');
axis([0 tp(iplot) 0 8]);
xlabel('time (day)')

figure(5);
plot(tp(1:p),hP(1:p,3),'b-
',tp(1:p),hoP(1:p,3),'bo',tp(1:p),xP(1:p,3),'b:',tp(1:p),xoP(1:p,3),'bx',tp(
1:p),hRP(1:p),'m--');
legend('h3','ho3','x3','xo3','hR');
axis([0 tp(iplot) 0 8]);
xlabel('time (day)')

figure(6);
plot(tp(1:p),hP(1:p,4),'c-
',tp(1:p),hoP(1:p,4),'co',tp(1:p),xP(1:p,4),'c:',tp(1:p),xoP(1:p,4),'cx',tp(
1:p),hRP(1:p),'m--');
legend('h4','ho4','x4','xo4','hR');
axis([0 tp(iplot) 0 8]);
xlabel('time (day)')

figure(7);
plot(tp(1:p),k1P(1:p),'bl-',tp(1:p),k1oP(1:p),'blo');
legend('k1','k1o');
axis([0 tp(iplot) 0 5]);
xlabel('time (day)')
ylabel('k (/day)')

figure(8);
plot(tp(1:p),k2P(1:p),'m-',tp(1:p),k2oP(1:p),'mo');
legend('k2','k2o');
axis([0 tp(iplot) 0 10]);
xlabel('time (day)')

figure(9);
plot(tp(1:p),alphaP(1:p),'r-',tp(1:p),alphaoP(1:p),'ro');
legend('alpha','alphao');
axis([0 tp(iplot) 0 1]);
xlabel('time (day)')

figure(11);
plot(tp(1:p),wtd_deriv_err(1:p),'b');
legend('wtd-deriv-err');

figure(12);
plot(tp(1:p),wtd_obs_err(1:p),'r');
legend('wtd-obs-err');


% ----------------------TO BE SET BY USER ABOVE----------------------------
```

APPENDIX D

TABLE D-1: INTERPRETATION OF VARIABLES IN ADROIT SCRIPT PROGAM (VISUAL BASIC)

Interval time between two runs (t_P)	dt_mod
Internal time to the EKF	dt_kal
Internal time to the DMC	dt_dmc
Inlet flow rate	F0_data
Level	H_data
Measured inlet chlorine concentration (C_0)	x0_data
Measured outlet chlorine concentration (C)	xr_data
Outlet chlorine concentration set-point	xr_sp
Kalman gain ε	alpha
DMC gain β	beta
Calculated inlet chlorine concentration (C_{DMC})	x0_mod
Calculated outlet chlorine concentration	xr_mod
Last calculated inlet chlorine concentration	x0_mod_last
Last calculated outlet chlorine concentration	xr_mod_last
Kalman filter matrix K	k_kal
Calculated kinetic factor	kk_kal

Smooth inlet chlorine concentration (C_{filter})	x0_data_slow
Smooth inflow rate (F_{filter})	F0_data_slow
Smooth pump flow rate (P_{filter})	pump_data_slow
Inlet chlorine concentration estimated for a certain value of the pump flow rate and the inflow rate (C_{pump})	x0_pred
Observed pump flow rate (P)	pump_data
Pump setting ($P_{setting}$)	pump_setting
Automatic (DMC is triggered)	auto
Control moves number	nopt
Optimisation steps to the horizon (P)	nDMC
Δm	B
Δm_{UQO}	dmopt
Δm_{PAST}	dmpast
e_{OL}	eol
B_{OL}	Bol
B_0	B0
Λ	lam
W	W

APPENDIX E

ADROIT SCADA System : Visual Basic Program

```
Sub fnAdd()


' Tags to be set up so that values can be changed on-line

'Pump feedforward model parameters
GRADPUMP = 6.15
INTPUMP = 94.5
FLOWPROTECT = 1

auto = Adroit.GetTag("AUTO.value")                  'auto/manual
F0_data = Adroit.GetTag("FIT20144.value")           'flow inlet
Fr_data = Adroit.GetTag("712IN001QI670AINT.value")  'flow outlet
H_data = Adroit.GetTag("712PL001LI701AI.value")     'level
x0_data = Adroit.GetTag("712CR001QI601AI.value")    'chlorine concentration inlet
xr_data = Adroit.GetTag("712PL001QI6211AI.value")   'chlorine concentration outlet ###### WORKS
pump_data = Adroit.GetTag("FI29007_AO.value")
x0_pred = (GRADPUMP * pump_data + INTPUMP) / (F0_data + FLOWPROTECT)
Adroit.SetTag "X0PRED.value", x0_pred

alpha = Adroit.GetTag("ALPHA.value")                ' Kalman filter tuning: filter gain
beta = Adroit.GetTag("BETA.value")                  ' DMC tuning: DMC gain
delta = Adroit.GetTag("DELTA.value") / 1000         ' should be 1.5  (means 0.0015 actually): factor of 1000 so can see 0.001 on
screen !!!!

xr_sp = Adroit.GetTag("XR-SP.value")                ' chlorine setpoint

sec = Adroit.GetTag("systemInfo.second")
minu = Adroit.GetTag("systemInfo.minute")
hr = Adroit.GetTag("systemInfo.hour")

time_now = hr * 3600 + minu * 60 + sec

'flag for first step  Set at startup

first_mod_step = Adroit.GetTag("CALL.value")

'Normal scan  START

'initialisation

' (1) Kalman filter parameter
dt_kal_factor = 20
```

```
' (2) DMC parameters
dt_dmc_factor = 36     'time step multiple for DMC
ndmc = 40              'time horizon
nopt_dmc = 2           'number of optimised control moves
pump_set_min = 0
pump_set_max = 50

Dim dmpast(40)       'past moves matrix
Dim b(40)            'step response vector
Dim BB(40, 40)       'Future dynamic matrix
Dim Bol(40, 40)      'Past dynamic matrix
Dim B0(40, 40)       'Present dynamic matrix
Dim eol(40, 40)
Dim z(40, 40)
Dim Admc(40, 40)
Dim W(40, 40)
Dim lam(40, 40)
Dim L(40, 40)
Dim WA(40, 40)
Dim AWA(40, 40)
Dim Weol(40, 40)
Dim AWeol(40, 40)
Dim h(40, 40)
Dim c(40, 40)
Dim g(40, 40)
Dim L_inv(40, 40)

If dt_kal_factor < 1 Or dt_dmc_factor < dt_kal_factor Then
  Adroit.SetTag "TIME-ALARM.value", "TIME-ALARM"    'system error report
End If
Adroit.SetTag "TIME-ALARM.value", ""

' Re initialize if requested

If first_mod_step = 1 Then

      'initialise
      first_mod_step = 0
      Adroit.SetTag "CALL.value", first_mod_step        ' toggle

      ' initialize time of previous call
      time_lastcall = time_now

      ' Kalman filter time gap : initialize earlier to force an execute
      t_kal = 0

      ' DMC time gap : initialize earlier to force an execute
      t_dmc = 0
```

```
'rate constant guess MM020919
kk_kal = 1.1
M_kal = 0.01

'initial state
x0_pred_kflast = x0_pred
x0_pred_dmclast = x0_pred
xr_data_kflast = xr_data

' zero entire "dmpast " vector
Adroit.SetTag ("D1.value"), 0
Adroit.SetTag ("D2.value"), 0
Adroit.SetTag ("D3.value"), 0
Adroit.SetTag ("D4.value"), 0
Adroit.SetTag ("D5.value"), 0
Adroit.SetTag ("D6.value"), 0
Adroit.SetTag ("D7.value"), 0
Adroit.SetTag ("D8.value"), 0
Adroit.SetTag ("D9.value"), 0
Adroit.SetTag ("D10.value"), 0
Adroit.SetTag ("D11.value"), 0
Adroit.SetTag ("D12.value"), 0
Adroit.SetTag ("D13.value"), 0
Adroit.SetTag ("D14.value"), 0
Adroit.SetTag ("D15.value"), 0
Adroit.SetTag ("D16.value"), 0
Adroit.SetTag ("D17.value"), 0
Adroit.SetTag ("D18.value"), 0
Adroit.SetTag ("D19.value"), 0
Adroit.SetTag ("D20.value"), 0
Adroit.SetTag ("D21.value"), 0
Adroit.SetTag ("D22.value"), 0
Adroit.SetTag ("D23.value"), 0
Adroit.SetTag ("D24.value"), 0
Adroit.SetTag ("D25.value"), 0
Adroit.SetTag ("D26.value"), 0
Adroit.SetTag ("D27.value"), 0
Adroit.SetTag ("D28.value"), 0
Adroit.SetTag ("D29.value"), 0
Adroit.SetTag ("D30.value"), 0
Adroit.SetTag ("D31.value"), 0
Adroit.SetTag ("D32.value"), 0
Adroit.SetTag ("D33.value"), 0
Adroit.SetTag ("D34.value"), 0
Adroit.SetTag ("D35.value"), 0
Adroit.SetTag ("D36.value"), 0
Adroit.SetTag ("D37.value"), 0
Adroit.SetTag ("D38.value"), 0
Adroit.SetTag ("D39.value"), 0
Adroit.SetTag ("D40.value"), 0
```

```
Else
    kk_kal = Adroit.GetTag("K.value")
    M_kal = Adroit.GetTag("M.value")
    x0_pred_kflast = Adroit.GetTag("X0PRED-KFLAST.value")
    x0_pred_dmclast = Adroit.GetTag("X0PRED-DMCLAST.value")
    xr_data_kflast = Adroit.GetTag("XRDATA-KFLAST.value")
    time_lastcall = Adroit.GetTag("TIME-LASTCALL.value")
    t_kal = Adroit.GetTag("T-KAL.value")
    t_dmc = Adroit.GetTag("T-DMC.value")

End If


' Main loop

dt_mod_normal = 5

dt_mod = time_now - time_lastcall
time_lastcall = time_now

' Protection against daily clock wrap-around and other time-upsets
If dt_mod < 0 Then
  dt_mod = dt_mod + 24 * 3600
End If
If dt_mod <= 0 Or dt_mod > 4 * dt_mod_normal Then
  dt_mod = dt_mod_normal
End If

Adroit.SetTag "TIME-LASTCALL.value", time_lastcall
Adroit.SetTag "DT-MOD.value", dt_mod

' get time steps from this
dt_kal = dt_kal_factor * dt_mod
dt_dmc = dt_dmc_factor * dt_mod
dt_modd = dt_mod / (24 * 3600)
dt_kald = dt_kal / (24 * 3600)
dt_dmcd = dt_dmc / (24 * 3600)


'Present parameter
AH = (2 * (116 * 76.5) / 1000)  'Active volume surface area [ML/d]  "####
Tau = (H_data * AH) / F0_data
Ac = -1 / Tau
Bc = 1 / Tau

'(1)  Kalman filter

t_kal = t_kal + dt_mod
```

```
Do While t_kal > 0

        t_kal = t_kal - dt_kal

        R_kal = 1               'kalman R weight
        'Clip limit
        k_kal_min = 0
        k_kal_max = 5

        Q_kal = alpha * R_kal  'kalman Q weight
        A_kal = Exp(Ac * dt_kald)
        B_kal = (A_kal - 1) * (1 / Ac) * Bc

        'Actual model here is
        ' xri+1 = A_kal*xri + B_kal*x0i-k_kal*xri*dt_kal
        ' So kalman filter for k_kal is
        'k_kali+1 = I*k_kali + 0 +k_kal*([-xri+1+A_kal*xri+B_kal*x0i]-xri*dt_kal]*k_kali

        Ak = 1
        Bk = 0
        Gk = xr_data_kflast * dt_kald
        yk = -xr_data + A_kal * xr_data_kflast + B_kal * x0_pred_kflast
        K_kal = M_kal * Gk * 1 / (Gk * M_kal * Gk + R_kal)
        kk_kal = kk_kal + K_kal * (yk - Gk * kk_kal)


        'clip for k

        If kk_kal < k_kal_min Then
           kk_kal = k_kal_min
        End If
        If kk_kal > k_kal_max Then
           kk_kal = k_kal_max
        End If


        M_kal = (1 - K_kal * Gk) * M_kal + Q_kal
        xr_data_kflast = xr_data
        x0_pred_kflast = x0_pred

        Adroit.SetTag "K.value", kk_kal
        Adroit.SetTag "M.value", M_kal
        Adroit.SetTag "X0PRED-KFLAST.value", x0_pred_kflast
        Adroit.SetTag "XRDATA-KFLAST.value", xr_data_kflast
        Adroit.SetTag "YK.value", yk
        Adroit.SetTag "A-KAL.value", A_kal
        Adroit.SetTag "B-KAL.value", B_kal
```

```
Loop

'storage de t_kal

Adroit.SetTag "T-KAL.value", t_kal

'(2) Dmc control algorithm

t_dmc = t_dmc + dt_mod

Do While t_dmc > 0

        t_dmc = t_dmc - dt_dmc

        'load back "dmpast" vector

        dmpast(1) = Adroit.GetTag("D1.value")
        dmpast(2) = Adroit.GetTag("D2.value")
        dmpast(3) = Adroit.GetTag("D3.value")
        dmpast(4) = Adroit.GetTag("D4.value")
        dmpast(5) = Adroit.GetTag("D5.value")
        dmpast(6) = Adroit.GetTag("D6.value")
        dmpast(7) = Adroit.GetTag("D7.value")
        dmpast(8) = Adroit.GetTag("D8.value")
        dmpast(9) = Adroit.GetTag("D9.value")
        dmpast(10) = Adroit.GetTag("D10.value")
        dmpast(11) = Adroit.GetTag("D11.value")
        dmpast(12) = Adroit.GetTag("D12.value")
        dmpast(13) = Adroit.GetTag("D13.value")
        dmpast(14) = Adroit.GetTag("D14.value")
        dmpast(15) = Adroit.GetTag("D15.value")
        dmpast(16) = Adroit.GetTag("D16.value")
        dmpast(17) = Adroit.GetTag("D17.value")
        dmpast(18) = Adroit.GetTag("D18.value")
        dmpast(19) = Adroit.GetTag("D19.value")
        dmpast(20) = Adroit.GetTag("D20.value")
        dmpast(21) = Adroit.GetTag("D21.value")
        dmpast(22) = Adroit.GetTag("D22.value")
        dmpast(23) = Adroit.GetTag("D23.value")
        dmpast(24) = Adroit.GetTag("D24.value")
        dmpast(25) = Adroit.GetTag("D25.value")
        dmpast(26) = Adroit.GetTag("D26.value")
        dmpast(27) = Adroit.GetTag("D27.value")
        dmpast(28) = Adroit.GetTag("D28.value")
        dmpast(29) = Adroit.GetTag("D29.value")
        dmpast(30) = Adroit.GetTag("D30.value")
        dmpast(31) = Adroit.GetTag("D31.value")
        dmpast(32) = Adroit.GetTag("D32.value")
        dmpast(33) = Adroit.GetTag("D33.value")
        dmpast(34) = Adroit.GetTag("D34.value")
```

```
dmpast(35) = Adroit.GetTag("D35.value")
dmpast(36) = Adroit.GetTag("D36.value")
dmpast(37) = Adroit.GetTag("D37.value")
dmpast(38) = Adroit.GetTag("D38.value")
dmpast(39) = Adroit.GetTag("D39.value")
dmpast(40) = Adroit.GetTag("D40.value")


'move older moves up to the stack
j = 1
Do While j < 40
   j = j + 1
   dmpast(j - 1) = dmpast(j)
Loop

dmpast(40) = x0_pred - x0_pred_dmclast
x0_pred_dmclast = x0_pred

Adroit.SetTag ("X0PRED-DMCLAST.value"), x0_pred_dmclast

Adroit.SetTag ("D1.value"), dmpast(1)
Adroit.SetTag ("D2.value"), dmpast(2)
Adroit.SetTag ("D3.value"), dmpast(3)
Adroit.SetTag ("D4.value"), dmpast(4)
Adroit.SetTag ("D5.value"), dmpast(5)
Adroit.SetTag ("D6.value"), dmpast(6)
Adroit.SetTag ("D7.value"), dmpast(7)
Adroit.SetTag ("D8.value"), dmpast(8)
Adroit.SetTag ("D9.value"), dmpast(9)
Adroit.SetTag ("D10.value"), dmpast(10)
Adroit.SetTag ("D11.value"), dmpast(11)
Adroit.SetTag ("D12.value"), dmpast(12)
Adroit.SetTag ("D13.value"), dmpast(13)
Adroit.SetTag ("D14.value"), dmpast(14)
Adroit.SetTag ("D15.value"), dmpast(15)
Adroit.SetTag ("D16.value"), dmpast(16)
Adroit.SetTag ("D17.value"), dmpast(17)
Adroit.SetTag ("D18.value"), dmpast(18)
Adroit.SetTag ("D19.value"), dmpast(19)
Adroit.SetTag ("D20.value"), dmpast(20)
Adroit.SetTag ("D21.value"), dmpast(21)
Adroit.SetTag ("D22.value"), dmpast(22)
Adroit.SetTag ("D23.value"), dmpast(23)
Adroit.SetTag ("D24.value"), dmpast(24)
Adroit.SetTag ("D25.value"), dmpast(25)
Adroit.SetTag ("D26.value"), dmpast(26)
Adroit.SetTag ("D27.value"), dmpast(27)
Adroit.SetTag ("D28.value"), dmpast(28)
Adroit.SetTag ("D29.value"), dmpast(29)
Adroit.SetTag ("D30.value"), dmpast(30)
```

```
Adroit.SetTag ("D31.value"), dmpast(31)
Adroit.SetTag ("D32.value"), dmpast(32)
Adroit.SetTag ("D33.value"), dmpast(33)
Adroit.SetTag ("D34.value"), dmpast(34)
Adroit.SetTag ("D35.value"), dmpast(35)
Adroit.SetTag ("D36.value"), dmpast(36)
Adroit.SetTag ("D37.value"), dmpast(37)
Adroit.SetTag ("D38.value"), dmpast(38)
Adroit.SetTag ("D39.value"), dmpast(39)
Adroit.SetTag ("D40.value"), dmpast(40)



'check if the controller is on auto or not

If auto = 1 Then
    ' AUTO : compute new output
    ' run Dmc algorithm


    i = 0
    Do While i < ndmc
      i = i + 1
      lam(i, i) = 1
      W(i, i) = beta * lam(i, i)
    Loop

    ' make a local unti step response
    A_dmc = Exp(Ac * dt_dmcd)
    B_dmc = (A_dmc - 1) * (1 / Ac) * Bc
    AA_dmc = A_dmc - kk_kal * dt_dmcd

    i = 0
    Do While i < ndmc
      i = i + 1

      If i = 1 Then
         b_last = 0
      Else
         b_last = b(i - 1)
      End If

      b(i) = AA_dmc * b_last + B_dmc * 1
    Loop

    'load the dynamic matrix BB
    i = 0

    Do While i < ndmc
      i = i + 1
```

```
  j = 0

  Do While j < i
    j = j + 1
    jj = i - j + 1
    BB(i, j) = b(jj)
  Loop

Loop

'load the matrices Bol and B0
i = 0
Do While i < ndmc
  i = i + 1

  j = 0
  Do While j < ndmc
  j = j + 1
     If ndmc < (ndmc + i - j + 1) Then
          jj = ndmc
     Else
          jj = ndmc + i - j + 1
     End If
     Bol(i, j) = BB(ndmc, ndmc + 1 - jj)
     B0(i, j) = BB(ndmc, j)
   Loop
Loop

'open loop error trajectory
i = 0

Do While i < ndmc
i = i + 1

' (Bol-B0)*dmpast line by line
   hj = 0
   j = 0
   Do While j < ndmc
     j = j + 1
     hj = (Bol(i, j) - B0(i, j)) * dmpast(j) + hj
   Loop
   eol(i, 1) = hj + (xr_data - xr_sp)  ' control model value - reset to plant value below for online
Loop



' make A
i = 0
Do While i < ndmc
  i = i + 1
```

```
  j = 0

  Do While j < nopt_dmc
     j = j + 1
     Admc(i, j) = BB(i, j)
  Loop
Loop

'only do least squares part, not constrained search
' calcul of inv(A'*W*A+lam) *A'*W*eol

'calcul de W*A
mult W, Admc, ndmc, nopt_dmc, ndmc, WA

'calcul de A'*W*A
trans_mult Admc, WA, nopt_dmc, nopt_dmc, ndmc, AWA

'calcul de L=Admc'*w*Admc+lam

i = 0
Do While i < nopt_dmc
  i = i + 1
  j = 0
  Do While j < nopt_dmc
     j = j + 1
     L(i, j) = AWA(i, j) + lam(i, j)
  Loop
Loop


'calcul of inverse of L

Invert L, nopt_dmc, L_inv

'calcul z=L*A'*W*eol
'calcul W*eol
mult W, eol, ndmc, 1, ndmc, Weol

'Calcul A'*Weol

trans_mult Admc, Weol, nopt_dmc, 1, ndmc, AWeol

'Calcul z
mult L_inv, AWeol, nopt_dmc, 1, nopt_dmc, z

dx0_pred = -z(1, 1)

' #### MM021210 : Convert directly to a pump setting using observed ratio
pump_setting = pump_data + dx0_pred * (F0_data + FLOWPROTECT) / GRADPUMP
If pump_setting < pump_set_min Then
```

```
            pump_setting = pump_set_min
          End If
          If pump_setting > pump_set_max Then
            pump_setting = pump_set_max
          End If

          Adroit.SetTag ("FI29007_AO.value"), pump_setting

        End If

Loop

Adroit.SetTag "T-DMC.value", t_dmc


End Sub

Sub mult(f, h, bnr, cnc, cnr, g)
'calcul of cofactor for a matrix

i = 0
Do While i < bnr
  i = i + 1
  j = 0
  Do While j < cnc
    j = j + 1
    Sum = 0
    k = 0
    Do While k < cnr
      k = k + 1
      Sum = Sum + f(i, k) * h(k, j)
    Loop
    g(i, j) = Sum
  Loop
Loop



End Sub

Sub trans_mult(h, c, bnc, cnc, cnr, g)
'calcul of a=b_transpose*c

i = 0
Do While i < bnc
  i = i + 1
  j = 0
  Do While j < cnc
    j = j + 1
    Sum = 0
```

```
      k = 0
      Do While k < cnr
         k = k + 1
         Sum = Sum + h(k, i) * c(k, j)
      Loop
      g(i, j) = Sum
   Loop
Loop

End Sub

Sub Invert(a, N, y)


Dim d
Dim indx(2)
Dim col(2)

d = 1


Lu_decomp a, N, indx, d  'decompose the matrix just once

      j = 0
      Do While j < N
         j = j + 1
         i = 0
         Do While i < N
            i = i + 1
            col(i) = 0
         Loop
         col(j) = 1
         Lu_sol a, N, indx, col
         i = 0
         Do While i < N
            i = i + 1
            y(i, j) = col(i)
         Loop
      Loop

End Sub

Sub Lu_decomp(a, N, indx, d)

'Initialisation

   'no row interchange yet
   d = 1
   tiny = 0.00000001
   sumi = 0
```

```
   imax = 1

   'vv stores the implicit scaling of each row
   Dim vv(2)
' Loop over the row to get the implicit scaling information
i = 0
 Do While i < N
   i = i + 1
   j = 0

   Do While j < N
      j = j + 1
      If Abs(a(i, j)) > big Then
         big = Abs(a(i, j))
      End If
   'No nonzero largest element
      If big = 0 Then
      End If
   'Save the scaling
   vv(i) = 1 / big
   Loop
Loop

'Loop over columns of Crout's method

j = 0
Do While j < N
   j = j + 1

   If j > 1 Then

      i = 0
      Do While i < (j - 1)
         i = i + 1
         sumi = a(i, j)
            If i > 1 Then
               k = 0
               Do While k < (i - 1)
                  k = k + 1
                  sumi = sumi - a(i, k) * a(k, j)
                  a(i, j) = sumi
               Loop
            End If
      Loop
   End If
   'Initialize for the search of the largest pivot element
   big = 0
   i = j - 1
   Do While i < N
      i = i + 1
```

```
   sumi = a(i, j)
   If j > 1 Then
      k = 0
      Do While k < (j - 1)
         k = k + 1
         sumi = sumi - a(i, k) * a(k, j)
         a(i, j) = sumi
         dum = vv(i) * Abs(sumi)

         'is the figure of merit for the pivot better than the best so far ?
         If dum >= big Then
               big = dum
               imax = i
         End If
      Loop
   End If

Loop

'Do we need to interchange rows ?
If j <> imax Then
   k = 0
   Do While k < N
      k = k + 1
      dum = a(imax, k)
      a(imax, k) = a(j, k)
      a(j, k) = dum
   Loop
   ' ...and change the  parity of d
   d = -d
   vv(imax) = vv(j)
End If

indx(j) = imax

If a(j, j) = 0 Then
   a(j, j) = tiny ' if the pivot element is zero the matrix is singular
End If

If j <> N Then
   dum = 1 / a(j, j)

   i = j
   Do While i < N
      i = i + 1
      a(i, j) = a(i, j) * dum
   Loop
End If   'go back for the next column in the reduction

i = 0
```

```
   Do While i < N
      i = i + 1
      vv(i) = 0
   Loop
Loop


End Sub

Sub Lu_sol(a, N, indx, col)

   ii = 0
   i = 0
   Do While i < N
      i = i + 1
      ip = indx(i)
      sumi = col(ip)
      col(ip) = col(i)

      If ii <> 0 Then
         j = ii - 1
         Do While j < (i - 1)
            j = j + 1
            sumi = sumi - a(i, j) * col(j)
         Loop

      Else
         If sumi <> 0 Then
            ii = i
         End If
      End If
         col(i) = sumi

   Loop

   i = N + 1
   Do While i > 1
      i = i - 1

      sumi = col(i)
         j = i
         Do While j < N
            j = j + 1
            sumi = sumi - a(i, j) * col(j)
         Loop
            col(i) = sumi * 1 / a(i, i)
   Loop

End Sub
```

APPENDIX F

ADAPTIVE PREDICTIVE CONTROL OF THE CHLORINE CONCENTRATION AT THE OUTLET OF THE CHLORINE CONTACT RESERVOIR

USER MANUAL

5.1 AIM OF THE ALGORITHM

This algorithm is to control the outlet chlorine concentration of the reservoir by manipulating the inlet chlorine concentration of the reservoir. The algorithm has two main parts. First, an extended Kalman filter estimates the kinetic factor of the chlorine concentration decay, then a Dynamic Matrix Controller manipulates the inlet chlorine concentration to achieve the outlet chlorine set-point.

5.2 VALUES SET BY THE OPERATORS

If the algorithm is used online, the ***online*** factor has to be set to 1.To trigger the Dynamic Matrix Controller the ***auto*** factor has to be set to 1 as well.

To force the filter to follow the observation, the Kalman gain factor (***alpha***) can be increased until 100, above this value, the filter can become unstable. The recommended setting is 10 so that it responds only to variation longer than 4 hours.

A Dynamic Matrix Controller gain (***beta***) has to be set as well. The higher it is, the better will be the performance of the controller, but if it is too high, the Dynamic Matrix controller becomes unstable. The recommended setting is 10.

To smooth the inlet chlorine concentration, the inlet flow rate and the pump flow rate, a filter has been created. Its smoothing factor is ***delta***. The recommended setting is 0.0005.

Each time the program needs to be re-initialised, or each time the program is activated, the ***initialisation*** factor is set to 1. It will come back automatically to the value 0 after a first iteration.

The operator set a value (between 0.8 and 1.2 mg/L), which has to be reach by the outlet chlorine concentration of the reservoir (***Set-point***).

5.3 TIME LOOP

The time of the day is set on the User Interface screen (***Time***). As the program needs to know the time of its last call (***time last call***), the internal time of the extended Kalman filter (***t-kal***), the internal time of the Dynamic Matrix Controller (***t-dmc***). Each time ***t-kal*** or ***t-dmc*** are positive, the extended Kalman filter or the Dynamic Matrix Controller are running.

The algorithm must be called each 5 seconds, which means that the time between two calls is set in seconds in the Script Agent Configurator as 5 seconds. This elapsed time appears on the User Interface screen, named ***t_now-t_last***. Thus, it is easy to verify that the program is called effectively each 5 seconds.

5.4 PROGRAM: VALUES STORAGE

5.4.1 *Extended Kalman filter parameters*

This updates at each iteration the kinetic factor (***kinetic factor***) and the covariance matrix (***M value***) and it needs them for the next call.

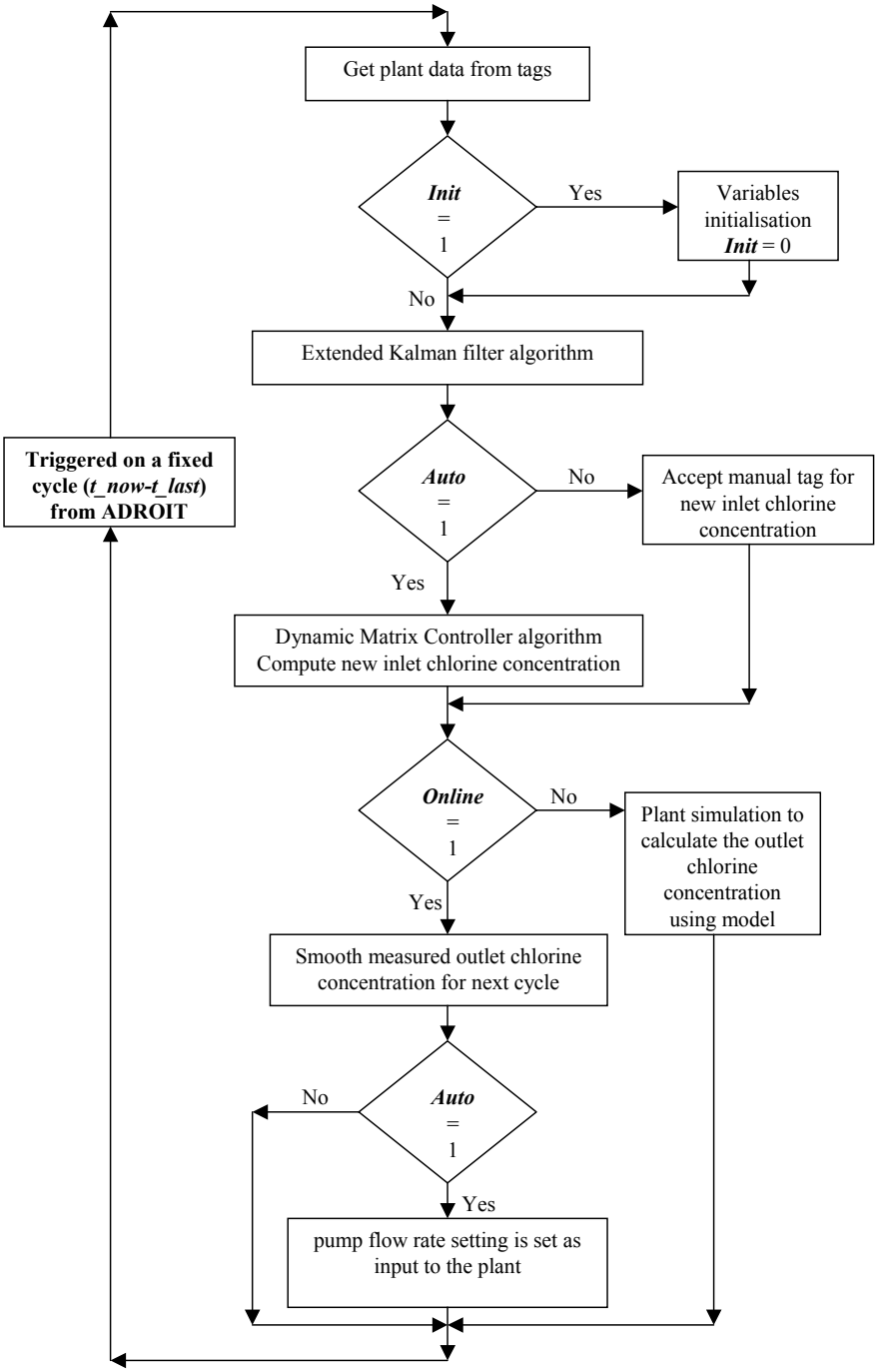
5.4.2 Dynamic Matrix Controller parameters

To run, the DMC needs the data from the previous call for the inlet and outlet chlorine concentration (***x0_data_last*** and ***xr_data_last***), for the calculated inlet chlorine concentration (***x0 mod last dmc***) and the matrix of the past inputs (***dm_past***). Finally, the inlet and outlet chlorine concentration calculated by the program (***Cl2_inlet_calculated*** and ***Cl2_outlet_calculated***) are set on User Interface screen.

5.4.3 Pump parameters

This update at each iteration the pump flow rate setting.

5.5 FLOW DIAGRAM OF THE PROGRAM



5.6 POSSIBLE PROBLEM

5.6.1 Time alarm

If the word: " ***Time alarm***" appears on the User Interface screen, that means that there is a problem in the setting of the time for the extended Kalman filter and for the Dynamic Matrix Controller. The time of the extended Kalman filter for the extended Kalman filter has to be greater than 1 and inferior to the time of the Dynamic Matrix Controller.

After solving this problem, the program has to be re-initialised by setting the ***initialisation*** factor equal to 1 again (noting that it will reset itself to 0 on the next cycle.

5.6.2 The program does not run or gives completely false results.

- Re-initialise by setting the ***initialisation*** factor equal to 1.
- The program is built to run with values non-equal to 0 for the level and the inflow of the reservoir. If these values are too close from 0, the program cannot run.
- If just one reservoir is used, the active volume area has to be divided by 2.