Combined Logical-Numerical Enhancement of Real-Time Control of Urban Drainage Networks

Hector Martín García
COMBINED LOGICAL-NUMERICAL ENHANCEMENT OF REAL-TIME CONTROL OF URBAN DRAINAGE NETWORKS
Combined Logical-Numerical Enhancement of Real-Time Control of Urban Drainage Networks

DISSERTATION
Submitted in fulfilment of the requirements of
the Board of Deans of Delft University of Technology
and the Academic Board of the International Institute for Infrastructural, Hydraulic and Environmental Engineering for the Degree of DOCTOR
to be defended in public
on Tuesday, 15 October 1996 at 13:30 h

by
HECTOR JESUS MARTÍN GARCÍA
born in Havana, Cuba
Master of Science in Hydraulic Engineering, IHE Delft
Now, if you believe that the universe is not arbitrary, but is governed by definite laws, you ultimately have to combine the partial theories into a complete unified theory that will describe everything in the universe. But there is a fundamental paradox in the search for such a complete unified theory... we are... free to observe the universe as we want and to draw logical conclusions from what we see. In such a scheme it is reasonable to suppose that we might progress even closer towards the laws that govern our universe. Yet if there is a complete unified theory, it would also presumably determine our actions. And so, the theory itself would determine the outcome of our search for it!... The discovery of such a theory... may not aid the survival of our species. It may not even affect our life-style. But ever since the dawn of civilization, people have not been content to see events as unconnected and inexplicable. They have craved an understanding of the underlying order in the world. Today we still yearn to know why we are here and where we came from. Humanity's deepest desire for knowledge is justification enough for our continuing quest.

Stephen W. Hawking, A brief history of time
Table of Contents

| ACKNOWLEDGMENTS | XI |
| ABSTRACT | XIII |
| NOTATION | XV |

Chapter 1  Introduction

1.1  Motivations  1
1.2  Optimisation for engineering systems. Foundations and historical background  3
1.3  Description of the undertaken research  10
1.4  The hydroinformatic framework  11

Chapter 2  Real-time Control of Unsteady Flow in Urban Drainage Networks

2.1  Description of the problem  15
2.2  Model based, real-time control of urban drainage network  18
2.3  Practical derivation of optimal control strategies based on both on-line measurements and forecasts  20
2.4  The problem of specification of default control strategies in real-time control systems using numerical optimisation techniques  24
2.5  An overview of potentially applicable techniques  26
2.6  Comparative analysis of existing methods  30
2.7  Hybrid solutions  33
2.8  The role of the forecasts in the optimisation process  34
2.8.1  Rainfall forecasts. The distributed rain feature  35
2.8.2  Surface runoff forecasts. The infiltration component  36
2.8.3  Pipe and channel flow forecasts. The use of the kinematic, diffusive and dynamic wave approximations  40
2.9  The economic model. Multi-objective optimisation  48
Chapter 3  Numerical Optimisation

3.1  Introduction 53
3.2  Mathematical background 55
3.2.1  The continuous form of the optimal control problem 55
3.2.2  The discrete form of the optimal control problem 56
3.2.3  Methods of solution 57
3.2.4  The optimal control problem in urban drainage networks 60
3.2.5  Necessary and sufficient conditions for the existence of an extreme of a region 61
3.3  Overview of unconstrained multi-variable search methods 62
3.3.1  Quasi-Newton methods. The matrix update system 64
3.3.2  The Broyden-Fletcher-Goldfarb-Shanno updating formula 65
3.4  The complementary single-line search subproblem 66
3.4.1  Introduction 66
3.4.2  The fibonacci search 68
3.4.3  Measuring line search effectiveness 70
3.5  The constrained problem 71
3.5.1  Introduction 71
3.5.2  Indirect methods for constrained optimisation 72
3.5.3  Direct methods for constrained optimisation 73
3.5.4  The feasible directions method 74
3.6  Finite-differences approximations to the derivatives 75
3.7  A word about errors and floating point arithmetic 78
3.8  The scaling problem 81
3.9  Closure 83

Chapter 4  Logical Optimisation

4.1  Introduction 85
4.2  The concepts of knowledge-based, real-time control 86
4.3  The domain independent reasoning knowledge 90
4.4  The domain specific reasoning knowledge 93
4.5  The factual knowledge 94
4.6  Object oriented representation 97
4.7  The rule based approach 98
4.7.1  Learning rulebases 100
4.8  Fuzzy logic 102
4.9  Closure 106
Chapter 5  The Combined Logical-Numerical Framework

5.1  Introduction 109
5.2  HYDNET_RTC. Overall description 110
5.3  The process of dynamic mapping of the non-linear economic model in HYDNET_RTC 113
5.4  Features oriented to real-time implementation in the numerical methodology 115
5.5  Levels of parametric optimisation 120
5.6  Detailed description of the combined framework 121
5.6.1  QNEWT: The constrained quasi-Newton optimiser 121
5.6.2  CFGEN: The cost function generator/editor/analyzer 122
5.6.3  CFCALC: The cost function calculator/evaluator/integrator 127
5.6.4  TUNING: The module to specify optimal parameters for the single line search 129
5.6.5  LEAD: The communications centre 130
5.6.6  PIPFLW: The interface with MOUSE ONLINE 132
5.7  The double blackboard architecture 132
5.8  The intelligent agent. Approaches used for decision-making 133
5.8.1  HYDNET: The knowledge base 135
5.8.2  LEARN: The learning section 137
5.8.3  UINFO: Module for post processing of the user supplied information 138
5.8.4  MASSBL: The mass balance in the system’s subnetworks 138
5.8.5  SHAPE: Resemblance of the natural behaviour of the system 141
5.8.6  INTERP: Convergence to a desired behaviour in the network 142
5.9  SmartElements-C working interface 143
5.9.1  KBRTC: The logical optimisation centre 143
5.9.2  CNVFRM: The format convertors 146

Chapter 6  The Real-Time Control Process with the Combined Logical-Numerical Optimiser

6.1  Introduction 149
6.2  Data specification. Trigger values 150
6.2.1  Specification of the dynamic information for the intelligent agent 150
6.2.2  The set of underlying relationship functions 159
6.3  Revealing internal links 160
6.4  The choice of the kind of optimisation. The real-time constraint 161
6.5  The chaining process 162
6.6  Output results 163
Chapter 7      Applications

7.1      Introduction 169
7.2      Example Nr. 1: A looped system 169
7.2.1    Results and comments 173
7.2.2    Preliminary conclusions. Test case 1 180
7.3      Example Nr. 2: A simple serial layout of regulators 180
7.3.1    The initial situation in the system 183
7.3.2    Results 183
7.4      Example Nr. 3: The Gothenburg urban drainage network 189
7.4.1    Brief description of the system 190
7.4.2    The initial situation in the system 193
7.4.3    Results 194

Chapter 8      Conclusions

8.1      Introduction 207
8.2      Some directions for further development 209
8.3      Final remarks 212

REFERENCES 217

ABOUT THE AUTHOR 223
Acknowledgements

This PhD dissertation, which started during the year 1991, in Delft, The Netherlands, during the first international course on Hydroinformatics at IHE, is the result of a joint research project between several educational and research institutions in Europe. Since then, a rapid development has taken place in the field of hydroinformatics. From a word often said with certain reticence when it all started, hydroinformatics has become a well-established and valuable conceptual-practical tool to solve modern technical problems in the aquatic environment.

Like every research, this research has been made possible by a number of valuable people, who would be almost impossible to enumerate here. I would like to thank all those people who supported me in one way or another during this study.

However, explicit mention should be made of those who had a decisive influence in the successful development of this study. This is the case of my main supervisor, Prof. Dr. Michael B. Abbott, who has been a constant source of inspiration and guidance. Also highly appreciated are his comments during the proofreading of the whole manuscript and the freedom and encouragement he always gave me to develop new ideas.

This thesis gained much from the series of inspiring discussions with a number of people among which I would like to mention my other two supervisors Dr. Lars K. Amdisen and Ir. Berislav Tomičić from DHI, who also performed a valuable contribution to the quality of the final draft with his remarks to the whole manuscript, specially to Chapter 3.

Among the employees of the River Hydraulic Division and the Hydroinformatic Centre at DHI, I would especially like to thank Ir. Ole Mark, not only for his comments to Chapters 1 and 2, but also for his valuable guidance during several ongoing projects at DHI in which I have participated during the time span of my PhD study. On the same note, I would also like to thank Ir. Niels E. Jørgensen, Arne Møller, and Nerkez Gavranović for their participation in one way or another in the tasks I developed during my PhD research.

I would also like to thank the direction of the River Hydraulic Division of the Danish Hydraulic Institute, and the staff of branch d (hydroinformatics) at IHE for their interest in this study. To the secretaries of the River Hydraulic Division at DHI, Susanne Kallemose and Hanne Schnøemann my very special gratitude for all their attention and support during this period.
To my wife, Brigitte, goes my deepest and very special gratitude for every single day in this journey. She was always there - in difficult as well as in good moments - to make it possible for me to look forward to the next working day. My family takes credit for being a constant source of inspiration and support as well.

Finally I would like to thank the International Institute for Hydraulic, Infrastructural and Environmental Engineering, the Danish Hydraulic and the Technical University of Delft for their essential support to this work. This support makes the present work possible.
Abstract

The pollution of natural recipients with emissions from combined sewer systems is a potentially dangerous, systematic and sometimes inadequately resolved kind of negative environmental impact on the water bodies of our planet. Briefly stated, it is possible to say that these emissions occur because during severe storm events the inflows entering the network on some occasions exceed the available transport and storage capacity causing damage to the surrounding areas in terms of flooding. Sometimes, the available wastewater treatment capacities are also exceeded, resulting in the spilling of certain amounts of partially treated or even untreated sewer overflows mixed with rain water, causing pollution of the natural recipients.

Among the several potential approaches to the solution of these problems, the real-time control of flow regulators in the network seem to be one which offers a rational and economic solution to this problem that is so essential to modern society. The introduction of real-time control in combined urban sewer-drainage systems aims at an optimal utilization of the available storage, transport and treatment capacities and thereby a reduction of undesirable effects by controlling the flow in the system by regulators.

An urban drainage network can be efficiently operated in real-time if the data about the ongoing process is combined with forecasts about the system state and is used in order to 'optimally' operate the flow regulators during the actual process, in order to minimise flooding and combined sewer overflows.

The aim when applying real-time control on an urban drainage system is the derivation of optimal control strategies for the active flow controllers in the network. These control strategies usually describe time series of setpoints or settings for the flow regulators which are obtained in order to prevent the above mentioned undesirable effects from occurring or at least to minimize their effects during the on-line operation of the system. Alongside with reduction of combined overflows, side benefits such as reduction of surface flooding, lower energy costs, flow equalisation, in-sewer sediment control, supervision and better understanding of the system’s operation can also be achieved.

The control of the pollution load is essential in order to maintain a certain quality of the receiving waters. This goal can in principle be achieved by minimizing the frequency of spills, their overflow volumes and occasionally by routing these spills to less sensitive water bodies surrounding the system.

The increasing availability and the decreasing cost of digital hardware and software has lead to an effort aimed at the development of sophisticated software for the operation of urban drainage networks which however, have not so far produced the desired results.
The present report describes the development of a combined logical-numerical framework which has been designed for the definition of control strategies for the real-time control of urban drainage networks. By applying one of the central ideas of hydroinformatics and semiotics (the proper merge of different approaches to solve complex problems) a prototype has been realised, which combines advantageous features of an intelligent agent (being again a blend of several knowledge-based techniques such as diagnosis, heuristics and fuzzy logic) and a discrete-continuous numerical methodology for the search of a constrained optimum in a non-linear function of many variables. In this way, the intelligent agent selects only the most promising scenarios for numerical evaluation, effecting a restriction of the 'search space' for the fully non-linear optimiser, which has proven to be essential for real-time implementation. Thus, the present framework approaches the solution in a process composed of several stages of increasing level of accuracy, which starts with a very fast, on-line logical optimisation step.

In addition to other - primarily academic - examples, this prototype has been simulated to be applied to the solution of the on-line control of the urban drainage network of the city of Gothenburg in Sweden during an extreme storm event, achieving satisfactory results. A reduction of nearly 54% of the global cost as defined by generalised surrogate, or synthetic reward units during the system operation was achieved there with the combined, multi-objective and non-linear optimisation process. A forecasted horizon of twelve hours was simulated with the help of the hydrodynamic models of the MOUSE system in approximately 47 minutes on a HP-700 platform, thus providing safe margins for real-time implementation in this case. Due to its general software design the framework can easily be extended so as to be applied to other real-time control of urban-drainage combined-sewer system.
Notation

∀  : The universal quantifier (for all)
∃  : The existential quantifier (there exists)
OCS : An acronym for Optimal Control Strategy
RS  : An acronym for Regulator Setting
RI  : An acronym for Rain Intensity
t  : Time
SSV : An acronym for System State Variable
S   : Variable representing the state of a system
C   : Cost associated to a control decision
A   : Variable representing a control action
CU  : An acronym for Cost Units
DB  : An acronym for DataBase
R(t)  : A rain hydrograph
Q_E : Evaporation losses
Q_w : Wetting losses
Q_i : Infiltration losses
Q_s : Storage losses
Q   : Flowrate
M   : Manning number
B   : Storage width
I_o, S_o : Bottom slope
I_f, S_f : Friction slope
y   : Hydraulic depth
A   : Cross sectional area
dt  : Time step
x   : Distance along a longitudinal axis (in relation to the flow)
β   : Momentum distribution coefficient
K   : Section’s conveyance
u, v, a : Velocities
h   : Water depth
g   : The acceleration of gravity (9.8 m/s²)
R   : Hydraulic radius
ζ   : Energy losses in general
Δx, h_F : Finite-difference interval
n   : A counter variable used for either time levels or number of 'events'
j   : A counter variable used for distances.
α_j, β_j, γ_j, δ_j : Pipe matrix coefficients
x^T : A column vector of n components x^T = (x_1, x_2, .., x_n)^T
t_0 : Initial time of an event
\( t_f \) : Final time of an event
\( \mathbb{R}^n \) : Euclidean space of ordered 'n-tuples' real numbers
\( \mathbb{R}_r \) : Space of all piecewise continuously differentiable functions from \([t_0, t_f]\)
\( \mathbb{R}^m \) : Banach and Hilbert spaces of square integrable and bounded measurable functions respectively.
\( \mathbb{R}^m \) : Mapping space for the Jacobian function's domain
\( \mathbb{R}^n \) : Mapping space for the gradient function's domain
\( A^{-1} \) : Inverse of the matrix A
\( A^T \) : Transpose of the matrix A
\( R(s,d) \) : A return function for the entities \( s \) and \( d \)
\( s, d \) : The incident functions or independent variables
\( \bar{s} \) : A transition function
\( \bar{u} \) : A control sequence
\( x^* \) : A stationary point of a multi-variable function
\( f \) : A general, multi-variable function
\( g(x) \) : A constraint function
\( \nabla f(x) \) : A gradient of the multivariable function \( f \)
\( H \) : The Hessian matrix of second order partial derivatives of the function \( f \)
\( D \) : Determinant of a principal sub-matrix
\( \alpha(k) \) : The step length along the search direction
\( \delta(k) \) : A measured offset between the search vectors in two subsequent iterations
\( \gamma(k) \) : A measured offset between the gradients in two subsequent iterations
\( I \) : The identity matrix
\( F_n \) : Series of Fibonacci numbers
\( \varepsilon \) : Fractional resolution based on the initial interval of uncertainty
\( I_0 \) : Initial interval of uncertainty
\( I_f \) : Final interval of uncertainty
\( \Psi \) : A penalty function
\( s \) : A penalty multiplier
\( s_T \) : A 'directional' search vector
\( \xi \) : An approximate quantity
\( e \) : An error function
\( \Phi \) : A non-zero value of the finite-difference approximation to the derivative
\( N \) : Set of nodes of a given network, \( N = \{n_1, n_2, n_j\} \)
\( E \) : Set of branches of a network, \( E = \{e_1(n_1, n_2), ..., e_j(n_{j-1}, n_j)\}\)
\( \zeta_j \) : A sensitivity function
\( p_j \) : An adjustable parameter in the learning set
\( M_j \) : An output function of the learning process
\( \mu \) : A membership function in the universe of discourse of the fuzzy sets
\( y_x \) : Classification parameter in fuzzy representation
FNLNOM : An acronym for Fully Non-Linear Optimisation methodology
\( \mathcal{R} \) : Representation hyperspace of \( m \) coordinate axes
\( \varrho \) : A mapping operator provided here by hydrodynamic simulations
\( \Omega \) : Solution hyperspace of \( n \) independent variables
PR : An acronym for pumping rate
WL : An acronym for Water Levels
C+ : A positive characteristic curve
C_ : A negative characteristic curve
\( n_s \) : number of stages used in the discretisation of the time-dependent vectors
\( n_f \) : number of experiments performed during the Fibonacci search
F : A single-variable function
a, b, c, d, e : Constants
CNVFRM : An acronym for format convertor
Chapter 1 Introduction

1.1 Motivations

Progress as we nowadays understand it, is not conceivable without science and technology. This is generally recognized by experts and non-experts in this field. As a result of his attempt to 'tame' the powerful forces of nature man needed to transform it, sometimes considerably. However, there are many examples to show that if not carefully applied, technology could become a dangerous boomerang against the most precious of our resources: the natural environment which makes our life possible.

The dangerous accidents of several nuclear power plants around the world, oil spills of dramatic consequences, contamination of soil and waters due to radioactive garbage, acid rains, holes in the ozone layer, illness and death of different animal species or even human beings due to many different forms of contamination, are examples which show the catastrophic consequences of this attempt to 'fight against' the colossal forces of nature, and certainly if these forces are not carefully studied. On the other hand, many examples are now present to support the statement that modern society and the natural environment can be integrated in a harmonious way. Not only that, but also that technology could and should be an instrument to prevent the approach of an environmental cataclysm rather than to accelerate it. This is the general, hydroinformatics, context in which this research has been conceived.

The pollution of natural recipients with emissions from combined sewer systems is a potentially dangerous, systematic and sometimes inadequately resolved kind of negative environmental impact on the water bodies of our planet. Briefly stated, it is possible to say that these emissions occur because during severe storm events the inflows entering the network on some occasions exceed the available transport and storage capacities in the network causing damage to the surrounding areas in terms of flooding. Sometimes, the available wastewater treatment capacities are also exceeded, resulting in the spilling of certain amounts of partially treated or even untreated sewer overflows mixed with rain water, causing pollution of the natural recipients.

Among the several potential approaches to the solution of this problem, the real-time control of flow regulators in the network seems to be one which offers a rational and economic solution to this problem that is so essential to modern society. The introduction of real-time control in combined urban sewer-drainage systems aims at an optimal utilization of the available storage, transport and treatment capacities and thereby a reduction of undesirable effects, such as untreated overflows or surface flooding, by controlling the flow in the system by regulators.
This results must be achieved with a minimal consumption of electrical energy for the operation of the network. Proper working conditions at the treatment plant are also an important item to be taken into account.

The aim when applying real-time control on a combined sewer system is the derivation of optimal control strategies for the flow regulators in the network. These control strategies describe time series of settings for the flow regulators which are obtained in order to prevent the above mentioned undesirable effects from occurring or at least to minimize their effects during the on-line operation of the system.

Among these undesirable effects, the first objective is often the control of the pollution load in order to maintain a certain water quality in the receiving waters. The two ways to achieve this goal are by minimizing the frequency of spills and their overflow volumes.

The increasing availability and the decreasing cost of digital hardware and software has lead to an effort aimed at the development of sophisticated software for the operation of urban drainage networks. This development has become particularly intense during the last ten years. Unfortunately, in most cases those efforts have not produced the expected results. The reasons for this are often attributed to the excessive complexity of the problem for its solution in real-time, the non-generality of obtained solutions, and in some cases to the lack of suitable models for its representation, among some others besides.

An accurate, general, real-time and logically and numerically feasible solution is required. That was the main premise for starting this research. The latest works in the field of artificial intelligence and optimization were the main sources of information. The desire of contributing to human welfare, progress and environmental protection was, above all though, the main source of inspiration.

This research has been carried out within the framework of a join venture between the International Institute for Hydraulic, Infrastmctural and Environmental Engineering (IHE) in Delft, The Netherlands and the Danish Hydraulic Institute (DHI) in Hørsholm, Denmark. Intense research is performed in both institutions on fields related to hydraulics and environmental protection under the new perspective offered by hydroinformatics.

---

* * It is a must to use the words of the author of the new concept, Prof. M. B. Abbott, in order to define hydroinformatics: "Hydroinformatics is concerned with the information flows that accompany and govern the flow of fluids and of all that these fluids transport. However, even if only because common-sense truths have also to be accommodated within hydroinformatics, this subject cannot itself be a science, even though it draws upon many sciences: it is, in a very essential sense, a technology."
1.2 Optimisation for engineering systems: foundations and historical background

This work draws upon several disciplines. An attempt has been made here to combine and implement the best features of each of several of these in order to produce a real-time, multi-choice, hybrid optimisation system developed within the framework provided by the MOUSE ONLINE hydroinformatic system.

In principle each of the independent optimisation systems developed by using the different approaches can be triggered by the MOUSE ONLINE real-time control module depending on the characteristics of the particular hydrodynamic network to be controlled; even its results could be combined or complemented. The resulting real-time control system has been fully conceived for real-time implementation, which means that the time consumed by the optimisation process has been reduced to the absolute minimum without sacrificing the required accuracy, thus leaving safe margins for real-time implementation. In this respect it should be observed that we are not concerned with any kind of 'absolute optimum' of the kind that is treated in mathematical texts, but only with approximate optima, such as will effect substantial economies and provide major enhancements in the natural environment. When it is necessary to emphasize this feature, we should speak of quasi-optimisation.

The general strategy used here for real-time control could be described as follows:

The real-time derivation of the quasi-optimal control strategies for the flow regulators in a given hydrodynamic network is to be understood as a process in which the result is achieved in successive stages. Each stage leads to a higher level of parametric optimisation than the former one. The number of stages triggered depend on the real-time available in the real-world situation. If the hardware platform can support such an operational mode, several optimisation processes could be run in parallel and the results used in a proper combined fashion. Whenever possible, rather definite quantitative criteria must be preferred for evaluating the consequences of a given control strategy on the system.

The characteristics of the hydrodynamic network to be controlled might suggest the kind of quasi-optimisation to be applied on the system. For example, a very large hydrodynamic network with a relatively high number of regulators which is to be quasi-optimised over a long period of time (forecasted horizon) might suggest either a logical kind of optimisation or a numerical optimisation only on the main regulators in the system, in which the less relevant regulators are left to the local (PID) controller. It must be remembered that a solution that is out-of-date when obtained is irrelevant to real-time control, regardless its nominal accuracy. In this case we may say that, although accurate, it is irrelevant.

Two main different kinds of quasi-optimisation systems have been developed within the framework of this research with the purpose of real-time control of hydrodynamic networks. They are a fully non-linear, numerical quasi-optimisation system and a knowledge-based quasi-optimisation system. Both have already been applied to the real-time control of urban-drainage combined-sewer networks.
In principle, these quasi-optimisation systems have been developed with the more general purpose of real-time control of hydrodynamic networks.

Numerical optimisation for engineering systems in general capitalises on the structure of the problem to obtain formal proofs of global or local optimality and to develop efficient algorithms for locating best values of the economic model while still satisfying constraints. This topic will be discussed in detail in Chapter 3.

According to Pike (1986) the objective of optimisation "... is to select the best possible decision for a given set of circumstances without having to enumerate all the possibilities. From experience designers learn to recognize good proportions and critical restrictions, so their preliminary work will not require significant modification and improvement."

Scientists, especially mathematicians, have always been occupied with questions of pure mathematical optimisation, i.e., finding extrema (maxima and minima). Euclid in 300 B.C. was associated with the problem of finding the shortest distance that could be drawn from a point to a line, and Heron of Alexandria in 100 B.C. studied the optimisation problem of light travelling between two points by the shortest path. Only did Fermat in 1657 develop the more general principle that light travels between two points in a minimum time. In 1857 Gibbs developed the principle that a system is in chemical equilibrium if its free energy is a minimum. The first modern book on the subject of optimisation was written by Hancock in 1917. This definite work is still used today as an authoritative source. After 1940, Dantzig recognized the mathematical structure of some military logistics problems and developed the widely used simplex algorithm of linear programming. But in the 1950's the subject of optimisation received a considerable boost with the advent of the space era. The optimal trajectory for a missile was one of a number of problems for which the methods of dynamic programming were developed.

When solving an optimisation problem by making use of numerical techniques, the structure and complexity of the economic model for the problem and the problem constraints are very important and most numerical computer algorithms take advantage of the idealised, mathematical form of these models.

The main areas in numerical optimisation are mathematical programming and variational methods. Mathematical programming aims at finding the best vectors which optimise the economic model for the process while variational methods are intended to find the best functions that most efficiently optimise this economic model.

Mathematics, since the invention of the differential and integral calculus in the seventeenth century, has worked out successfully the concept of a vanishingly small increment of time with which to model continuous development in time. In the case of numerical hydrodynamic models, digital computers, being fundamentally manipulators of discrete binary sequences, replicate continuous time development as a series of discrete 'pictures' of the state of the system at certain predefined time levels. This sequence of pictures is contained in the so-called time series of the system state variables.
On the other hand, the historical background for logical optimisation is found (Steels, 1989) to be related to an old dream of mankind: the goal of representing human knowledge in a form that would support problem solving in a rather algorithmic way.

Back in the 13th century a Spanish scholar, Lullus, invented the *Ars Magna*, a system of rotating concentric circles. The Lullus device is illustrated in figure 1.

Each circle contained basic, or primitive concepts and the position of the circles represented a combination of these primitives which was supposed to be ‘true’.

New inferences were made by spinning the wheels, sometimes leading to unexpected new combination of concepts. This device, which became very popular through Europe at the time, was used not only to pose philosophical and theological questions, such as the conflict between free will and predestination, but also to represent medical and common-sense knowledge.

Although itself primitive, this, basically epistemological, device played an important role in the later efforts in this field. However, at the time that it was developed the conditions for its application were not created and the ideas of epistemology that lay behind it were temporarily forgotten.
The Lullus device was recalled as a source of inspiration by Leibnitz in the 17th century. He had already built a numerical calculator based on the emerging technology of sophisticated clocks. Leibnitz began a project to construct a universal symbolic language and an equally universal inference machine with which *new knowledge could be derived from facts*. Although far from reaching their proclaimed goal, his ideas were very influential and provided a foundation to the later developments of symbolic logic. As one part of this development, Leibnitz introduced and developed the principles of *binary arithmetic* in detail.

Several scholars, including Leibnitz, saw these machines as primitive models of the intelligent mind. Leibnitz developed his *theory of monads*, as primitive intelligent beings of the kind that we shall later identify with agents, alongside this line of inquiry. Around the same time, the first philosophical speculations exposing the latter view started to emerge. This matter was discussed by the French philosopher Delamettrie.

Meanwhile symbolic logic, and in particular the propositional calculus and formal representations of the Aristotelian syllogisms, continued to be developed as tools for knowledge representation in the 18th and 19th centuries, often inspired by, and following Leibnitz. In the 19th century several machines were built performing logical deductions in a mechanical way. An example is found in Jevenson’s logic machine in Great Britain, of 1870. This machine was inspired by the propositional logic of Boole. As input the machine would receive logical expressions and reduce the number of possibilities of certain facts occurring to the ones logically compatible with the given expressions. The individual symbols represent propositions and their occurrences as pairs in the individual registers correspond to their combinations. See, further, Steels, (1989). Jevenson’s logic machine is illustrated schematically in figure 2.

Formal languages were developed to overcome some of the apparent 'drawbacks' of natural language (such as its vagueness and its appeal to presuppositions). As an illustration of the above a simple example is often used: the phrase "unicorns do not exist" already talk about unicorns as if they exist. A more precise way of stating the same thing would be: "For all $x$ such that if the predicate unicorn applies to $x$, then it is not true that the predicate exists applies to this same $x$". The limitations to this formalisation of propositional relations and the reason for the supposed vagueness in the vernacular language is that the word 'exist' has itself a wide range of connotation (and in this case no definite denotation).

Thus, for example, although the phrase 'unicorns do not exist' is true in a zoological sense, unicorns remain of great significance in all manner of myths, and they appear also as mythical elements in legends and sagas. Thus, in the depth-psychological sense, they do have a kind of 'existence'. This difficulty can be avoided (but not of course overcome) by saying something like 'unicorns do not exist, but they function' see Foucault (1970), and in relation to applications in hydrology, ecology and geography see Abbott (1992).

The advent of the digital computer made it possible to start synthesising very complex 'logical' systems. In the late 1950's, Artificial Intelligence emerged as a field concerning the development of systems which are capable of effectively solving real-world problems by using a machine represented 'intelligence'.
Artificial Intelligence comprises many fields such as natural language communication, motor and control vision, machine learning, etc. The automation of the process of logical inference has become one of the most important results of Artificial Intelligence. For example, it has become possible to represent logical formulae inside computer systems and to build automatic theorem-provers which executed the logical inference rules that logicians have proposed.

Among the factors which still limit the success of Artificial Intelligence in modelling human expertise it would be possible to point out that natural problem solving often involves handling contradictory information, reasoning from incomplete knowledge, and reasoning with uncertainty. Strict logical inference is not designed to handle these phenomena because the original objective was seen as one of dealing with 'absolute', consistent and complete knowledge. There is also a problem which arises when trying to express logical inference in predicate calculus. Gödel has shown that logical inference for predicate calculus is undecidable, meaning that it is not possible to come up with an algorithm which will always find out whether a logical formula is true or not given a set of axioms.

Some theorem-provers do in fact demonstrate combinatorial explosions of possibilities. Moreover, Gödel's work shows that although the existing non-deducible problem now become deducible in this way, the door is also opened to new non-deducible problems, and that this process can continue 'without end'. This problem can only be overcome in such situations by
adding more knowledge, which is not always available. Another 'solution' to this problem is so to restrict the logical language that it becomes more like a traditional programming language which is so to say, totally enclosed within a closed symbolic 'box. The 'Physical Symbol System' of Newell and his coworkers (eg. Newell, 1983) provides a formal description of all such systems in general terms. The widely spread language PROLOG is a good example of this approach. Let us use simple examples in order to illustrate the representation of a certain linguistic expression in predicate calculus. For example, the sentence "All blocks are small" can be represented in traditional notation as:

$$(\forall x) \ [ BLOCK(x \rightarrow SMALL(x) ) ] . \quad (1.1)$$

Or in list structure notation as:

$$(\text{ALL} \ (x) \ ( (IS\,-\,A \times BLOCK) \rightarrow (SMALL \ x)) ) . \quad (1.2)$$

in the same fashion our statement about "Unicorns do not exist" would become - in traditional compact notation - :

$$(\forall x) \ [ UNICORN(x \rightarrow \exists (x) ) ] . \quad (1.3)$$

Or in list structure compact notation as:

$$(\text{ALL} \ (x) \ ( (IS\,-\,A \times UNICORN) \rightarrow (\exists (x)) ) . \quad (1.4)$$

The strength of predicate calculus is that has well-understood interpretations to express many different sentences (well formed formulas or wffs). These formulas make use of the universal quantifier "\( \forall \)" (for all) and the negation of the existential quantifier "\( \exists \)" (there never exists). These two quantifiers of predicate calculus have also became an important representation tool for many of the logical languages and shells which exist today.

Expert systems are knowledge bases encoding or encapsulating the knowledge necessary to perform problem solving in a given domain. These systems manipulate knowledge in order to perform certain tasks. According to Hayes-Roth & Waterman (1983) "... an expert system is one that has expert rules and avoids blind search, performs well, reasons by manipulating symbols, grasps fundamental domain principles, and has completer and weaker reasoning methods to fall back on when expert rules fail to perform and to use in producing explanations ..."

Due to its relevance to this work, the detailed characterisation of knowledge and of its representations will be discussed in sections 4.1 to 4.4. At this moment it is probably enough to define the knowledge representation formalism as the set of descriptions, relationships and procedures that hold within a certain universe of discourse.

It is also important to emphasise that the knowledge encoded in a knowledge base is really only highly structured symbolic data representing a model of the relationships between data and the procedures to be applied on them.
The discipline of 'knowledge engineering' associated to the development and use of expert systems has had a considerable impact in many areas of human activity where such formalisable knowledge provides the power for solving important problems.

According to Efstathiou (1985, p. 44), the non-optimality, the impreciseness and the even inconsistency present in many [human derived] control strategies are two good reasons for seeking to automate the control of ill-understood process. In order to implement such ill-defined strategies the original rule-based controllers make use of fuzzy logic.

This logic was devised to cope with such problems, based on the premise that much human decision-making activity was based on sets which did not have the sharp, well defined boundaries usually associated with mathematical reasoning. Instead, fuzzy logic accommodates elements which are partial members of a set with a grade of membership lying somewhere between zero and one. Therefore fuzzy logic provides a representational tool for the so-called α logic problems to which we shall return in section 1.4.

The power of rule-based controllers resides on their inference mechanisms. The inference mechanism combines the knowledge encoded in the knowledge base with data from the outside world to produce conclusions and explanations. Rather more precisely, it is by these means that encapsulated knowledge identifies information as data and acts upon it.

The characteristics of the process control make it possible for rule-based systems, to avoid generating and testing hypotheses so that in some cases it is possible to use a forward (data driven) reasoning mechanism.

The distinctive feature of rules expressed using a fuzzy logic is that they have a region of influence, that is defined by the spread of the fuzzy sets which define the linguistic terms. Only those rules with a degree of fulfilment greater than zero will be triggered. Therefore fuzzy rules are not mutually exclusive and more than one rule may contribute to a particular control action. Then a 'defuzzification' method (section 4.7) has to be used in order to translate the fuzzy command into a specific control action.

Problem solving in process control necessarily implies taking advantage of specific (site-dependent) knowledge 'available' for the domain (Guida, 1985, p. 136). The problem of analysing different levels of knowledge representation seems to have been first realized independently by Hart and by Michie (both in 1982). They provided a distinction between 'deep' and 'shallow' knowledge. Shallow systems are characterized by having no explicit representation of concepts such as causality, intent or basic physical principles. They mostly rely on empirical associations between data and partial conclusions of interest, stimuli and responses, actions and effects. Shallow knowledge often becomes 'too large' to deal with in an efficient way in our domain of interest. On the other side, 'deep' knowledge comes from observing a system 'from inside', and discovering explicit causal relations and patterns of operation and general behaviour. Deep knowledge thus contains a generalised representation of the domain and therefore is of central interest here also.

Even within a single individual application domain (Barnett 1982), there may be specific tasks requiring the use of deep knowledge and others which can be more effectively accomplished
through shallow knowledge. In general terms it is possible to state that shallow knowledge is appropriate to represent causal relationships between facts that are otherwise apparently unrelated.

Deep knowledge is considered to be structured, certain, theoretical, domain-independent, general, broad but non-effective, while shallow knowledge is specific, narrow, approximate, domain-dependent, fragmentary, empirical but very efficient. This is known as the knowledge representation paradigm. The kind of knowledge to be used for problem solving in a specific domain depends on the structure and characteristics of knowledge in the domain.

1.3 Description of the undertaken research

A description of this research work, as carried out at the Danish Hydraulic Institute in Hørsholm, Denmark, will be provided here in terms of objectives and scope. The present research can be placed within the overall framework provided by the European SPRINT real-time control project SP226 launched in early 1993. The present work is in the first place a hydroinformatics study that could be applied in other fields besides, but in this particular case is applied to urban drainage systems. Within this context, the main objectives of the present study were:

i) To evaluate the potential of several integrated logical and numerical computational techniques for providing a satisfactory solution to the problem of real-time control, as applied in the present case to urban drainage networks.

ii) To develop a general and sufficiently accurate hybrid (integrated logical and numerical) framework able to perform quasi-optimal, real-time control of such networks.

iii) To implement techniques (preferably those making use of knowledge about the domain) in order to reduce as much as possible the computational load for the fully non-linear numerical optimiser.

iv) To investigate the suitability of the choice of the control variables (such as, within the urban drainage context, the discharge at the controllable structure, the setting of a regulator, the speed of movement of a valve opening, etc), with consequences for the treatment of the constraints imposed within the numerical optimisation system.

v) To provide a solution to the problem of finding the steepest descent direction in systems with correlated flow controllers.

vi) To provide tools for quantitative evaluation of the costs as expressed in generalised cost units or reward criteria associated with a given control strategy.

vii) To show the potential of object orientation at all levels of process representation and data modelling in non-linear optimisation.
The full accomplishment of these objectives implied the solution of numerous sub-tasks along the way leading to the implementation of the desired real-time control system. The prototype control system discussed here has been successfully tested in two experimental and one real-world urban drainage network, as exemplified later.

The rapid development of new techniques with application to the field of optimisation, specially those related to the different fields of Artificial Intelligence and computational genetics make it possible to predict within a reasonable level of accuracy the future development of these systems. Still, numerical optimisation techniques offer features such as accuracy and generality to an extent that none of the other approaches have so far provided for the problem of effective, general quasi-optimisation of the real-time control of such networks as are found in urban drainage systems. This matter, which is obviously of central importance, will be discussed in sections 2.5, 2.6 and 7.2.

1.4 The hydroinformatic framework

In his "Hydroinformatics, information technology and the aquatic environment", M.B Abbott (1991, p. 29) points out that:

"The pragmatic task of hydroinformatics is to combine and integrate all standard scientific information and further, all common-sense truths to the extent that these truths can be given a standard-scientific informational representation. An initial feature of such a hydroinformatic system is that it allows the use of numerical simulation subject to constraints that are expressed in natural language. When seen as a knowledge-based system, the hydroinformatic system provides the usual possibilities for its user to pose questions such as 'how?' and 'why?' and 'when?'"

In order to proceed further it is necessary to provide a description of the components which are found in hydroinformatic systems. This can also be found in the above mentioned reference (p. 31).

"...They commonly constitute combined knowledge-based and algorithmic-based systems for both the off-line and the on-line design, control and management of aquatic environments.

...The principal pragmatic task of hydroinformatics is to reduce the total costs to society and to ecosystems. In so far as these operations interact with the aquatic environment. Thus in many cases hydroinformatic systems will incorporate optimisation procedures.

...The most common form of the hydroinformatic product, to date, is the expert advice system.

...General hydroinformatic tools will be designed in future with the flexibility to answer to different concerns.

... measuring and operating systems become as far as possible interfaced electronically to the domain knowledge encapsulator and its frame. There is then a two way communication:
Monitoring systems may generate signals (alarms) to the system and the system may request information from subsystems as well as from its information and knowledge bases.

"The main users of such systems are those charged with the management of the aquatic environment."

"Real-time events interact with these systems and they will themselves respond using hierarchical system of priorities in each application area.""

The above explanation makes it possible to state that MOUSE ONLINE is a hydroinformatic system because it fulfils the requirements listed above. This system is a typical product of hydroinformatics and the fifth generation modelling. It is also a new product of the new concept. It is something for which modern society is getting prepared for; to fully understand it, to apply it and to enjoy its benefits.

As it is possible to derive from the above discussion the optimisation sub-system is an important part of the overall hydroinformatic system.

The problem of optimisation of the real-time control of urban drainage networks is a centre of attention at many research institutions all over the world. Legislation for the protection of the environment imposes strong limitations on combined sewer overflows from these systems both in terms of volumes and frequencies. A correct operation of the urban-drainage combined-sewer network is related to an efficient use of the storage capacity. Real-time control based on the operation of active flow regulators is becoming the world-wide standard (and rational) solution for this important problem.

In spite of the many attempts at implementing effective control systems, a lack of such products is still observed, partially due to the need for new and more effective concepts for the real-time control of urban drainage networks. These concepts are now provided by the integrated hydroinformatic systems of which this is a fair example.

Under this new perspective we understand for example the word 'model' as "a collection of signs that serve as a sign" (Abbott, 1992). Extending the concept we can understand model based control as ".. a technique which operates upon collections of signs in order to gather useful knowledge about an observable environment which in turn will be used to improve its behaviour in terms of well defined quantitative criteria."

(Martin, 1995).

Real-time control systems must be able to monitor and respond to ongoing processes in the real world in the face of time (or other) constraints.

In an urban drainage network the operational objectives are typically:

i) to minimise environmental damage due to combined sewer overflows

ii) to reduce surface flooding

iii) to equalise inflows to the treatment plant

iv) to reduce operational and maintenance costs.
Although the above mentioned objectives are of great importance, they are not alone. In effect, other 'secondary' objectives may well appear to be just as important (or even more important) in some particular real world problems. These often conflicting objectives give rise to the need for multiple decision making.

Cost functions are used with the purpose of quantifying each objective in a relative sense in such a way that to minimise the cost functional over a certain time interval is to achieve optimal control. Dealing with conflicting objectives occasionally might mean to sacrifice minor objectives in order to achieve the most important ones in the sense of costs.

"Decisions rely on information .." (Nielsen et al, 1992). Numerical models transform collections of signs (input information) in order to predict a behaviour. It can be easily understood that if the information on which they work is not accurate, a non-satisfactory result is to be expected. In a sense, this fact become even more critical in the case of optimisation systems which use numerical models in order to produce forecasts. In effect the further we go up in the process of 'integration of software' the more critical the need for validating input information. The use of remotely transmitted on-line information from sensors provides a unique possibility to produce reasonably accurate forecasts and therefore optimal control decisions. The possibility of correction of forecasts with on-line information has a considerable impact in the quality of the control and will be discussed in detail in section 2.3.

Let us close this section with a word about logic and numerics connected to the third cycle in the characterisation of hydroinformatics (Abbott, 1991).

As known, a large part of Artificial Intelligence is currently taken up with symbolic processing. McCarthy (1979) has described the LISP language as "computing with symbolic expressions rather than numbers .. control structure based on the composition of functions to form more complex functions .. recursion as a way to describe processes and problems .."

The LISP example has been used (Abbott, 1991) to illustrate the so-called symbolic paradigm in relation to symbol manipulation codes in Artificial Intelligence and the interpretation of their results in terms of an average intelligibility within the universe of discourse. As pointed out in the same reference, the symbolic paradigm has still limitations in Artificial Intelligence in the sense - for example - that an expert system cannot explain the cause of its own failures.

According to the symbolic paradigm, logic and symbol manipulation are viewed as appropriate descriptions only of the few cognitive processes that explicitly involve logical reasoning. Related concepts are those of $\beta$ logic which deals with completely connected systems of thought and $\alpha$ logic dealing with what cannot be formulated precisely as concepts, but which remains instinctual, or prelinguistic or prepredicative and thus prescientific.

Returning to our field of interest of model based control it is possible to describe this phenomenon by using M.B. Abbott's words (1991) summarizing the experience of many investigators, namely the fact that "the model understand the world in the way that it does is something that we can always describe scientifically but never thereby truly understand .."
The ideas expressed by the symbolic paradigm are of central importance in the present study since they provide the means to effect a logic in a digital computer. Knowledge based systems using fuzzy logic are a good example of the above.

As a matter of fact, it is the practical knowledge (although heuristic, fragmented and ill-structured) that deserves our special attention in the domain of urban drainage networks since it provides a unique source of knowledge in a world which does not show complete regularity in the domain.

The 'logic behind the numbers' which is considered to explain (for example) most of the instabilities observed in numerical modelling is again enhanced by the discipline of hydroinformatics and arises in a totally new dimension. It is common to find (see Abbott & Basco, 1989) that "the model comes to teach us things of which we were previously unaware, even though the model is itself entirely a product of our minds .."

Expressing a personal point of view, this phenomenon appears to be related to the well known phenomenon of the often unconscious feature of the human knowledge.

There are limits to what can be modelled. On the other hand individual measurements are only indirect (partly for isolated) representations of the system's state. If the modeller (Crutchfield, 1993) interprets the measurements as a the actual state of the system it will unwittingly be forced into a class of computationally less powerful representation. Using his own words ".. mathematical objects, with which one typically does not associate meaning, do imply a semantic structure for the act of measurement. And it is this semantics that gives models their scientific value .."

Information is currently defined nowadays as that which resolves an state of uncertainty. A great part of the formal description of our real world is reduced (for modelling) to sequences of binary elements of the form:

\[ 010010110101000110100... 01001 \] (1.5)

The feature of knowledge encapsulation as one of the most sophisticated ways of coding is rapidly becoming an important part of modern technology.

The new technology ".. concerned with the information flows that accompany and govern the flow of fluids .." provides a powerful perspective and the necessary tools to deal with old problems in a completely different - and more efficient - way. Therefore, its concepts have been extensively used as foundations for this work.
Chapter 2  Real-Time Control of Unsteady Flow in Urban Drainage Networks

2.1  Description of the problem

In general, the runoff to an urban-drainage combined-sewer network is influenced by precipitation - and occasionally snowmelt -, sewage inflow, infiltration from the surrounding soil, surface topography - and its topological characteristics -, evaporation, and some other factors that are usually of lesser importance. These parameters allow us to assemble the so-called interference vector which constitutes the input to the system (Fuchs et al, 1987, p. 272). On the other hand the vector of the system state is composed by the set of measured and/or computed parameters given a certain control strategy for the flow regulators (pumps, gates, weirs, valves) of the system.

The 'cost hypersurface' or economic model for the problem is defined, in this case, by the integration of all surrogated costs (as expressed in cost units and not in real currencies) caused by a given control strategy applied on the system. As examples we could mention surrogated costs resulting from sewage overflow, costs of surface flooding (occasionally street flooding), energy and operational costs, depreciation of equipment, waste water treatment costs and so on.

The vector of constraints for the on-line control of urban drainage networks is imposed by the set of physical constraints applied to the system such as the available storage volume in detention basins, operational boundaries of active flow regulators, pump capacities, network topology and, essentially, by the real-time constraint. In practice, the problems encountered in finding the optimal control strategy are introduced by the following restrictions:

i) The actual state of the interference vector is unknown for the following time steps (it can only be approximated).

ii) The vector of the system state is a non-linear function due to the stochastic characteristic of the interference vector and also due to the control decision.

iii) The vector of constraints and the economic model are often defined by non-linear functions.

iv) Some cost functions like sewage overflow or surface flooding are rarely given but they have to be defined for a particular optimisation methodology.

v) The system state variable might be partly unknown because of incorrect or incomplete measurements or calculations.

* In the case of the boundaries for the flow regulators, the structure of the function strongly depends on the choice of the control variable to be optimised. See for example (Tomicic, 1989).
The above mentioned facts together with the *real-time constraint* are the main reasons to consider the problem of real-time control of urban drainage networks as a quite complex optimisation problem. Moreover, phenomena such as 'loop effects', spatial interactions between regulators, serial layouts of controllable devices, etc, may also intervene and need also to be carefully analyzed. They all contribute with their own share of additional complexity to this problem.

In order to proceed with the description of the characteristics of the inflow event in an urban drainage system, let us now define the above mentioned phenomena of the 'loop effect'. By 'loop effect' we understand the phenomena by which a *backwater curve* (primarily defined by a \( C_\) characteristic curve) is generated from a given downstream section of limited conveyance at the junction in node N4 (internal boundary condition) thus affecting the node N1 through two branches: the direct connection \([N1, N4]\) and the side loop defined by \([N1, N2, N3, N4]\). Necessary conditions for the existence of such a 'loop effect' are: i) there is a looped network ii) initial flow is directed from N1 to N4, and iii) there is subcritical flow condition (as supercritical flows are not affected by the downstream boundary conditions). In practice, this effect is often observed only occasionally, according to the development of the inflow event. This phenomenon is illustrated in figure 3.

![Figure 3: Illustration of the 'loop effect' in looped networks with subcritical flow](c:\phd\loop.wpg)

The structure of the network in itself can in general be dendritic (also called branched or tree-like) or looped. Real world urban drainage systems are often a combination of the two, with branches in which 'loop effects' are occasionally observed in relation to the state of the flow.
As explained above, the inflow to urban-drainage combined-sewer systems consists, on the one hand, of a basic sewer flow - domestic and industrial wastewater - combined with the dry weather hydrological contribution (infiltration from the surrounding soil, which is approximately uniformly distributed in time) and, on the other hand, a surface runoff/precipitation component which is usually very irregularly distributed in time. The latter component is, in principle, the most significant in magnitude so that the centre of attention in real-time control of urban drainage networks is in some cases focused upon a relatively short time span of high intensity rain storms. One of the reasons is that during severe storms, the flow entering the network may exceed the available transport (conveyance) and storage capacities, causing damage to the surrounding areas in terms of flooding. In the same vein, it is observed that if the available wastewater treatment capacities are exceeded, the untreated (or partially treated) sewer overflows could cause damage to the natural recipients located in the neighbourhood of the network.

This is only one side of the problem. The other side is related to operational and maintenance costs. If the existing treatment plant is sensitive to increases in input water levels over an extended period of time, or if an enlargement of the plant is not feasible there might be an interest in equalising the inflows. Otherwise an increase in the operational costs may be observed due to the potentially 'excessive' consumption of electrical energy in the pumping stations. The maintenance costs which are often associated to the working condition of the network after severe rain storms, may also be increased.

Real-time control based on the operation of active flow regulators is recognized as one possible solution to this important problem. The classical (and generally expensive) solution to the problem was to invest in order to physically increase the storage capacity in the system by building large detention tanks and increasing the size of the sewer mains, among others. This situation started changing only in the last five to ten years.

The main idea when performing real-time control of urban drainage networks is to make a better, because more dynamic, use of the storage, transport and treatment capacities which exist in the network. In this way a temporary storage of water - during and shortly after the rainstorm- is achieved in a dynamic and integrated manner, thus preventing, or at least decreasing sewage overflows, so that the existing sewer urban drainage network comes to function in another way, see (Abbott, 1991, p. 8).

As it can be derived from the above explanation necessary conditions to implement this approach are the existence of a number of active flow regulators and an appropriate automatic device for derivation and implementation of the optimal control strategies. In such a framework, an optimal control strategy is defined here as a set of time series of settings -covering the so-called forecasted horizon - of each of the flow controllers which will result in a maximal utilisation of the storage capacity in the network and correct operational regime, thus reducing as much as possible all the undesirable effects in the system.
2.2 Model based, real-time control of urban drainage networks

The MOUSE ONLINE concept for the derivation of optimal control strategies in real-time is related to the use of both remotely transmitted on-line measurements of rainfall and the flow situation in the system, as well as the use of numerical hydrodynamic models, in order to predict the future development of the flow situation (runoff and pipeflow forecasts). Therefore, there are two sources of information about the past and present state of the system and in principle one main source of information about the future stages of the inflow event. The conceptual aspects about this contribution are schematized in figure 4.

In this framework the online information about the state of the system is used in order to correct next the forecasts. In the case of large observed discrepancies between measured and forecasted parameters, a fault diagnosis module is required in order to identify the cause of the discrepancy and decide the further course of actions to be taken (Babovic, 1991 and Amdisen, 1992). As pointed out by Lindberg et al (1990), this considerably increases the robustness and reliability of the real-time control system.

As pointed out by Tan et al (1990) and Schilling (1991), the use of forecasts of the flow situation improve the control since the control action to be taken will depend not only on the present situation but also in the expected one. In order to produce the flow forecasts the numerical hydrodynamic models use a rainfall prediction based on on-line measured precipitation in the catchments. An architecture which combines on-line measurements and forecasts as suggested by Lindberg et al (1990), is shown in figure 5.

In general it is possible to state that the application of a real-time control technology to a given system involves the following types of problems (from Schilling 1991):

i) Legal problems of compliance with standards, damage compensation and other disputes.

ii) Liability problems due to obtaining a global improvement at the cost of creating new local problems.

iii) Organisational problems.
iv) **Institutional problems:** traditionally urban drainage networks are operated by different groups among which collaboration may initially prove difficult.

v) **Different kind of technical problems,** among which the lack of quality software tools for design and operation of real-time control systems is one of the most critical.

Due to its relevance to our further discussion, let us now put forward a few words about the use of models of any kind in decision-making processes and the way of evaluating their performance.
As explained in section 1.4 simulation, in the sense that we understand this here, is a modelling process in which a dynamic reality, either actual or projected, is imitated in terms of computer actions. As a discipline matures, its established models (paradigms) are continually being compared, improved or discarded, according to their usefulness.

According to Evans (1988, p. 11) by using models "... we develop a framework of expectations which enables us to reconstruct reality in a coherent way, upon which basis we can make decisions ... The model is the matrix for the piecing together of otherwise disconnected experiences."

We should at this point observe the definition used in hydroinformatics (Abbott, 1991), that "Reality is the name that we give to the interface between our inner and our outer worlds", to which is conjoined, "and a truth is the intimation of the oneness of these two worlds". Thus, 'reality ' and any individual truth can only be defined relative to a specific social, cultural and even religious environment.

We consider that a model is a good model if its predictions are in agreement with observations which we make about an observable environment, within the limits imposed by the always present uncertainty principle. Just like everything else, engineering practice does not escape from this principle. In effect, in order to carry out observations we need to evaluate the quality of a model of whatever we conceive as 'reality' we need recording devices (sensors) which carry intrinsic inaccuracies. In the worst of cases, the sensors, which are also subject to malfunctions, could register and transmit distorted information to the real-time control systems. A robust real-time control system must be capable of simulating these sensor malfunctions in order to identify them - and so take the proper decision - in the real world. In MOUSE ONLINE the importance of this matter is acknowledge and today the Mouse Simulator package can deal with this kind of situation (Martin, 1993).

It is generally accepted today that the only possibility for effective real-time control systems is an architecture based on both forecasts and on-line measurements. This topic will be discussed in detail in the next section.

2.3 Practical derivation of optimal control strategies based on both on-line measurements and forecasts

A discussion will be opened in this section about the way in which on-line measurements and forecasts could be combined in practice in order to perform effective real-time control. The process which will be described, here although designed for the MOUSE ONLINE environment, can in principle be applied to any system with similar configuration. For the sake of a better understanding of the further explanations, let us first illustrate the generic characteristics of the automated on-line control system (adapted from Tomicic, 1989). These are schematized in figure 6.

In relation to this prototype for the automated real-time control system we must outline several important features:
Figure 6: A prototype for the automated real-time control system
i) In this combined logical-numerical framework the control action is obtained on the basis of both remotely transmitted on-line measurements and forecasts.

ii) At the beginning of each optimisation cycle the forecasts are corrected - updated - with online information from sensors.

iii) The optimal control strategies to be applied on the system are realised in successive stages of increasing level of accuracy.

iv) The process ends when the optimal control strategies covering the so-called forecasted horizon are obtained on the basis of reliable information. This means that no significant improvements are expected from further updates of forecasts.

v) In principle the fully non-linear optimisation module makes use of a dynamic wave approximation for producing pipeflow forecasts, although an intelligent switch between modules could be implemented in order to accommodate the simpler models which are available within the MOUSE system.

vi) The built-in expert system plays an important role in the optimisation process not only in relation to the link with the numerical optimiser but also due to the fault diagnosis module.

vii) In the MOUSE ONLINE environment communications are performed in a fast, safe and efficient way through the use of a so-called 'double blackboard' architecture.

viii) All the elements are equally essential to the success of the real-time control process.

From an operational point of view we could describe the process of correction of forecasts with on-line information within this environment in the following way: MOUSE ONLINE has three working modes, namely monitoring, forecasting and controlling. The monitoring process is usually 'on', meaning that on-line information from the standard SCADA (Supervisory Control and Data Acquisition) system is constantly recorded and sent to the real-time control system to be analyzed. In principle the change from the 'default' (monitoring) mode to the forecasting mode in the MOUSE ONLINE system used elsewhere in the SP226 project is produced:

- If it has rained for more than a certain period of time in any of the catchments - threshold value A.
- If the intensity of the rain has exceeded the threshold value B.
- If the inflow to the treatment plant has reached the threshold value C.

In the same fashion, the change from forecasting mode to controlling mode is produced:

- If the inflow to the treatment plant is about to exceed threshold value C1.
- If certain critical levels at the keypoints are exceeded (set of thresholds D[]).
On the other hand, the switch from controlling back to monitoring is operated when a significant and sustained decay in the active parameters is registered.

The control action taken relies heavily on the flow forecasts. If these forecasts are not accurate the control decision cannot normally be good either. Therefore, in order to perform effective control it is necessary to guarantee that the forecasts are accurately produced. This can only be assured through a cycle-by-cycle correction process.

Due to the complexity of the problem that they address, numerical optimisers generally require a relatively large number of cost function evaluations - each involving one hydrodynamic simulation - in order to construct the gradient vectors and the matrix of second order partial derivatives (also called the Hessian matrix). It is therefore very important to specify a default control strategy to apply on the system while the quasi-optimal control strategy is being obtained by the numerical optimiser. This default strategy which must be something better than just 'open regulators' is derived by what we should shortly identify as the intelligent agent in the process of providing the initial points for the fully non-linear, numerical optimiser. In effect, the process of deriving a quasi-optimal control strategy by making use of the intelligent agent takes practically no time and therefore ensures the effectiveness of the real-time control process. We will explain this in detail in the next section.

This process is illustrated in figure 7, where we have tried to define an imaginary 'physical plane' in which the different control strategies are scaled and mapped together with the rain hydrograph in terms of rain intensity against time. By using this representation it is possible to show the role of the system of update of forecasts with on-line information from sensors in a process which speeds up considerably as time elapses due to the reduction of the forecasted horizon. The following notation has been employed:

On the X axis (t).

SR : Start of the rain event.
SF : Start of forecasts (threshold values have been exceeded)
SO : Start of optimisation (first cycle).
1SO : First 'optimal' control strategy (OCS) is obtained (with time shift to 1SA).
1SA : First OCS is applied and enough on-line information is available.
2CY : Start of the second cycle of optimisation (based on corrected forecasts).
2SO : Second OCS is obtained (also with time shift)
2SA : Second OCS is effectively applied.

On the Y axis [Rain intensity (RI) / Regulator Setting (RS)].

RI : Rain Intensity.
RS : Regulator settings.
DCS : Default control strategy for the regulators (RS vs t).
ThV : Threshold value.
RRP : Actual (observed) rain pattern (RI vs t).
1FR : First forecast of rain (RI vs t).
2FR : Second forecast of rain (RI vs t).
1CS : First OCS (RS vs t).
2CS : Second OCS (RS vs t).
2.4 The problem of specification of default control strategies in systems using numerical optimisation techniques

The operation of the flow regulators in urban drainage systems can have a major impact in the flow pattern in the network. Indeed, this is the purpose of the control. In an effective real-time control system, the optimal control strategies provided to minimise undesirable effects in the system's operation have themselves to be derived and applied with a safe margin in relation to the ongoing real-time process.

There are two main kinds of control which can be applied to the solution of the problem of real-time control of hydrodynamic networks. They are:

i) Stationary control: This kind of control is performed if the control strategy is not time dependent, that is, the flow is controlled on the basis of the current situation in the system and a given initial control strategy. On-line devices are the main sources of information about the system's behaviour.

ii) Dynamic (predictive) control: This kind of control is based on the application of time series of control decisions specified in advance. Usually a much better utilization of the system's
storage potential is obtained, but this requires reliable forecasting tools in order to estimate
the system's behaviour over the so-called forecasted horizon, as well as some decision-making
module. Corrections to the control strategies can be achieved by refining the accuracy of the
forecasts based on up-to-date online information from sensors.

The present combined logical-numerical framework has been conceived for application in
dynamic control (B). As will be easily understood, the need of deal with time-dependent
vectors considerably increases the complexity (in relation to the size) of the problem and thus
the computational time required for its solution. In a sense, it is possible to state that *discrete-
continuous* non-linear multivariable optimisation applied to dynamic control (B) replicates
the stagewise representation of the process used in dynamic programming numerical
optimisation techniques. This research has proven that this hybrid framework is certainly
capable of providing an accurate and real-time-feasible solution to complex problems of on-
line control of the flow in networks of the kind used in urban drainage systems, unlike the
existing applications of dynamic programming techniques where the real-time constraint is
usually unsatisfied unless the problem's structure has been considerably simplified (Tien,
1995, p. 70).

Although many features have been implemented in this system in order to reduce the search
space for the numerical optimiser, still the time used for the derivation of optimal control
strategies requires the specification of proper 'default' control strategies for the system while
a better strategy is being derived.

This is a very important item to be taken into account in on-line systems using numerical
optimisation techniques. As suggested in section 2.3, these 'default' strategies must be
something better than just regulators open - or closed - for the sake of the overall effectiveness
of the control process.

In our case these 'defaults' are provided by the intelligent agent in its task of selecting initial
vectors for the numerical optimiser since the derivation of quasi-optimal strategies by using
knowledge about the system is a very fast process. Therefore, the result of this stage are
available *well in advance* in relation to the critical moments of the inflow event. Although,
the design of this process will be covered in section 5.3, it can already be anticipated that the
optimisation process with the present framework is achieved in a sequence of stages which
starts with a logical optimisation step.

* Also called predictive or dynamic non-linear programming. What this means in practice is that the vectors of the
system state variable and control strategies are projected over the event horizon in several discrete stages in time.
2.5 An overview of potentially applicable techniques

The purpose of this section is to provide a short discussion of techniques - of any kind - which in principle could be applied to the solution of the real-time control problem. Some of them are out of the scope of the present research and therefore they have not been explored. However, the fast development of optimisation methods in general, and particularly those associated with areas of Artificial Intelligence and Genetics makes it necessary to take them into account because they represent a certain promise. But again, with the present state-of-the-art, the most advantageous characteristics to real-time control seem to be found in knowledge-based systems and numerical optimisation techniques. At the moment of writing this report, research was being performed at the Danish Hydraulic Institute in order to explore the potential of Artificial Neural Networks and classifier systems, to the solution of this problem.

The problem of the derivation of optimal control strategy can be attacked by several means. Among these are:

1. Heuristic rules (KBS)
2. Diagnosis (KBS)
3. Verification (KBS)
4. Numerical optimisation (NO)
5. Genetic algorithms (Gas)
6. Classifier systems (GAs/SE)
7. Artificial Neural Networks (ANNs)

Here 'KBS' refers to 'Knowledge-Based Systems' and 'SE' stands for 'Software Engineering'. The first three are covered by what is called symbolic paradigm of Artificial Intelligence (AI) and the others belong, each in its own way, to the sub-symbolic paradigm of AI.

Each of the mentioned approaches has its advantages and disadvantages. In spite of the substantial research work performed in recent years, none of these approaches had succeeded so far in demonstrating a decisive advantage over the remaining ones in relation to the solution of this problem. Moreover, it seems that none of them taken alone contains a potential which is sufficient to solve this problem satisfactorily.

As it is a common feature of Semiotics (Eco, 1977; Abbott, 1993), that the combination of sign representations that lead to the greatest and lower increases in semantic knowledge, it can be assumed here that the right combination of methodologies will also provide the most efficient description.

Ad 1) A heuristic rule-based controller may be regarded as an special kind of knowledge-based (or expert) system which encodes heuristic knowledge (also called strategic or high-level knowledge) about the domain in multiply-connected rules. These rules derive control actions on the basis of the knowledge they encapsulate representing the best available experience about the operation of the system. The inference mechanism which is generally separated from the knowledge base itself performs reasoning on the basis of the knowledge contained in rules,
which are triggered in order to achieve a certain goal (in the case of the backward-chaining mechanism).

It has been proven that the *heuristic rules*, when implemented in a real-world application (city of Aalborg, former ROSA System, DHI), are capable of producing satisfactory results (see Amdisen, 1992). Once the consequences of the initial control strategy are known, it takes only a short time before a decision for the control strategy modification is obtained. This is extremely important in a real-time applications, where the time available for setting up a new strategy is usually very short.

However, the utilization of the available storage capacity actually achieved is far from being optimal. Additionally, such systems so far lack generality, and so far have been one-off systems, based on site-specific analysis, and they have therefore proven to be expensive to develop.

*Ad 2)* The *diagnosis* is based on the comparison of the *desired state* of the system with the *current state* and it describes the discrepancies. The necessary control actions are identified based on reasoning about the cause of the discrepancies. The reasoning is then based on general knowledge about causalities, i.e. relations between cause and effect, in the sewer network (Amdisen, 1993). Due to its relevance we will return to this technique in sections 2.7 and 4.3.

*Ad 3)* The *verification* is based on matching the current state of the system with a number of *known states* stored in a database. The known state which matches the best with the current state is identified. Then, the control actions corresponding to the known state are applied. However, practical difficulties arise due to the level of matching that real-world events actually achieve in relation to the so-called known events. Due to the strongly non-linear characteristic of the stochastic and basically unpredictable input (rain event), interpolation techniques *cannot* be applied, as and by themselves, to resolve this problem.

*Ad 4)* *Numerical optimisation* techniques determine the extrema (in this case minima) of a function. This objective function is a numerical description of the cost of the current state based on the objectives of the control process. The application of this method requires that the information and the knowledge about the system can be translated into numerical descriptions of cost of either meeting or missing certain criteria.

Several attempts have been made to develop a generalized optimization tool for the derivation of an optimal control strategy in conduit networks with free-surface, unsteady flows, based on generalized simulation models and non-linear optimization techniques (and specially steepest descent methods). None of the attempts have resulted in an operational tool, but many potential problems, as well as advantages have been revealed. The major issues in this respect can be summarized as follows:
i) Advantages
- absolute generality of the level of the code
- high accuracy in dealing with complex non-linearities, when correctly implemented.

ii) Disadvantages
- high computational load, making it very close to the limits of applicability for real-time applications with the current serial computational hardware platforms and with present (raw) computational solutions
- problems with suboptimal solutions (local optima). However, this last disadvantage might be attributed to all methods, when applied individually.

Ad 5)
Together with classifier systems, genetic (a class of evolutionary algorithms) algorithms (GAs) have probably the shortest history of all, in that so far as is known they had not been applied at all in this context.

However their successful application in other, related areas suggests they do represent a certain promise, so that they must be considered also.

These algorithms select a number of promising control actions, and apply them, analysing the results in order to generate new control actions based on combinations (crossover) of the most successful ones. These combinations are performed according to a certain predefined strategy. More precisely stated, evolutionary algorithms maintain a population that evolve according to certain rules of selection. Each individual in the population receives a measure of its performance in a given environment. This measure of performance is often referred to as its fitness. After a certain (generally large) number of iterations, the sampling of the search space and the exchange of information among individual structures (through the so-called crossover operator) produces convergence to a local optimum.

However, the best way to apply these methods to real-time control of urban drainage networks has not been explored, and much further - and careful - research will still be required in this field.

Ad 6)
Learning classifier systems - a technique connecting Artificial Intelligence and genetics - are essentially sub-symbolic rule-based systems that guide behaviour in environments where certain generalisation over states is possible. A learning classifier system learns appropriate behaviour from experience, is adaptive and supposedly escapes from the brittleness inherent to traditional, symbolic, rule-based systems.

When applied to optimisation, the objectives can be translated into cost functions. By updating the cost prediction based on experience, a mapping from states (S) and actions (A) to cost predictions (C), $S \times A \rightarrow C$, is learned.
The main limitations of this technique seem to be associated with the generality (range of validity) in including the stochastic input component (rain event) in the resulting rulebase. Additional restrictions arise due to the fact that site-specific (network dependent) features are incorporated into the final (if-then-action) set of rules, during the off-line learning process on a given network. Therefore, an attempt to transport the encapsulated (learned) rulebase to another network - other than the one upon which it was trained -, would fail.

Thus, each application of this product to a given network requires a new and rather costly learning process (since expensive resources such as man-time and computer resources must be allocated during several CPU-weeks, for an acceptable level of training). However, the response time is very short, once the classifier system is 'trained' in site. An application which have been developed at the Danish Hydraulic Institute (Wilson, 1995) (off-line, stationary control) has provided satisfactory results in cases of inflow events whose characteristics are similar to those for which the learning classifier system has been trained. However, due to its short lifetime further research will still be required in this field.

Ad 7)
The idea of Artificial Neural Networks (ANNs) was originally inspired by the sciences of neurology and pathology. These techniques replicate the behaviour of the brain by emulating the operations and connectivity of biological neurons. They are often regarded as 'black-box' models.

The main idea is that the ANN adjusts a series of connecting weights in order to fit a series of inputs to another series of known outputs.

When the training set for the ANN is large enough, the system is capable of producing an output for a given input - for which it has not been trained -, if this input is contained in the original range of validity. While the training period for the ANN in this field could last from a few hours to several days of CPU time, the response of the trained ANN is almost instantaneous.

An application developed at the Danish Hydraulic Institute (Khondker, 1995) has shown satisfactory results in optimising water resource systems, and its extension to other similar problems seems to be possible.

The restrictions of this technique seem to be associated with the range of application (range of validity) since a real world problem might appear for which the Artificial Neural Network 'has not been trained', in which case the response could be highly misleading.

The present research shows the potential contained in a hybrid system which takes advantage of the best characteristics of a knowledge base (combining elements of diagnosis, heuristics and fuzzy logic) and a discrete-continuous non-linear numerical optimisation methodology which operates according to a well defined strategy (sequence of steps). However, it is by no means excluded, and indeed, as stated earlier, it is to be expected that this potential can be further enhanced by combining the present solution with one or more of the others described above.
2.6 Comparative analysis of existing methods

The purpose of this section is to provide an approximate picture of the state-of-the-art [1995] in the field of real-time control of combined-sewer urban-drainage networks. The main source of information for this review is the paper to the sixth international conference on urban storm drainage in Niagara Falls by Gonwa et al (Sept, 1993).

Intense research has been performed in this field during the last few years. This overview appears to contain a rather complete picture of the present situation and provides essential elements to our further discussion.

Numerical control algorithms require flow modelling. Flow modelling methods range from the full Saint Venant's equations to storage routing with hydrograph translation (with which it is not possible to take into account backwater effects). On the other hand, inflow forecasting may be deterministically or statistically based. As explained above, inflow monitoring may be substituted (or complemented) by forecasting in certain systems. Forecasting over the event horizon considerably increases the size of the optimisation problem.

The existing methods found in the above mentioned review are summarized in table 2.1. When using an overview like this, it is rather difficult not to employ its author's criteria. This is the reason why the table has been modified in some cases and our own experience and points of view accumulated in the present study have been included for the sake of its completeness and topicality.

Once again, it is necessary to emphasize that the criteria summarized in this table apply to each approach as a 'stand-alone' technology, whereas it is a central precept of hydroinformatics that technologies have to be 'blended' or 'merged' in order to realise their individual potentials.
<table>
<thead>
<tr>
<th>Implementability</th>
<th>Robustness</th>
<th>Applicability</th>
<th>Adaptability</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Full St Venant equations</strong></td>
<td>Excessive time consumption for on-line use. Rather familiar technology</td>
<td>Occasionally unstable. Internal cycles required. Needs complete data</td>
<td>All dynamic effects are properly simulated specially mild slope pipes</td>
<td>Models are easily modified to account for new facilities and flow rates</td>
</tr>
<tr>
<td><strong>Zero Inertia equations</strong></td>
<td>Execution time up to 10x less than St. Venant’s Technology uncommon</td>
<td>Seldomly unstable, being inherently diffusive. Also requires complete data</td>
<td>Models most dynamic effects properly. It often has enough accuracy for mild sloping pipes</td>
<td>Models are easily modified to account for new facilities and flow rates</td>
</tr>
<tr>
<td><strong>Kinematic equation</strong></td>
<td>Fast. Moderate complexity. Technology uncommon</td>
<td>Usually robust. Can generate unrealistic hydraulic grade line if assumptions not fulfilled</td>
<td>Cannot handle backwater or backflow. Reduced accuracy in mild slope networks</td>
<td>Models are easily modified to account for new facilities and flow rates</td>
</tr>
<tr>
<td><strong>Storage Routing w/ Hydrograph translation.</strong></td>
<td>Fast. Little complexity. Technology is common. Simplifications require careful verification</td>
<td>Robust. Requires complete data</td>
<td>Cannot model dynamic effects. Least accurate modelling method. Simplifications required on all systems</td>
<td>May be difficult to modify depending on how outlet rating curves are computed</td>
</tr>
<tr>
<td><strong>Non-Optimal methods. Local automatic control</strong></td>
<td>Fast. Concept is simple. Common technology. (PID &amp; DDC controller)</td>
<td>Robust. Little data required. Careful design to avoid excessive cycling</td>
<td>Restrictive but no models required. Unaffected by dynamic effects</td>
<td>Control unaffected by flow and structural changes. Easily mod. control</td>
</tr>
<tr>
<td><strong>Remote supervisory control</strong></td>
<td>Fast. Concept is simple. Common technology (operating rules)</td>
<td>Robust. Unaffected by incomplete data</td>
<td>Restrictive but no models required. Unaffected by dynamic effects</td>
<td>Adaptable by changing operating rules</td>
</tr>
<tr>
<td><strong>Global Automatic Control with Heuristic algorithms</strong></td>
<td>Fast. Concept simple. Technology common. Complex rules are difficult to formulate</td>
<td>Robust, unless operating rules are incomplete. Good design can handle missing data and avoid excessive cycling</td>
<td>No models required. Unaffected by dynamic effects</td>
<td>If operating rules are based upon modelling studies, method shows poor adaptability. Otherwise control is adaptable</td>
</tr>
<tr>
<td><strong>Expert systems</strong></td>
<td>Fast. Concept simple. Rather new technology. Knowledge elicitation difficult. Often expensive</td>
<td>Robust unless coding incomplete. Good design can handle missing data and avoid excessive cycling</td>
<td>No models required. Unaffected by dynamic effects</td>
<td>Easily adaptable if expert system’s code is properly designed</td>
</tr>
<tr>
<td><strong>Fuzzy set theory</strong></td>
<td>Fast. Concept moderately difficult. New technology. 'Traceable' logic</td>
<td>Generally robust. Good design can handle missing data and avoid excessive cycling</td>
<td>No models required. Unaffected by dynamic effects</td>
<td>Adaptability restricted by understanding how control logic works before modifying</td>
</tr>
<tr>
<td><strong>Learning production theory</strong></td>
<td>Moderately fast. Concept rather difficult. New technology. Traceable logic Original set of rules need not be complete</td>
<td>Stability and performance improves with time. Can handle missing data</td>
<td>No models required. Unaffected by dynamic effects</td>
<td>Self-adapting</td>
</tr>
</tbody>
</table>
Table 2.1 (Continued): Evaluation of existing stand alone RTC techniques.

<table>
<thead>
<tr>
<th>Implementability</th>
<th>Robustness</th>
<th>Applicability</th>
<th>Adaptability</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural Networks</td>
<td>Once trained, fast.</td>
<td>Usually robust.</td>
<td>No models required.</td>
<td>Self-adapting</td>
</tr>
<tr>
<td></td>
<td>Concept rather difficult.</td>
<td>Unpredictable</td>
<td>Unaffected by</td>
<td>After training may</td>
</tr>
<tr>
<td></td>
<td>Logic untraceable.</td>
<td>performance if data is</td>
<td>dynamic effects</td>
<td>perform almost as</td>
</tr>
<tr>
<td></td>
<td>Black-box model</td>
<td>missing.</td>
<td></td>
<td>well as optimal methods</td>
</tr>
<tr>
<td>Optimal methods</td>
<td>Fast. Concept is</td>
<td>Stability guaranteed.</td>
<td>Linear model only</td>
<td>Difficult to modify</td>
</tr>
<tr>
<td>Linear regulators</td>
<td>difficult.</td>
<td>Requires complete data.</td>
<td>approximates system</td>
<td>for structural changes</td>
</tr>
<tr>
<td>theory</td>
<td>Technology for</td>
<td>Excessive cycling</td>
<td>performance. Systems</td>
<td>Can be self-adapting</td>
</tr>
<tr>
<td></td>
<td>large scale systems</td>
<td>(underdamping) or</td>
<td>must be greatly</td>
<td>for flow changes</td>
</tr>
<tr>
<td></td>
<td>is new. Design of large</td>
<td>slow response</td>
<td>simplified. Can</td>
<td></td>
</tr>
<tr>
<td></td>
<td>systems is difficult</td>
<td>(overdamping) may</td>
<td>account for backwater</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>occur</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear Programming</td>
<td>Rather slow.</td>
<td>Usually robust.</td>
<td>Linear models can</td>
<td>Difficult to modify</td>
</tr>
<tr>
<td></td>
<td>Concept moderately</td>
<td>Convergence to global</td>
<td>only approximate</td>
<td>Requires in-depth knowledge of Linear</td>
</tr>
<tr>
<td></td>
<td>difficult. Technol.</td>
<td>optimum guaranteed.</td>
<td>system performance.</td>
<td>Programming</td>
</tr>
<tr>
<td></td>
<td>uncommon. Logic</td>
<td>Requires complete data.</td>
<td>Systems must often</td>
<td></td>
</tr>
<tr>
<td></td>
<td>untraceable. Design can</td>
<td>Excessive cycling</td>
<td>be greatly simplified.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>be moderately difficult.</td>
<td>may occur</td>
<td>Cannot account for</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>backwater</td>
<td></td>
</tr>
<tr>
<td>Discrete-continuous</td>
<td>Slow. Concept is</td>
<td>Usually robust.</td>
<td>Non-linear model</td>
<td>Difficult to modify</td>
</tr>
<tr>
<td>non Linear and Dynamic</td>
<td>difficult. Technology is</td>
<td>Convergence to local</td>
<td>improves accuracy.</td>
<td>Requires in-depth knowledge of the</td>
</tr>
<tr>
<td>Programming</td>
<td>uncommon. Logic</td>
<td>optimum guaranteed.</td>
<td>Systems require</td>
<td>particular subject</td>
</tr>
<tr>
<td></td>
<td>untraceable. Design is</td>
<td>If precautions taken it</td>
<td>simplifications. Can</td>
<td></td>
</tr>
<tr>
<td></td>
<td>difficult</td>
<td>may be the global</td>
<td>account for backwater</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>optimum. Requires</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>complete data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Excessive cycling</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>may occur</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

So far, very few of the systems reviewed make use of numerical optimisation techniques (optimal methods) to the solution of the real-time control problem. The reason for this can be found in the non-realistic process representation (excessive simplification) offered by the linear programming methods and the heavy computational load associated with discrete-continuous non-linear or dynamic programming according to existing implementations. However these two methods (dynamic and discrete-continuous non-linear programming) are acknowledge as the more accurate of all existing optimisation methods, given sufficient computer capacity related to time.

Therefore, one of the purposes of this research was to prove that with a careful design - in which generic features of the code are very important -, it would be possible to implement a system which indeed makes optimal use of the best characteristics of this complex technology.

The results are encouraging. As it will be explained later, the interaction with the intelligent agent proves to be essential from all the points of view. A considerable potential hidden in this combination has therefore been revealed.
2.7 Hybrid solutions

The Theory of Semiotics proposes (as does hydroninformatics in the aquatic environment) that combining in an appropriate way the features of different approaches represents the most effective way to solve problems, including those of on-line control.

In our problem, this approach seems to be more rational - both conceptually and practically - than does the variant of developing a real-time control system for urban drainage networks which totally relies on a single technique for its solution. The main reasons for this are related to the possibility of attacking the problem in a more efficient way, by approaching the solution from different directions, in order to achieve a solution in a sequence of stages of increasing accuracy (which has proven to be an quite flexible scheme for real-time control). In this way it is possible to ensure a broader coverage of real life situations as the spectra of methods applied in the solution is also wider. In other words, the characteristics of the problem to be solved (ill-structure, strong non-linearities, stochastic input vectors) already suggest a hybrid control tool (Martin, 1993, p. 31).

In practice, the combination of techniques to be used is generally suggested by the desired characteristics of the final product. In this case the required features of the final prototype were:

i) Generality (total applicability)
ii) Speed
iii) Accuracy
iv) Transparency of the control process (logic traceability and explanation capacity)
v) Robustness
vi) Adaptability
vii) Maintainability

It is seen that features i), iii), v) are fully supplied by numerical optimisers, which also contribute to a certain extent to reinforce features vi) and vii). But ii) and iv) are certainly not to be found in those techniques. Rather, they are present in knowledge based related techniques (Fuchs et al, 1987, p. 271).

According to this discussion, a promising possibility seems to arise from the combination of fully non-linear numerical optimisation techniques with knowledge-based, Artificial Intelligence related techniques. What it appear clear now is that a non-simplified, numerical (or equivalent) optimisation technique must be present in whatever variant we choose, in order to ensure coverage of most real-world situations and also for the sake of fine tuning of the optimal strategies. Of course, more variants could still be introduced, according to this reasoning.

For example, another potentially interesting choice could in principle be expected from a combination of learning classifier systems with a discrete-continuous, non-linear optimisation methodology.
Another remark is related to the features of the techniques used for combining methodologies, namely that the desired features must be *complementary*. This means that the weak sides (in combination) of one technique must be covered (complemented) by the strong sides of the other one.

For example, a *non-interesting* combination would arise from combining genetic algorithms with non-linear or dynamic programming methodologies simply because both of these techniques have *similar* advantages and disadvantages for the solution of this problem.

As explained above a hybrid framework which takes advantage and combines advantageous features of a fully non-linear, numerical optimisation methodology and a logical, knowledge-based approach (using elements of diagnosis and heuristics) has been developed in the present PhD study. This framework has been successfully applied to the on-line control of real world urban drainage networks.

Several generic features have been incorporated in this framework to restrict the so-called *curse of dimensionality* which characterises the existing implementations of dynamic programming and fully non-linear optimisation algorithms (Petersen, 1987, p. 28). It will be subsequently argued that the most effective solution to this problem is that of using high level heuristic knowledge about the domain in order to reduce the search space. This will be explained in detail in section 5.3.

An important item to be taken into account for combined frameworks is the sequence of steps or the working strategy to be employed. In a sense that has to be explicated later, the real-time control process using this approach shows itself to be *asymmetrical* (or non-commutative). Although similar, different strategy vectors will be derived depending on the order of arrangement of the optimisation steps. In our case, the more convenient arrangement is obtained by a logical optimisation step followed by a numerical step.

### 2.8 The role of forecasts in the optimisation process

In this section an analysis will be made of the role of flow forecasts in this optimisation process. The contents of this discussion are closely related to the notions introduced in sections 2.4 to 2.7.

*: Curse of dimensionality is defined in discrete-continuous non-linear and dynamic programming as the phenomenon by which the number of objective function evaluations, the size of computational storage and other computer resources increase exponentially in relation to the number of control variables (ncont) and the number of flow stages to be optimised (projection of the system state vector over the event horizon (npstr)). In a fully non-linear, multivariable optimisation methodology the order of the basic gradient matrix is \( \{ncont \times npstr\} \). This problem is so serious that in the early stages of development of the Dynamic Programming, some authors expressed their doubts about if "there would ever be possible to solve problems with more than four or five resources" (from Tomicic, 1989). The several alternative ways in which the curse of dimensionality has so far being restricted will be explained in detail in section 5.1.
It has been estimated that nearly ninety-five percent of the time employed by a numerical (non-linear) optimisation methodology is devoted to the fully dynamic pipeflow simulations. Although this figure varies from platform to platform, it is still possible to state that in general most of the computer time is taken up in this process.

Although hard to implement, we could imagine that from the point of view of time consumption an 'optimal' hybrid system using a numerical methodology should be able to switch between several standard technologies (table 2.1) employed to produce flow forecasts depending on the characteristics of the particular problem to be solved.

That is the reason why we decided to offer a review of the theory underlying the most used forecast technologies employed for the solution of the dynamic non-linear optimisation problem.

2.8.1 Rainfall forecasts. The distributed rain feature

Among the many techniques currently employed to forecast rainfall events, radar instrumentation seems to be the most common and accurate (Reed, 1987). However, other techniques based on data extrapolation from long statistic records of rainfall in the area have also been successfully applied in some cases.

Rainfall forecasting is mainly a concern of meteorologists. In the MOUSE ONLINE environment, forecasted overall rainfall at fifteen minutes sampling interval is supplied on the basis of the collected on-line information about the current event and past events in the area as well as from radar instrumentation from a local meteorological agency. The forecasted rainfall data is then used as input to the surface runoff model.

When dealing with large scale urban drainage systems another important factor to be taken into account is the spatial distribution of the rain over the catchment. Many of the existing modelling packages still consider the rainfall to be uniformly distributed over the area, an assumption which may lead to very misleading results in some real world situations (and then, of course, mostly in large scale systems).

The MOUSE ONLINE system includes the distributed rain feature as a way of "providing a more effective (realistic) modelling tool as well as to reveal important interactions existing in the system .." (Martin, 1993, p. 48). The practical idea behind this feature is to assign a [potentially] different rain event to each and every sensor in the network, including the possibility of assigning no rain in a specific area. This is achieved through a dynamically updated association record between sensors and events in the on-line database.

The realisation of this feature in the MOUSE ONLINE system is illustrated in figure 8 (depicting the user interface developed).
Figure 8: The distributed rain feature of MOUSE ONLINE (after Martin, 1993)

2.8.2 Surface runoff forecasts. The infiltration component

The surface runoff component (watershed) enters the network through multiple points (manholes, catchment basins, gully pots). Therefore, the flow circulating through each of the different branches is the sum of the inflows coming from upstream routing and the inflows entering these points in each branch.

The objective of the rainfall-runoff model is to determine the exact values of the time series of inflows which are entering the network through different points. Therefore the results of a runoff simulation consists of runoff hydrographs (Q against t) at each catchment, which are used as inputs to the pipe flow model.
In general, three types of models can be distinguished:

a) **Distributed physics-based models**, which use the laws of conservation of mass, energy and momentum as well as parametric representations of other processes to describe the motion of the watershed over the surface and occasionally through the unsaturated and saturated zone. The resulting system of partial differential and algebraic equations is then numerically solved at all points in the two or three dimensional grid. Very few of these models have been developed and the computational requirements are currently much too high to employ in optimisation processes for urban drainage systems.

b) **Lumped conceptual** (quasi-physical) models, which use a simplified representation of the surface runoff process, frequently involving several linked stores and simple budgeting procedures which ensure that the mass balance between inputs, outputs and inner storage is always kept.

c) **Input-Output black-box models**, which make use of empirical relationships between the system's input (rainfall) and streamflow output without trying to understand the internal mechanism of transformation. The unit hydrograph* method is an example of linear black-box model.

In the MOUSE system the user can choose between four runoff models, that are referred to 'levels' A, B, NAM and 'French runoff'. The relatively simple level A, describes the surface runoff by means of a single loss and a time/area function. The more complex level B, includes infiltration as expressed by Horton's formula in addition to the initial losses such as wetting, evaporation, storage depression, etc. A brief description of some of these models follows:

**Level A:**
The hydrological process in the surface is described by using the following data (user supplied):

* Hydrological reduction factors indicating the amount of rain in the area which is directly causing runoff (typically between 0.80 and 0.95)

* Initial loss describing wetting and storage (corresponding to the filling of depression and holes in the terrain). This is typically between 0.05 and 1.00 mm for impervious area.

* Concentration time expressing the time elapsed before the runoff from the entire catchment reaches the junction.

* Time area curve to be chosen in relation to the shape of the catchment (for example divergent, rectangular or convergent).

---

*: The **unit hydrograph** is defined as the watershed hydrograph resulting from one unit (usually one centimetre) of effective rainfall of uniform rate which is uniformly distributed over the catchment during a unit period of time.
In level A the rain is supposed to be uniformly distributed over the catchment. In every time step, the accumulated rain depth is calculated and the runoff is considered to start as soon as this depth exceeds the initial loss for the catchment. Only the impervious part of the area causes runoff. This runoff is then multiplied by the catchment's hydrological reduction factor. The resulting runoff hydrograph is then calculated on the basis of the concentration time and the chosen time/area curve.

Level B:
At this level the model computations are based on the continuity and the kinematic wave equations. The hydrologic and hydraulic processes involved are described in figure 9.

The hydrologic process is characterised by a loss process and by considering the continuity equation for a specific area. The effective rainfall \( I_{e} \) (m/s) per unit area can be written as:

\[
I_{e}(t) = R(t) - Q_{E}(t) - Q_{w}(t) - Q_{I}(t) - Q_{S}(t)
\]  

(2.1)

where:

- \( R(t) \) = Rain hydrograph
- \( Q_{E}(t) \) = Evaporation
- \( Q_{w}(t) \) = Wetting (When the surface is wet \( Q_{w} = 0.0 \))
- \( Q_{I}(t) \) = Infiltration
- \( Q_{S}(t) \) = Storage

With all values expressed in the corresponding metric units. The evaporation loss is normally insignificant during the precipitation period. The wetting is a discontinuous loss (in the model it is assumed that the precipitation minus the evaporation is used for the wetting of the surface). The rain is considered to be a random time function, but uniformly distributed over the catchment.

The infiltration loss \( Q_{I} \) is calculated using the well-known Horton's equation:

\[
Q_{I} = Q_{0} + (Q_{I} - Q_{0}) \times e^{(-k \times t)}
\]  

(2.2)

where:

- \( Q_{I} \) = Infiltration at \( t = 0 \)
- \( Q_{0} \) = Infiltration at \( t = \infty \)
- \( k \) = Characteristic soil parameter

For each timestep a resulting depth on the surface is calculated in relation to the rain intensity and the magnitude of the individual loss terms. The actual runoff starts when the resulting depth on the surface as calculated from the hydrologic processes is larger than zero. The hydraulic process is described with the kinematic wave equations for the entire surface simultaneously. This description assumes steady flow condition and uniform water depth.
Figure 9: The dynamic processes in the surface runoff model (level B)
over the surface. It must be remembered that a simple kinematic model does not take into account the storage induced delay over the surface. This model is often called a non-linear reservoir model.

Finally the runoff is determined as:

\[ Q(t) = M \times B \times \sqrt{I} \times \frac{3}{\sqrt{y^5}} \]  
(2.3)

where:

- \( Q(t) \) = Runoff as a function of time
- \( M \) = Manning number of the catchment
- \( B \) = Runoff width
- \( I \) = Catchment surface slope
- \( y \) = Depth computed from the continuity equation

With all values expressed in the corresponding metric units. The depth \( y \) is computed according to:

\[ Q(t) = \frac{dy}{dt} \times A \]  
(2.4)

where:

- \( Q(t) \) = Runoff as a function of time
- \( A \) = Surface area
- \( dt \) = timestep
- \( dy \) = change in depth during the time interval

The solution of the equations is performed by arranging equations (2.1) to (2.4) in such a way that water depth and flow rate can be determined implicitly on the next time step. The recommended range for the timestep is between 30 s to 120s.

2.8.3 Pipe and channel flow forecasts. The use of the kinematic, diffusive and dynamic wave approximations

The \textit{Saint Venant} equations are perhaps the most used representation of one-dimensional free-surface flows which are found in standard numerical software packages today. They provide the \textit{mass-momentum} couple of conservation laws applicable to both discontinuous and continuous situations in a structure that is advantageous for computer solving. These equations of motion determine the flow condition (variations in depth and flow rate) in a pipe or channel when they are solve with proper initial and boundary conditions. In a general form these equations are written as:
Continuity equation:

\[ \frac{\partial Q}{\partial x} + \frac{\partial A}{\partial t} = 0 \]  \hspace{1cm} (2.5)

Momentum equation:

\[ \frac{\partial Q}{\partial t} + \frac{\partial}{\partial x} \left[ \beta \frac{Q^2}{A} \right] + gA \frac{\partial v}{\partial x} = gA (I_o - I_f) \]  \hspace{1cm} (2.6)

where:

- Q : flowrate (m³/s)
- A : cross sectional area (m²)
- x : longitudinal axis (in relation to the flow)
- t : time (sec)
- g : acceleration of gravity (m/s²)
- \( \beta \) : velocity distribution coefficient (over the cross section)
- \( I_o \) : bottom slope
- \( I_f \) : friction slope (slope of energy grade line = \( Q^2/K^2 \) where K: conveyance)

These equations are derived under the following assumptions:

- nearly one-dimensional flow.
- hydrostatic pressure distribution (smoothly curved streamlines)
- the effects of friction and turbulence can be taken into account by the same equations used for steady state flow (usually employing the manning formulation).
- small average bottom slope (\( \cos \psi = 1 \))

The above equations are valid for free surface flow only. In the case of pressurized flow (full pipes) they are extended by making use of the so-called Preissman slot of width (MOUSE 3.10 documentation, p. 5-7) equal to:

\[ B = g \times \frac{A_o}{a^2} \]  \hspace{1cm} (2.7)

where:

- B : Slot width (m)
- \( A_o \) : Area without excess pressure (m²)
- a : speed of sound in water (m/s). This is taken as 1000 m/s for most pipes.
In general, it is necessary to guarantee a smooth transition between free and pressurized flow computations. For this reason the theoretical slot width has to be multiplied by a coefficient greater than one. There are tables showing the values of the slot width in relation to the relative depth of flow (y/d). In general, if y/d > 1.5 then B is usually taken as 0.010 m.

In practice, further simplifications of the equations (2.5) and (2.6) are often introduced. The most common is that of assuming a wide prismatic (often simplified to rectangular) channel. In this case the equations can be rewritten as:

Continuity equation:

$$\frac{\partial Q}{\partial x} + b_s \frac{\partial h}{\partial t} = 0$$

(2.8)

Momentum equation:

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x} \left( \beta \frac{Q^2}{A} \right) + gA \left( \frac{\partial h}{\partial x} - T_o \right) + \frac{g Q |Q|}{C^2 A R} = 0$$

(2.9)

where:

- $b_s$: storage width (width of cross section measured at free surface in m)
- $h$: water depth (m)
- $R$: hydraulic radius (m)
- $C$: Chezy resistance coefficient

Each of the terms in the momentum equation (2.9) can be regarded as a slope. The first two terms are the so-called inertia or acceleration terms (Cunge et al, p. 45), representing the slope of the energy line due to variations in acceleration and velocity head. The third term is the so-called pressure term which takes into account the component of momentum favouring the flow coming from the curvature of the streamlines (free surface gradient).

In this equation (2.9), it is also possible to identify the gravity term (momentum from bottom slope) and the friction term (fourth term in the equation), which represents the slope of the energy line due to friction losses.

The following approaches to the solution of the Saint Venant equations which come from including (or neglecting) the corresponding terms in the momentum equation are available as options in the pipeflow model of MOUSE:

- Kinematic wave approximation [gravity + friction terms]. This does not allow the representation of phenomena such as backwater effects (typical of subcritical flow) or surcharge.
This is an approach used for modelling of the flow in steep, partially-full pipelines where the flow is basically controlled by a balance between gravity and friction forces. This is the simplest rational representation.

- **Diffusive wave approximation** [gravity + friction + pressure terms]. This is a theoretically better approximation than the kinematic in the sense that the inclusion of the pressure term in the momentum equation allows the introduction of the downstream boundary condition and thus 'consider' (albeit in an incomplete way) the downstream (or backwater) effects. In practice, this model can produce a reasonably accurate computation of the surcharge phenomena.

- **Dynamic wave approximation** [gravity + friction + pressure + inertia terms]. With the inertia terms included in the momentum equation, a hydraulically correct representation of most of the complex phenomena arising in subcritical flows is possible. The main practical difference as compared with the diffusive wave approximation is that the dynamic equations are better at computing sudden changes in the flow, such as - for example - the effect of a rapidly rising gate (or weir), which is often very important in numerical optimisation.

In general, it is possible to state that in order to obtain an accurate hydraulic representation of the flow conditions which normally exist in urban drainage network a dynamic wave approximation is required, although some other methods have been reported (Gonwa, *et al*, 1993) with satisfactory performance and therefore they should be taken into account as a supplementary possibility.

Even the dynamic wave approximation cannot take account of motions involving significant vertical accelerations, such as the roll waves and surges, and for this purpose further extensions such as the Boussinesq equations, (see Abbott, 1979) have to be introduced. Although this is widely done in other areas of application (wave disturbance tests in harbours) it is rarely if ever applied in the area of interest here.

Probably the most frequently found and important special flow condition is the channel junction. The model is then (Cunge *et al*, p. 49) "considered as a set of reaches in which the de Saint Venant hypotheses are valid, linked by special points where different laws are introduced ...". The channel junction requires three (for three reaches) independent conditions the first of which is:

\[ Q_3 = Q_1 + Q_2 \]  
(2.10)

In practice, the other two conditions often come from the equality of energy levels in each of the reaches (in real problems, all relevant losses should of course be taken into account):

\[ h_1 + \frac{u_1^2}{2g} = h_2 + \frac{u_2^2}{2g} = h_3 + \frac{u_3^2}{2g} + \zeta \]  
(2.11)
where:

\[ Q_1, Q_2, Q_3 \]: discharges in each of the reaches.
\[ h_1, h_2, h_3 \]: water levels in each of the reaches.
\[ u_1, u_2, u_3 \]: main velocities in the reaches.
\[ \zeta \]: total energy losses through the singularity.

This conditions are called, by analogy, the **interior boundary conditions**. An example of two interior boundary conditions is given in figure 10. Another possible - although obviously more physically restrictive - condition is the equality of water levels in each of the reaches.

In the case of urban drainage networks a channel junction is observed in the case of a pipe discharging into a manhole where the relevant parameter is the instantaneous water level in the manhole which again is a function of the inflow and outflows to and from the manhole. In order to model a complex branched pipe system, it is required to specify a set of equations for each individual manhole. A critical depth formulation is often used in order to describe free inlet to a manhole. The so-called Carnot formulas (used in the MOUSE package) seem to provide a fairly accurate description of the problem of the energy losses which arise in special flow conditions, although it is recognized that this matter will require further research.

In the most general case of the flow through a manhole, the losses are divided into:

i) Losses due to change in the flow direction for manholes consisting of two inlet pipes and one outlet pipe.

ii) Losses due to change in elevation where the level of the bottom of the manhole is lower than the level in the pipe.

iii) Losses associated to a flow contraction (usually important).

Finally the total energy loss is obtained by adding up the corresponding terms at the outlet pipe. For more details see the MOUSE documentation, pages 5-14 to 5-21.
The other factor to be taken into account for the real-time control process using numerical optimisation techniques is related to the numerical method used to discretise the equations. In the MOUSE model (MOUSE technical documentation, p. 5-25).

In this package the transformation of equations 2.5 and 2.6 to a set of implicit finite differences equations is performed on a staggered grid consisting of alternating $q$ and $h$ points, which is automatically generated by the model. This situation is outlined in figure 11, where the grid has been generated on a simple pipeline connecting three manholes.

![Figure 11: Computational grid in a pipe branch](c:\phd\grid.wpg)

The numerical scheme used by the MOUSE package is a six-point Abbott-Ionescu scheme as shown in figure 12.

Generally speaking, in order to obtain a stable solution to the finite differences scheme, the so-called kinematic Courant condition* should be satisfied. However, this should only be understood as a general observation since the violation of this condition do not means that a stable solution will not be obtained with MOUSE.

![Figure 12: Centered six point Abbott-Ionescu scheme](c:\phdscheme.wpg)

* Numerical means exist, of course, to relax this condition, but this particular matter is out of the scope of the present analysis.
The above is to say that, in principle, the following condition should be satisfied:

\[ \Delta x \geq v \times \Delta t \]  
\hspace{1cm} (2.12)

where:

- \( v \): main velocity (m/s)
- \( \Delta t \): time step (s)
- \( \Delta x \): distance between nodes (m)

Since the continuity equation is centred in \( h \)-points at the time level \( n+1/2 \), the individual terms in the continuity equation (2.8) can be approximated according to the following formulas:

\[ \frac{dQ}{dx} \approx \frac{1}{2 \Delta x_j} \left[ \frac{1}{2} (Q_{j+1}^n - Q_{j-1}^n) + \frac{1}{2} (Q_{j+1}^{n+1} - Q_{j-1}^{n+1}) \right] \]  
\hspace{1cm} (2.13)

and:

\[ \frac{dh}{dt} \approx \frac{1}{\Delta t} [h_{j+1}^{n+1} - h_j^n] \]  
\hspace{1cm} (2.14)

where:

- \( \Delta t \): timestep (s)
- \( \Delta x \): distance between computational nodes (m)
- \( h \): water levels (m)
- \( Q \): discharges (m³/s)

On the other hand, the momentum equation is centred in \( Q \)-points at the time level \( n+1/2 \). Then, the derivatives in equation (2.9) are approximated as follows:

\[ \frac{dQ}{dt} \approx \frac{1}{\Delta t} [Q_{j+1}^{n+1} - Q_j^n] \]  
\hspace{1cm} (2.15)

\[ \frac{d}{dx} \left[ \beta \left( \frac{Q^2}{A} \right) \right] \approx \frac{1}{2 \Delta x_j} \left[ \left( \beta \left( \frac{Q^2}{A} \right) \right)^{n+1/2}_{j+1} - \left( \beta \left( \frac{Q^2}{A} \right) \right)^{n+1/2}_{j-1} \right] \]  
\hspace{1cm} (2.16)

The remaining parameters in equation 2.9 are expressed on time level \( n+1/2 \). Now, substituting equations (2.13) to (2.17) in equations (2.8) and (2.9) - continuity and momentum - the
\[
\frac{\partial h}{\partial x} = \frac{1}{2} \frac{1}{\Delta x_j} \left[ \frac{1}{2} \left( h_{j-1}^{n} - h_{j-1}^{n-1} \right) + \frac{1}{2} \left( h_{j+1}^{n+1} - h_{j-1}^{n+1} \right) \right]
\]  
(2.17)

following equations are obtained (MOUSE reference, DHI):

\[
\alpha_j Q_j^{n+1} + \beta_j h_j^{n+1} + \gamma_j Q_j^{n+1} = \delta_j
\]

(2.18)

\[
\alpha_j h_j^{n+1} + \beta_j Q_j^{n+1} + \gamma_j h_j^{n+1} = \delta_j
\]

(2.19)

where (for eq. 2.19):

\[
\alpha_j = f(A)
\]

\[
\beta_j = f(Q_j^n, \Delta t, \Delta x, C, A, R)
\]

\[
\gamma_j = f(A)
\]

\[
\delta_j = f(A, \Delta t, \Delta x, \alpha_j, u, C, h_{j-1}^n, Q_{j-1}^{n+1}, Q_j^n, h_{j+1}^n, Q_{j+1}^{n+1})
\]

The four coefficients \(\alpha, \beta, \gamma\) and \(\delta\) are incorporated into the pipe matrix to express the connection between the energy level in manhole 1 (eq. 2.11) and the situation in the first two computational points \((h, Q)\) analogue to the transformation of the momentum and continuity equation. This situation is shown in figure 11. Therefore the structure of the pipe matrix before local elimination would be:

\[
\begin{bmatrix}
\alpha_0 & \beta_0 & \gamma_0 & \delta_0 \\
\alpha_1 & \beta_1 & \gamma_1 & \delta_1 \\
\alpha_2 & \beta_2 & \gamma_2 & \delta_2 \\
\vdots & & & \\
\alpha_n & \beta_n & \gamma_n & \delta_n
\end{bmatrix}
\]  
(2.20)

In which the coefficients \(\alpha_0, \beta_0, \gamma_0, \delta_0\) are provided by the manhole energy equation; coefficients \(\alpha_1, \beta_1, \gamma_1, \delta_1\) are provided by the momentum equation and the coefficients \(\alpha_2, \beta_2, \gamma_2, \delta_2\) are provided by the continuity equation.

The pipe matrix is then transformed by applying a local elimination procedure, and the resulting transformed matrix \(A^*\) is then solved (together with the set of initial and boundary conditions) by a double sweep algorithm. The forward sweep is carried out by applying recurrence relations to compute coefficients \(A, B, C, and D, E, F\) obtained from the above mentioned transformation to the equations of continuity and momentum, while the backward (return) sweep usually initiated at the downstream boundary, computes \(h\) and \(Q\) values by using the corresponding equations (Abbott & Basco, 1990). In this formulation it is possible to express the system state variable \((h, Q)\) at a given node \((j)\) as a function of the corresponding variable at the neighbouring points.
As suggested above, a *dynamic* wave approximation is generally required in order to achieve a correct hydraulic representation of the complex phenomena which might arise in subcritical flow in urban drainage networks. However, the diffusive and even the kinematic wave approximation have been applied to the solution of the real-time control problem as reported, for example, by Gonwa et al (1993). This approach could be a realistic representational tool in the cases of steep, partially full pipelines where the flow is controlled by a balance between gravitational and frictional forces. In these cases, no significant increase in accuracy should be expected from using a dynamic wave approximation which usually requires a *rather longer execution time* in relation to the other two approximations. The above situation is further complicated in the presence of looped networks and by the occurrence of supercritical flows. The treatment in these cases is described by Kutija (1994).

2.9 The economic model. Multi-objective optimisation

The real-time control of urban drainage networks aims at performing a *quasi-optimal* use of the storage capacity in the network during and shortly after the storm event in order to minimise - if possible avoid - all undesirable effects which might arise from the system’s operation.

As discussed in section 1.4, the operational objectives to be achieved with the real-time control of flow regulators are:

1) to minimise environmental damage due to combined sewer overflows
2) to reduce surface flooding
3) to equalise inflows to the treatment plant
4) to reduce operational and maintenance costs

These objectives are often *conflicting*. This means that a situation may arise in which when trying to meet one of these criteria, another objective is violated. An example of the above is the observed situation in which by closing the regulators towards downstream areas in the system we reduce the frequency and volume of combined sewer overflows but we increase surface flooding in upstream areas of the system.

The importance of an acceptably accurate quantification of the rewarding criteria associated with a given control strategy is now widely recognized. This quantification is realised by the introduction of the so-called *surrogate cost functions* (also called *synthetic objective or reward functions*) which describe the cost of either meeting or missing certain criteria at keypoints in the network. The *mapping* of each and all of these 'cost functions' on a unique hypersurface - on which the optimisation process is performed - is the main operator in the construction of the so-called *economic model* of the process.

In non-linear and dynamic programming the economic model of the process is obtained from *non-restricted functionals* which in principle can have *any shape*. The only restrictive condition in our problem is that these functions should be continuous (preferably smooth) in order to allow an acceptably accurate approximation of the derivative. From the point of view
of the programmer, the non-linearity of the cost functions means that a cost functional cannot be predefined. In effect, only the system analyst is capable of defining (for each given real-world situation) which are the cost functions that provide the best economic representation of the area of interest as a part of the universe of discourse.

The economic model contains a mathematical representation of the objectives imposed by the system's analyst. The evaluation of the objective functions according to a suitable criteria provides an exact measure of the fitness of that solution in terms of appropriate cost units (CU). Then, by comparing several feasible solutions a 'better' solution can be chosen according to the defined criteria.

It is clear that the result of the optimisation process can be only as good as the objective function itself is a good representation (model) of the actual objectives to be achieved with the system's operation. This is the reason why the formulation of the economic model is a very important step in the optimisation process.

One of the desirable features sought in optimisation is the possibility of simultaneously representing multiple objectives associated to different areas in the system according to a predefined scale of priorities. These objectives must then be integrated in a proper fashion in order to obtain a global surrogate cost -the composition of all these different objectives-associated to a control strategy. This is referred to as multi-objective optimisation.

The main reason why dynamic control (section 2.4) is superior to stationary control is because the control decision selected as the 'best strategy' is not only based on the present situation in the system but also on the future (forecasted) situation. In order to implement this approach, good forecasting tools are required, as described in section 2.8.

As a matter of fact, the availability of reliable forecasts about the system state variable is only a part of the dynamic control problem. The other part is related to the effectiveness of the method employed for cost evaluation. In effect, the cost evaluation procedure must also ensure that the evaluation of a control strategy as 'good' or 'bad' takes into account the effect of the strategy on future stages of the system.

The integration of rewarding or surrogated cost units over the event horizon seems to be the method which allows the most effective economic representation of the real-time control process. The main idea behind is that a set of 'cost time series' is constructed from the cost functions at keypoints in the system which are 'filtered' through time series of the system state variables at these keypoints. These cost time series which are composed by instantaneous cost values associated to the keypoint - given a control strategy - are then integrated over the event horizon and the result is regarded as a global cost of the control strategy at the keypoint. Then, the sum of the individual (integrated) costs at each of the keypoints provides the total cost associated to the control strategy applied on the system.

As indicated before this cost takes into account not only the effect of the strategy on the present state of the system but also on future stages. In other words, by using this system for cost evaluation, it is possible to avoid the problem of considering a control strategy to be
'good' because it improves the present state of the flow while it might be 'bad' or even 'disastrous' for the rest of the storm event. This process is repeated whenever a new strategy is being tested during the optimisation process.

The process of integration of costs over the event horizon is illustrated in figure 13. The time series of the system state variable at each of the keypoints are obtained in this framework through fully dynamic numerical simulations (with all terms included in the Saint Venant equations).

The present implementation of the combined logical-numerical framework for real-time control of urban drainage networks employs a fully non-linear numerical optimisation methodology designed to be employed in dynamic control using reliable forecasting tools.
A) - Provided control strategies at N regulators
B) - User supplied cost functions at M keypoints
C) - Time series of the SSV at M keypoints (HD simulations).
D) - Cost time series obtained at M keypoints. IC -> integrated costs.

Figure 13: Integration of cost over the event horizon which takes place within the program 'CFCALC'(the cost function calculator/evaluator/integrator)
Chapter 3  Numerical Optimisation

3.1 Introduction

The subject of numerical optimisation is a fascinating blend of heuristics and rigour, of theory and experiment. It can be studied as a branch of pure mathematics, yet has applications to almost every branch of science and technology.

According to Fletcher (1987), optimisation might be defined as the science of determining the 'best' solutions to certain mathematically defined problems which are often models of physical reality. It involves the study of optimality criteria for problems, the determination of algorithmic methods of solution, the study of the structure of such methods, and computer experimentation both under trial conditions and on real-life problems.

Two main areas can be distinguished within numerical optimisation. They are mathematical programming and variational methods. The main difference between the two is that in mathematical programming the objective is to locate the best point (vector): \( x(t, x_2, ..., x_n) \) that optimises the economic model of the process. On the other hand, in variational methods the objective is to locate the best function that optimises the economic model. Generally speaking, mathematical programming methods are more readily applicable to steady-state problems while variational methods are designed to deal with dynamic problems. However, the complexity of the variational methods when applied to real-life problems on one side, and the possibility to reformulate dynamic problems into equivalent or approximate stationary problems on the other, causes a much wider application of mathematical programming techniques with their intrinsic advantages for serial computation. The main areas in numerical optimisation are summarized in table 3.1.

This piece of the study (numerical optimisation) is basically about discrete-continuous nonlinear multivariable optimisation supported by forecasting models. The forecasting capability allows the projection of the vectors of the system state variable as well as the control strategy vectors over the event horizon. Therefore, rather than solving \( n \) relevant - instantaneous - stages during the storm event, with an equal number of optimisation cycles, a 'policy' vector is found which carries \( n \) significant discrete stages (covering the event horizon) for each of the flow regulators in the system. This is only possible with the help of reliable forecasting tools.
Table 3.1: Main areas in numerical optimisation

<table>
<thead>
<tr>
<th>Mathematical Programming</th>
<th>Variational Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objective:</strong> find the best points that optimise the economic model.</td>
<td><strong>Objective:</strong> Find the best functions that optimise the economic model.</td>
</tr>
<tr>
<td><strong>Mathematical formulation:</strong></td>
<td><strong>Mathematical formulation:</strong></td>
</tr>
<tr>
<td>Optimise: y(x)</td>
<td>Optimise: ( I[y(x)] = \int F[y(x), y'(x)] )dx</td>
</tr>
<tr>
<td>Subject to: ( f_i(x) \geq 0 )</td>
<td>Subject to: Algebraic, integral or differential equation constraints</td>
</tr>
<tr>
<td>where: ( x = (x_1, x_2, ..., x_n) )</td>
<td></td>
</tr>
</tbody>
</table>

**Methods**
- Analytical methods
- Geometric programming
- Linear programming
- Quadratic programming
- Convex programming
- Dynamic Programming
- Non-Linear programming (multivariable)
- Integer programming
- Separable programming
- Goal programming (multicriterion)
- Combinatorial programming
- Maximum principle (discrete)

Before 1940, relatively little was known about methods for numerical optimisation of functions of many variables. It is clear now that the advent of the computer era was paramount in the development of optimisation methods and indeed in the whole of numerical analysis.

The 1940s and 1950s saw the introduction and development of an important branch of the subject known as **dynamic programming**. Due to the fact that the ideas of continuous variational methods necessarily have to be incorporated when solving a problem of dynamic nature (as the real-time control of urban drainage networks during storm events) a certain attention will be given to this topic too.

Numerical optimisation relies heavily on the concepts and techniques of matrix and numerical linear algebra. It is assumed here that the reader has a knowledge of this subject. An excellent overview of these topics can be found in Gill et al (1981, p. 7 to 56).
3.2 Mathematical background

3.2.1 The continuous form of the optimal control problem

The problem of optimal, real-time control of urban drainage networks can be classified as a continuous (dynamic) optimal control problem. The continuous form of the problem offers as a solution a continuous function which describes the optimal control. However, because of their complexity an analytical formulation is not possible for the vast majority of practical problems. Therefore a discrete formulation is required, which in our problem is achieved by a stagewise representation scheme of the time-continuous process.

The general, continuous optimisation problem (Polak, 1971, p. 2) is formulated as:

Given

\[ \frac{d}{dt} x(t) = f (x(t), u(t), t), \quad (3.1) \]

with: \( x(t) \in \mathbb{R}^y, u(t) \in \mathbb{R}^u, t \in [t_0, t_f] \),

where:

- \( x(t) \) : state of the system at time \( t \)
- \( u(t) \) : control applied to the system at time \( t \)
- \( t_0 \) : initial time
- \( t_f \) : final time (sometimes not specified)

find a measurable control function \( u(.) \) defined on \([t_0, t_f]\), a corresponding trajectory, \( x(\cdot) \), and the final time \( t_f \) (if not specified) which minimise the cost functional:

\[ \int_{t_0}^{t_f} f^0 (x(t), u(t), t) \, dt + \phi(x(t_f)), \quad (3.2) \]

subject to the constraints:

\[ s(u(t)) \leq 0 \quad \text{for } t \in [t_0, t_f] \quad (3.3) \]

\[ g(x(t), t) = 0 \]

\[ q(x(t), t) \leq 0 \quad \text{for } t \in [t_0, t_f] \quad (3.4) \]

where \( f: \mathbb{R}^y \times \mathbb{R}^u \times \mathbb{R}^i \rightarrow \mathbb{R}^y \) and \( f^0: \mathbb{R}^y \times \mathbb{R}^u \times \mathbb{R}^i \rightarrow \mathbb{R}^1 \) are continuously differentiable in \( x \) and in \( u \), \( \phi: \mathbb{R}^y \rightarrow \mathbb{R}^1 \), \( g: \mathbb{R}^y \times \mathbb{R}^i \rightarrow \mathbb{R}^1 \) and \( q: \mathbb{R}^y \times \mathbb{R}^i \rightarrow \mathbb{R}^m \) are continuously differentiable in \( x \),
and $s: \mathbb{R}^n \rightarrow \mathbb{R}^m$ is continuously differentiable in $u$.
In addition, $f, f', \partial f/\partial x, \partial f/\partial u, \partial f'/\partial x, \partial f'/\partial u, g, q, \partial g/\partial x, \partial q/\partial x$ are piecewise continuous in $t$.

In other terms, it is possible to state that $f'$ describes continuously in time the response of the controlled system to the applied controls (the function containing the set of the system state variables). On the other hand, $f^0$ is a cost functional which, in addition, evaluates the obtained trajectory through the time interval $[t_0, t_f]$ as well as the final state of the system in relation to the objectives imposed. The $s'$ represents a set of functions which demarcate a feasible domain for the control function ($\delta(\cdot)$). This set of functions can be interpreted as the set of active constraints of our problem. $g'$ and $q'$ are equality and inequality constraints, respectively, imposed upon the trajectory function. The set of $g'$ constraints is present in systems where a part of the trajectory is predefined to follow exactly the imposed constraints. On the other hand, the $q'$ describes the region of feasible states and is determined by the physics of the system.

### 3.2.2 The discrete form of the optimal control problem

Unfortunately, the continuous form of the optimal control problem is not suitable for its representation on a digital computer. Therefore, the most useful **discrete form of the optimisation problem** is actually used. This form can be stated as follows; Given a dynamical system described by the difference equation:

$$x_{i+1} - x_i = f_i(x_i, u_i), \quad x_i \in \mathbb{R}^v, \quad u_i \in \mathbb{R}^u, \quad i = 0, 1, 2, \ldots, k-1 \tag{3.5}$$

where:

- $x_i$ : state of the system at time $i$
- $u_i$ : control applied to the system at time $i$

find a control sequence $\delta = (\delta_0, \delta_1, \delta_{k-1})$ and a corresponding trajectory:

$$\mathcal{X} = (x_0, x_1, \ldots, x_k)$$

determined by (3.5), which minimise the cost functional:

$$\sum_{i=0}^{k-1} f^0_i(x_i, u_i) + \phi(x_k), \tag{3.6}$$

subject to the constraints:

- $s_i(u_i) \leq 0, \quad i = 0, 1, \ldots, k-1$,
- $g_i(x_i) = 0$,
- $q_i(x_i) \leq 0, \quad i = 0, 1, \ldots, k$,
where \( f : \mathbb{R}^r \times \mathbb{R}^u \to \mathbb{R}^r \) and \( f^\circ : \mathbb{R}^r \times \mathbb{R}^u \to \mathbb{R}^1 \), \( \varphi : \mathbb{R}^r \to \mathbb{R}^1 \), \( s_i : \mathbb{R}^u \to \mathbb{R}^{u_i} \), \( g_i : \mathbb{R}^r \to \mathbb{R}^l \) and \( q_i : \mathbb{R}^r \to \mathbb{R}^{m_i} \) are continuously differentiable functions.

The integer \( k \) represents the duration of the control process. Yet another optimal control formulation is the general non-linear optimisation problem which can be found in Polak (1971, p. 1).

The result obtained by solving the discrete optimisation problem represents a sequence of control decisions distributed along the duration of the control process (event horizon). The control function is then replaced with piecewise constant functions of \( k-1 \) equidistant discontinuities. It can be proved that by increasing the number of discrete decision points to infinity the discrete form of the optimal control form becomes the continuous form (Polak, 1971, p. 3).

### 3.2.3 Methods of solution

Discrete optimisation problems of dynamic nature (as the problem of optimal real-time control of urban drainage networks) can in general be solved by two main numerical approaches, namely:

a) **Dynamic programming**

This method, independently attributed to Richard Bellman and L. S. Pontryagin is considered the most general method of solution of the discrete OCP. The approach is based on the stage structure formulation of the problem, that is, the conversion of the large and complicated optimisation problem into a series of smaller subproblems, related to stages which are interconnected through transition functions. The result is a series of partial optimisation steps requiring a reduced total effort to find the optimum solution.

The dynamic programming algorithm can then be applied to find the optimum of the entire process by using the connected partial optimisation of the smaller subproblems (one for each stage). In a way, the state variables (which link the stages) serve as the path for the dynamic programming algorithm to perform the optimisation of the entire process. The optimisation process within each stage can be carried out by any of the available optimisation techniques.

The dynamic programming method has a unique nomenclature which must be defined before proceeding any further. An individual process (or a unit of time) can be represented as a stage. A stage globally represents the economic model of the unit. Thus, the economic model is called the return function \( R_i(s_i, d_i) \) and gives the measure of profit for the stage. The \( d_i \) represents the set of decision (control) variables which can be manipulated independently, while \( s_i \) represents the set of state (connecting) variables which are the input to the stage from the former stage. Decision variables and state variables form the set of independent variables in dynamic programming.
There are transition functions \((s_i = T_i (s_j, d_j))\), at each stage, which are in charge of connecting the stages. This process is illustrated in figure 14.

![Figure 14: Representation of a three-stage process in dynamic programming](image)

As suggested, a number of partial optimisations are performed with the purpose of obtaining the set of independent variables \(d_i\) and \(s_i\) which optimises (minimises or maximises) a return of the economic model \((R_i (s_i, d_i))\) defined at each stage. This information is finally used in order to locate the optimum for the entire process.

The shortest route problem -see for example (Pike, 1986, p. 309)- is often used to explain the performance of dynamic programming algorithms.

Although different implementations are found in relation to the characteristics of the system, dynamic programming often starts at the final stage of the control process and moves gradually towards the beginning. The method has been very well developed, although so far its application has been restricted due to the already explicated problem of the 'curse of dimensionality' (section 2.1).

**b) Discrete-continuous [non-linear multivariable] methods**

A typical continuous optimisation problem is described in equation (3.1). Another approximated solution to the continuous optimal control problem is possible due to the fact that some algorithms (specially the non-linear programming algorithms) for static optimisation can be used more or less successfully for the solution of the optimal control problem.

The discrete-time formulation of the optimal control problem can be transcribed (Polak, 1971, p. 3) into the non-linear programming problem as follows:

Consider the time-discrete control problem, expressed in equation (3.5), and let:

* The transition functions are also called constraint equations since they impose the restrictive condition that the input to a stage must be a function (for example, \(s_i = s_j\)) of the output from the former stage.
so as to express the system’s response (solution of equation 3.1) to the given control expressed as \( (\bar{\mathbf{u}} = (u_0, u_1, \ldots, u_{k-1}) ) \). The 'k' is an integer expressing the total length of the control process, taken as a number of discrete intervals, and the 'i' determine distances from the start of the control process in the same sense. Further we can write:

\[
\mathbf{f}^0(z) = \sum_{i=0}^{k-1} \mathbf{f}_i^0(\mathbf{x}_i(x_0, u), u_i) + \mathbf{v}(\mathbf{x}_k(x_0, u));
\]

and then the set of constraints is expanded to:

\[
\mathbf{r}(z) = \begin{bmatrix}
g_0(x_0) \\
g_1(x_1, u) \\
\vdots \\
g_k(x_k(x_0, u))
\end{bmatrix};
\]

\[
\mathbf{f}(z) = \begin{bmatrix}
g_0(x_0) \\
g_1(x_1, u) \\
\vdots \\
g_k(x_k(x_0, u)) \\
s_0(u_0) \\
\vdots \\
s_{k-1}(u_{k-1})
\end{bmatrix}.
\]

It is possible to see how, with these definitions, the discrete-time optimal control problem (3.5) reduces to the form of the non-linear control problem (Polak, 1971, p 3).

In order to approximate the continuous optimal control, time must be broken into discrete intervals thus allowing the control vector (also referred to as the 'policy' vector) to be modified at the beginning of each interval. Each stage in the discretised control contains the set of all possible states of the system at the beginning of the time interval. As the size of the time intervals vanishes (or approaches zero size) the optimal control strategy for the discretised problem approximates the optimal control strategy for the continuous-time problem.

Then, some proper interpolation technique is applied in order to replicate the 'discrete-continuous' control strategy. The applicability of each particular static optimisation algorithm
depends on the nature of the actual problem, the acceptable level of simplification and the resources available to solve the optimal control problem.

The problem of real-time control of urban drainage networks does not allow the successful linearisation of the problem (Martin, 1993). Therefore, use must be made here of non-linear multivariable optimisation techniques as an alternative to dynamic programming. As a matter of fact, several optimal real-time control problems have been successfully solved by making use of static optimisation methods (see Tomicic, 1989).

Strong non-linearities of the economic model and interrelations among constraints and dependent variables are sources of additional complexity for this case, making the solution of each particular problem a very difficult and complicated task.

Similarly to dynamic programming methods, these solutions lead to a very fast growth of the size of the overall problem with increasing refinements in the accuracy of the solution.

Non-Linear programming methods (NLP) offer big advantages particularly in those cases where the constraints and the economic model of the system are interrelated (non-separable) benefits.

3.2.4 Optimal control problem in urban drainage networks

The specific features of the algorithm play a central role in controlling the so-called 'curse of dimensionality' in these problems. Several ideas can be used in order to constrain this phenomenon to an extent which allows the real-time implementation of discrete-continuous NLP techniques. Although they will be discussed in detail in section 5.3, we will just state now that four main approaches have been investigated, developed and implemented within the present study in order to solve this problem. These are (in order of importance):

a) The use of knowledge* about the domain in order to restrict the search space.

b) The use of knowledge about the domain in order to guide (steer) the main stages of the numerical optimisation process.

c) The use of knowledge about the domain in order to specify optimal parameters for performing the single-line search.

d) A built-in grid-refinement system to specify the characteristics of the stagewise solution for the numerical optimiser.

* The reference made here to the use of knowledge about the domain do not yet establish the necessary distinction between 'deep' and 'shallow' knowledge. They are both conveniently used according to the need of the particular subtask to be solved.
In this way, a framework has been developed to perform a process of approaching a quasi-optimal solution to the discrete-continuous optimal control problem in successive stages of increasing level of accuracy.

This process can be stopped at any time by the system’s analyst. In this way a very fast, initial, logical optimisation step is performed which is aimed at obtaining the initial vectors for the numerical optimiser and a first iterate of the gradient matrix. A fully-non-linear, numerical optimisation step is then performed aimed at further approaching the 'optimal' solution.

This framework has been applied to the on-line control of the urban drainage network of the city of Gothenburg, Sweden. The real-time control of a storm event over a forecasted horizon of twelve hours has been performed here in about two CPU-hours, thus providing safe margins for real-time implementation in this case.

3.2.5 Necessary and sufficient conditions for the existence of an extreme in a region

The necessary conditions for the existence of an extreme in a given region (referred to as a hyper-surface in the case of more than 3 independent variables) are summarized by the Weierstrass theorem, which is proved by contradiction:

"Every function that is continuous in a closed domain possesses a maximum and minimum value either in the interior or on the boundary of the domain."

In order to develop the criteria (sufficient conditions) for a local maximum or minimum for a stationary point \((\mathbf{x}_*)\) in a function \(f\) of \(n\) independent variables, it is necessary to perform a Taylor's series expansion about this point:

\[
f(\mathbf{x}) = f(\mathbf{x}_*) + \sum_{j=1}^{n} f_{x_j}(\mathbf{x}_*) (x_j - x_{j*}) + \frac{1}{2} \left[ \sum_{j=1}^{n} \sum_{k=1}^{n} f_{x_j x_k}(\mathbf{x}_*) (x_j - x_{j*}) (x_k - x_{k*}) \right] + \text{higher order terms}
\]

(3.11)

where the subscripts \(x_j x_k\) denote second order partial derivatives. \(\mathbf{x}\) must be selected sufficiently close to \(\mathbf{x}_*\) so that the higher order terms become negligible compared to the second order terms. As the first order derivatives are zero at the stationary point, the equation (3.11) can be written, in a matrix-vector notation as:
\[ f(x) = f(x_*) + \frac{1}{2} (x - x_*)^T H_* (x - x_*) \] (3.12)

where \( x \) is the column vector of \( n \) independent variables and \( H_* \) is a matrix of second order partial derivatives (the so-called Hessian matrix), evaluated at the stationary point \( x_* \). The second term in (3.12) is the so called differential quadratic form \( Q \).

\[
H_* = \begin{bmatrix}
\frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2}
\end{bmatrix} \quad (3.13)
\]

After analysing the definiteness of the principal submatrices of \( H_* \) and computing its determinants the following sufficient conditions are derived for the case of \( n \) independent variables:

\( f(x_*) \) is a minimum when: \( H_{ii} > 0 \) for \( i = 1, 2, \ldots, n \)

\( f(x_*) \) is a maximum when: \( H_{ii} < 0 \) for \( i = 1, 3, 5, \ldots \)

and: \( H_{ii} > 0 \) for \( i = 2, 4, 6, \ldots \) (3.14)

3.3 Overview of unconstrained multi-variable search methods

In general, multi-variable search methods are divided into two main groups. These are:

1. Unconstrained optimisation problems
2. Constrained optimisation problems.

Although most real-life problems are constrained, the understanding of unconstrained multi-variable search methods is important because basically all the ideas derived from unconstrained search also hold for the constrained problem. The inclusion of constraints in real-world engineering problems will be explained in section 3.5.

In this section our attention will be devoted to unconstrained multi-variable search methods in relation to the optimisation of the - non-linear - problem of on-line control of urban drainage networks. As explained in Chapter 2, this is a very dynamic problem in which the selection of a proper optimisation methodology is essential for the real-time solution.

Multi-variable search methods are iterative. This means that in existence of the right conditions* a gradual convergence towards the optimum is expected from iteration to iteration. Basically, there are two main steps in each iteration:
1) Determination of an appropriate search direction from the **current point**. In the case of non-linear minimisation this direction is referred to as the descent direction, for it points towards a direction in which a better point (in terms of cost) is supposed to be found.

2) When such a direction is determined, a line search procedure is performed. This is essentially a single-variable optimisation along the search direction, with variable step length, whose objective is to find the best possible solution along the search direction (stationary point). After finding a stationary point, the current point is changed to a new position.

The procedure is finished when a move of finite length from the current point produces a worse solution meaning that a local optimum has been found. Robust real-life multivariable search algorithms show stable convergence towards the optimum in any point of the search space.

The mathematical form of this procedure is:

\[
\mathbf{x}_{(k+1)} = \mathbf{x}_{(k)} - \alpha \mathbf{H}_{(k)} \nabla_y (\mathbf{x}_{(k)})
\]  

(3.15)

where:

- \(\mathbf{x}_{(k+1)}\) = the 'k+1'-th iterate
- \(\mathbf{x}_{(k)}\) = the 'k'-th iterate
- \(\alpha_{(k)}\) = the step length along the search direction
- \(\mathbf{H}_{(k)} \nabla_y (\mathbf{x}_{(k)})\) = the search direction in the 'k'-th iteration (\(S_{(k)}\))

Unconstrained, multi-variable search methods can again be grouped into **first and second derivative methods** (Gill, 1981). Second derivative methods require an additional effort at each iteration for the computation of the **Hessian matrix** and they are applied in cases where the first two derivatives of the objective function are available. On the other hand, first derivative methods require less computational effort to approximate the inverse of the **Hessian matrix** in each iteration (which can be done by using finite difference approximations to the derivatives), thus increasing the speed and robustness of these algorithms.

First derivative methods include the so-called quasi-Newton methods and **discrete Newton's methods**, while second derivative methods include the family of the Newton-like methods. Quasi-Newton methods are widely recognized (Dennis *et al.*, 1977; Pike, 1986; Fletcher, 1987) as the most effective, robust and elaborate of all multi-variable search procedures.

Wilde (1964) has proposed a strategy (analogous to the strategy of chess) for describing the performance of multivariable search methods that contains some relevant ideas. This strategy has an **opening gambit, a middle game and an end game**, which are all equally important to

* i.e., if the methodology has been properly chosen according to the characteristics of the particular problem to be solved and the optimal parameters have been specified to perform the numerical optimisation process.
the success of the overall optimisation process. In the opening gambit, a starting point is selected. The middle game involves moving from this starting point to a point near the optimum as rapidly as possible. In the end game a quadratic fit to the economic model is performed (by some methods) in order to avoid stopping at a saddle point or sharp ridge.

However, in the problem of on-line control of urban drainage networks, the specification of a proper initial vector may be decisive to the convergence towards the global optimum. The result of the opening gambit is in general, external (supplied from outside) to the numerical algorithm, at least during the first optimisation cycle. Therefore, the comparative analysis of the effectiveness of optimisation algorithms have to be performed on the basis of the analysis of their performance during the middle and end game. Quasi-Newton methods are particularly strong at these stages, if we consider their rate of convergence towards the optimum.

Although we shall return to this topic in section 5.3, it can be said that, within this framework, the initial search vectors are provided by a starting, on-line, logical optimisation step.

When the Hessian matrix \( (H) \) is indefinite (or analytical first order derivatives are not available), a 'popular solution' (Gill et al, p. 107) is to construct a related, positive-definite matrix \( (H^+) \). In these cases because \( H^+ \) is positive-definite, the convergence towards a stationary point is guaranteed.

As explained above, quasi-Newton methods rank among the most powerful methods for multivariable optimisation. Their stability in most search spaces in the presence of inexact line searches, as well as their rate of convergence towards the optimum even in the vicinity of the stationary point, were decisive reasons why this algorithm was selected for implementation in this study in the numerical methodology that is built in the combined logical-numerical framework.

3.3.1 Quasi-Newton methods. The matrix update system

The effectiveness of a multivariable optimisation procedure depends on several, interrelated factors. These are, the optimisation theory, the computer program, the programming language, the skills of the programmer, the hardware platform and, of course, the problem to be solved. Yet, the set of initial vectors supplied to the numerical optimiser has a great influence on the observed performance.

The first step when attempting to develop an effective numerical optimisation system is to select the most suitable general method. Quasi-Newton methods are attractive for designers of numerical optimisation algorithms due to several of their features. These are:

a) They show a superlinear rate of convergence towards the optimum, a property which is maintained even in the close neighbourhood of the stationary point. In these cases the property of quadratic termination is observed if the economic model has been defined on the basis of smooth objective functions. The above holds even if inexact line searches are performed (unconditional search stability).
b) They do not require second derivatives. Instead, the Hessian matrix is approximated by a positive-definite, related matrix \((\tilde{H})\). Therefore, the computational effort required to approximate \(H\) at each iteration is considerably smaller than in Newton methods.

c) They provide very robust algorithms when finite-difference approximations to the derivatives have to be used due to the no availability of analytical derivatives.

For Quasi-Newton methods, the search algorithm is given by equation (3.15). As might be observed from 3.15 the \(\tilde{H}\), approximations of the Hessian matrix contain information about the curvature of the search space. For these methods \(H\) is a series of matrices beginning with the identity matrix \(I\) (or any positive definite matrix), and ending with the inverse of the Hessian matrix \(H^{-1}\). As the algorithm proceeds, a quadratic approximation of the profit function is constructed from gradient measurements. This is the source of their successful performance in smooth optimisation.

3.3.2 The Broyden-Fletcher-Goldfarb-Shanno updating formula

Within the quasi-Newton family, the so-called Broyden-Fletcher-Goldfarb-Shanno rank-two updating formula provides one of the most elaborated procedures for unconstrained multi-variable search (Fletcher, 1987; Dennis, 1977) because it steadily improves the search by incorporating in its matrix the information gathered about the search space in the searches performed before.

The Broyden-Fletcher-Goldfarb-Shanno updating formula for the series of \(H_{(k)}\) matrices is:

\[
H_{(k+1)} = H_{(k)} + A_{(k)} + B_{(k)}
\]  

(3.16)

where the matrices \(A_{(k)}\) and \(B_{(k)}\) are determined according to the following expressions:

\[
A_{(k)} = -\left[ \frac{H_{(k)} Y_{(k)} \delta_{(k)}^T + \delta_{(k)} Y_{(k)} H_{(k)}}{\delta_{(k)}^T Y_{(k)}} \right]
\]  

(3.17)

\[
B_{(k)} = \left[ \frac{Y_{(k)} H_{(k)} Y_{(k)}}{\delta_{(k)}^T Y_{(k)}} \right] \left[ \frac{\delta_{(k)} \delta_{(k)}^T}{\delta_{(k)}^T Y_{(k)}} \right]
\]  

(3.18)

where:
\[ \delta_{(k)} = x_{(k+1)} - x_{(k)} \]  

(3.19)

\[ \gamma_{(k)} = \nabla y(x_{(k+1)}) - \nabla y(x_{(k)}) \]  

(3.20)

represent measured (computed) offsets in the control variable \( (x_k) \) and in the gradient \( (\nabla y(x_k)) \) from iteration to iteration.

The matrices \( A_{(k)} \) and \( B_{(k)} \) have been constructed so that their sums have the specific properties shown below:

\[ \sum_{k=1}^{n} A_k = H^{-1} \]  

(3.21)

\[ \sum_{k=1}^{n} B_k = -H_0 = -I \]  

(3.22)

It is again possible to observe - now in mathematical terms - how the quasi-Newton methods gradually incorporate during the search process the information they gather about the solution space during the searches performed before. Unlike what occurs in Newton methods, this updating system ensures the maintenance of the property of self-orientation in the close neighbourhood of the optimum. Moreover, the property of minimisation is ensured by the positive-definiteness of the approximation to the Hessian matrix \( (H_0) \).

It is believed that the matrix update system which characterises the Quasi-Newton methods is essential to the successful implementation of a numerical algorithm for multivariable search to the problem of real-time control of urban drainage systems. As this is a problem of a very dynamic nature in which the system's conveyance cannot be neglected in relation to the storage, the short duration of the event horizon (usually limited to few hours) gives rise to the need for a 'good' starting vector for the numerical optimiser in order to ensure convergence to the 'global' optimum (or at least a 'good enough' point). In this situation, it is necessary to ensure convergence to the extreme in the 'close' neighbourhood of the point (where a Newton-like method would fail to converge showing a hemstitching pattern). This is exactly one of the strong sides of the Quasi-Newton methods.

3.4 The complementary single-line search subproblem

3.4.1 Introduction

The single-line search is a very important step in multivariable optimisation. A good choice here is important since this algorithm has a considerable influence on the overall performance of the method in which it is embedded. The availability or not of the first order derivatives
Numerical Optimisation

(in connection with the method employed, see above) is a primary consideration: if derivatives are not available (Fletcher, 1987, p. 33) then there is not much theory available to act as a guide to how the line search should be terminated. Then, practical -engineering- criteria are of major importance. In on-line control, accuracy might be a secondary factor. An out-of-date solution is irrelevant to real-time control regardless its accuracy.

Therefore, a sound numerical optimisation methodology must be as flexible as possible, that is, it must performed according to user-defined criteria in order to satisfy the important real-time constraint.

The control of the number of experiments to be performed in the single-line search, in relation to the reduction of the initial interval of uncertainty, is one of the decisive factors in real-time implementation of non-linear multivariable algorithms.

It must be taken into account that the decision of discarding - or keeping - intervals in the single-line search can only be made on the basis of a comparative analysis of the cost associated with experiments (vectors in the search space) which in our problem is obtained after integration of cost time series from the results of fully dynamic - and time consuming - simulations.

A line search algorithm is an iterative method which generates a sequence of estimates (experiments) denoted here as \( \{x_i \} \). The sequence terminates when an iterate is located which satisfies some 'standard' criteria for an acceptable point. It can also be controlled according to the number of iterations calculated to obtain a certain reduction in the interval of uncertainty.

There are two distinct parts of any line search algorithm. These are:

a) **The bracketing phase.** In this phase the objective is to find a *bracket*, that is a non-trivial interval, say \([a, b]\), which is known to contain the extrema sought.

b) **The sectioning phase.** In this phase the bracket is sectioned (divided) so as to generate a sequence of brackets \([a_i, b_i]\) whose lengths tend to zero. In this phase several experiments are placed at intervals which vary from method to method. The interval that does not contain the optimum, is *systematically* discarded.

Search plans can be classified as either simultaneous or sequential. In a simultaneous search the location of all experiments in the sequence is specified and the outcome of the measurements is obtained 'at the same time'. In a sequential search scheme, the outcome of

\footnote{In constrained optimisation the length of the initial interval of uncertainty may be taken as the distance between the current point and the point on the boundary which is indicated (pointed to) by the search direction.}
an experiment is determined before another experiment is made. Sequential search methods show advantages in relation to simultaneous search methods which increase exponentially with the number of experiments (Wilde, 1967), however, sometimes simultaneous methods must be applied since a reasonable guess of the outcome of an experiment* is not always available. There are several available techniques within the field of sequential methods. The main techniques are:

- Fibonacci search
- Golden section search
- Lattice search

3.4.2 The Fibonacci search

For convenience we will concentrate on the Fibonacci search. As will be explained in next section the Fibonacci search is considered to be the most efficient of all search plans but it requires the number of experiments to be specified in advance.

The Fibonacci search is a sequential search plan which proceeds by conducting two experiments at the time, in order to achieve the interval reduction. The derivation of the Fibonacci method is obtained backwards, that is, by placing the last two experiments optimally in the interval preceding the final interval of uncertainty (\(I_n\)). This technique is based upon the Fibonacci numbers (F), which have the following properties:

\[
F_0 = 0, \\
F_1 = 1, \\
F_n = F_{n-1} + F_{n-2}
\]  

(3.23)

It can be proven that the length of the final interval of uncertainty (\(I_n\)) can be obtained from the length of the initial interval of uncertainty (\(I_0\), which is known to contain the optimum) according to the following expression:

\[
I_n = \left[ \frac{1 + \varepsilon}{F_{n+1}} \right] I_0
\]

(3.24)

* In relation to our problem (sequential) what it is possible to do, is to use information from past experiments in order to locate subsequent experiments. In this way only one additional function evaluation is needed when locating a new experiment since the former one is used as 'reference'. This allows a significant reduction in the number of function evaluations required in the numerical optimisation process.
where:

- $I_n$ : Final interval of uncertainty.
- $I_0$ : Initial interval of uncertainty.
- $\varepsilon$ : Fractional resolution based on the initial interval of uncertainty.
- $F_{n+1}, F_{n-1}$ : Fibonacci numbers in position $n+1$ and $n-1$ in the series, respectively.
- $n$ : number of experiments in the Fibonacci search.

The extreme fractional resolutions are related to the corresponding intervals according to:

$$\delta I_n = \varepsilon I_0 \quad (3.25)$$

where:

- $\delta$ : Fractional resolution based on the final interval of uncertainty.

Now we must provide the expressions for locating the first two experiments ($\alpha_1, \alpha_2$) in the Fibonacci search since once this has been done, the rest of the experiments will be sequentially located in Fibonacci proportion in the non-discarded interval according to the minimax principle, which states that the best search plan (Fibonacci) locates two experiments symmetrically in the accepted interval separated by its resolution ($\varepsilon$). Thus, the second experiment ($\alpha_2$) will be located at a distance $I_2$ (given by 3.24) from the left side of the interval according to:

$$\alpha_2 = I_2 = F_n I_n - \varepsilon F_{n-2} I_0 \quad (3.26)$$

The first experiment ($\alpha_1$) will be located symmetrically to $\alpha_2$ at a distance $I_2$ from the right side of the interval according to:

$$\alpha_1 = I_3 = F_{n-1} I_n - \varepsilon F_{n-3} I_0 \quad (3.27)$$

The procedure continues after the first two experiments have been evaluated by discarding the interval that does not contain the optimum* and placing the remaining experiments symmetrically to the previous successful one. Finally, the optimum in the line is rapidly found within the tolerance specified by the final interval of uncertainty. The procedure is illustrated in figure 15, for the search of a maximum in the line. The rapid interval reduction which is obtained with the fibonacci search can be shown by using a simplified form of the equation (3.24) in which $\varepsilon = 0$ so that:

$$I_n = \frac{I_0}{F_{n+1}} \quad (3.28)$$

Therefore the interval reduction rate increases very rapidly with the number of Fibonacci experiments. For example, with the following conditions:

---

* On the basis of cost analysis in the extremes of the chosen sub-interval, during the sectioning phase.
\( \varepsilon = 0, n = 5, I_0 = 1; \)

an interval reduction of *five times* is achieved in the initial search interval since the fifth term in the series of Fibonacci numbers is also the number five. With just two more experiments (seven) this interval reduction is now *thirteen times*, since this is the corresponding (seventh) Fibonacci number in the series.

However, this reduction is not calculated in such a straightforward way since often the fractional resolution \( \varepsilon \neq 0 \). In practice, the number of Fibonacci experiments is usually selected between three and six within the present combined framework designed for real-time control of urban drainage networks.

**Figure 15: Locating experiments for the Fibonacci search**

3.4.3 Measuring line search effectiveness

The purpose of this section is to define the criteria used for measuring line search effectiveness. The need for developing an on-line optimisation system using a fully non-linear numerical methodology give rise to the necessity to implement the most effective methods available for numerical optimisation. As explained, the single-line search is an important part in any optimisation methodology.

In order to compare search plans, the measure of effectiveness must be independent of the functions being optimized. This is required to eliminate functional dependence, bias and luck. Consequently, it is necessary to have the measure of the effectiveness of search plans depending on the placements of the experiments and not on its outcome (Pike 1986, p 165). Therefore, the criterion to be used in comparing search plans is the size of the *largest* interval of uncertainty possible, having determined the location of the experiments. This does not
depend on the outcome of the experiments. This is often referred as the interval reduction capability.

The minimax principle states that the best search method is the one which has the smallest of the largest intervals of uncertainty as defined above. In mathematical terms this is written as:

\[ I_n^* = I_n(\alpha_n^*) = \min_{\alpha_n} [I_n(\alpha_n)] \quad (3.28) \]

In order to evaluate the largest final interval of uncertainty for each search plan it is necessary to consider that each of the \( n \) experiments \( \alpha_1, \alpha_2, ..., \alpha_n \) may have the largest possible outcome.

For simplicity, let us consider a sequential search with only three experiments \( n = 3 \). From the minimax principle, the best search plan is to place two experiments symmetrically in the interval separated by the resolution. Therefore:

\[ I_2^* = 0.5 + \frac{\varepsilon}{2} \quad (3.29) \]

and:

\[ I_3^* = \max [\alpha_2, \alpha_3 - \alpha_1, 1 - \alpha_2] \quad (3.30) \]

According to these criteria, the Fibonacci search is considered to be the best plan of all the line search procedures, because the progression of the series of Fibonacci numbers (eq. 3.23) follows exactly the plan for the location of experiments specified in the minimax principle. The 'disadvantage' of this technique is that it requires that the number of experiments be specified in advance.

However, since the interval reduction is predictable under certain conditions, this only introduces a minor inconvenience. This topic will be explained in section 5.5.

### 3.5 The constrained problem

#### 3.5.1 Introduction

Most of real-world problems in optimisation are constrained problems. Only rarely is an unconstrained problem found in engineering. Constraints, include additional - in some cases considerable - complexity in the optimisation problem. In the presence of constraints, the independent variables cannot be varied freely throughout the solution space. Therefore, only a limited region in the solution space contains the set of physically feasible solutions to the real problem. This is known as the feasible region (or domain). Any solution outside this area involves violation of the active constraints and has to be rejected.
In general terms, there are two approaches to the solution of the problem. These are:

a) Indirect methods.
b) Direct methods.

*Indirect methods* transform the original constrained problem into an *equivalent* unconstrained problem and then use any of the methods of unconstrained optimisation in order to solve it. On the contrary, *direct methods* deal with the constrained problem as such taking into account the feasible region and the set of constraints.

Each of the methods has its advantages and disadvantages. However, there is no consensus on this complex topic. What it seems clear now is that the theory of constrained optimisation is related in an important way to the concept of the so-called *Lagrange multipliers*.

What still has to be found is a general method which will unify all the existing approaches to the problem in the way that - for example - the Simplex algorithm does in Linear programming. In practice, the selection of the method for constrained optimisation is often determined by the characteristics* of the problem to be solved.

### 3.5.2 Indirect methods for constrained optimisation

Some of the *indirect* methods are:

*The Method of Penalty and Barrier Functions (MPF)*

*Augmented Lagrangian Functions (ALF)*

We will concentrate here only on the Method of Penalty Functions. A complete description of the Augmented Lagrangian Functions can be found in Pike (1986, p. 263).

The idea of the Penalty and Barrier methods is to transform the original (constrained) function $f(x)$ into an equivalent (unconstrained) *penalty function* $\Psi = (x, r)$ by adding the so-called *penalty terms* which cause a very large increase of the function value (large enough to dominate the original value of $f(x)$) in zones outside or even in the vicinity of the constraint, thus deflecting the trajectory towards the feasible domain. The scalar ‘$r$’ is a constant that regulates the increase imposed on the penalty function.

A further distinction is possible between the exterior and the interior penalty function according to the way in which they approach the constraint. Exterior penalty functions can be expressed as follows:

$$
\Psi (x, r) = f(x) + r \sum_{i}^{m} (g_j)^2
$$

\[3.31\]

* *i.e.*, the size of the problem (the solution array), the number of independent variables, the kind of constraints and economic model, the set of active constraints and other (including the time) constraints.
Numerical Optimisation

where:

\[ z : \text{a non-negative constant (usually taken as 2)} \]
\[ <g> : \text{a 'bracket function', defined as:} \]
\[ (g) = \begin{cases} 
  g, & g \geq 0 \\
  0, & g < 0 
\end{cases} \quad (3.32) \]

Large values of 'r' force strong deflections towards the feasible region but at the same time cause excessive difficulties in the minimisation of \( \Psi \).

The interior penalty function has the following form:

\[ \Psi (x, r) = f(x) - r \sum_{1}^{m} \frac{1}{g_j(x)} \quad (3.33) \]

The operational characteristics are similar since they just establish a differentiation according to the way that the constraint is approached. The performance of the MPF is shown in figure 16 for the following penalty function (\( \Psi \)):

\[ \Psi (x, s) = \sqrt{x} + s (x - 1)^2 \quad (3.34) \]

### 3.5.3 Direct methods for constrained optimisation

On the other hand, the main direct methods are:

- **Successive Linear Programming (SLP)**
- **Successive Quadratic Programming (SQP)**
- **Generalised Reduced-Gradient methods (GRG)**
- **Methods of Feasible Directions (MFD)**

Although, it might seem at a first glance that there are sufficient criteria to provide a 'standard ranking' for these techniques (Pike, 1986, p. 278), experience shows, that it is very important to analyze each of the techniques in the context provided by the characteristics of the problem to be solved.

In this sense, no method here is absolutely superior to the others. In effect, the reasons why a particular method might in general not be attractive might disappear (or at least decrease) under certain conditions to such an extent that the method in particular becomes more suitable than another method - usually considered more general - for the solution of our problem.
Figure 16: Trajectory of the penalty function $\phi$ for different values of the penalty multiplier $s$

The numerical methodology built into this combined logical-numerical optimisation framework adopted a direct scheme to deal with the constrained optimisation problem. This decision was made on the basis of the characteristics of the problem's constraints according to our formulation, as well as to satisfy the real-time constraint.

Since the present framework utilises a simplified feasible directions algorithm which takes advantage of the structure of the constraints and economic model according to the formulation employed, the next few lines will be devoted to a general description of this method.

3.5.4 The feasible directions methods

This group of methods has been developed for solving general, non-linear, inequality-constrained optimisation problems. The feasible direction methods start by locating a feasible direction and then they perform a single-variable search along the gradient line in order to locate the optimum. If in the move towards the optimum a constraint is found, the gradient line is projected on the constraint and the search proceeds in this projected-gradient direction along the constraint.
These methods are suitable for 'coupling' with any of the methods of non-linear multivariable optimisation, including of course those of the quasi-Newton family.

The formal statement of this method indicates that any vector 's' is a feasible direction from the point defined by the vector 'x' if there is a step of finite length such that after performing the 'move', the new point is still in the feasible domain. In mathematical terms, this appears as:

$$s^T \cdot \nabla f < 0,$$

i.e. the gradient and the vector 's' form an obtuse angle. That is, the angle between a constraint tangent and the search vector 's' has to be between 90° and 180°.

Another method in the same family, initially developed for non-linear constraints, is the gradient projection method. As indicated by its name, the gradient vector is 'projected' along the active constraint through a so-called projection matrix. The overall performance of these methods is illustrated in figure 17.

### 3.6 Finite-difference approximations to the derivatives

When optimising a smooth function whose exact analytical derivatives are not available, an obvious strategy is to replace the exact gradient with a finite-difference approximation. Unfortunately, this adaptation is non-trivial, and it is essential to consider some rather important modifications of the reasoning developed under the assumption of exact derivatives because of inevitable errors* contained in the approximations.

Due to its importance for constrained optimisation, this topic will deserve our attention in the next section. In this section we will concentrate on the approximation of first-order derivatives since the Quasi-Newton methodology employed here for numerical optimisation ranks as a first-order derivative method.

The first order derivatives can be approximated in several ways. The first is through the so-called forward-difference approximation:

$$\frac{\partial F(x)}{\partial x} = \frac{\Delta F(x)}{\Delta x} = \frac{F(x + \Delta x) - F(x)}{\Delta x}, \quad (3.36)$$

the backward-difference approximation:

$$\frac{\partial F(x)}{\partial x} = \frac{\Delta F(x)}{\Delta x} = \frac{F(x) - F(x - \Delta x)}{\Delta x}, \quad (3.37)$$

or the central-difference approximation:
Figure 17: Illustration of the procedure of the feasible directions methods on a simple example

The above equations hold for the case of single-variable functions (F). In the case of multi-variable functions (f), the approximation formulas become:

for the forward-difference approximation:

\[
\frac{\partial f(x)}{\partial x_i} = \frac{\Delta f(x)}{\Delta x_i} = \frac{f(x_1, x_2, \ldots, x_n + \Delta x_i, \ldots, x_n) - f(x)}{\Delta x_i}
\]  

(3.39)

and so on. It is seen that x now represents a column vector of n components, i.e. \( x = (x_1, x_2, \ldots, x_n) \).

* It is not possible to minimise the importance of this problem. In effect, certain cases have been reported in which certain existing algorithms for minimisation have lost the unconditional-descent property because of 'sparsity destroying matrix effects' caused by errors (see, for example, Gill, et al, 1981).
In our case, the choice of the finite-difference interval ($\Delta x$) is **not a trivial matter** either.

In effect, large values of $\Delta x$ will introduce considerable errors in the approximation of the derivative, thus producing false information about the curvature of the surface especially about *[singular points]*, see figure 18. On the other hand, too small values of $\Delta x$ may create the problem of providing *no information* about the curvature of the economic model since this model is 'mapped' on the basis of the results of numerical hydrodynamic simulations given a control strategy.

In order to explain the later case, let us imagine that the criteria used in order to 'measure' the gradients in the economic model of our problem is the comparative analysis of *global costs* associated with several control strategies in terms of *settings*. In this case, in order to provide elements for carrying out such a comparison these control strategies have to be tested on the system during the process of 'slope evaluation'. For this, it is necessary to generate a *perturbation* in each of the discretized points of the time series of settings at each regulator (solution array) and to measure their costs.

In these terms, the magnitude of the perturbation can be regarded as $\Delta x$. If this parameter is 'too small', *no significant physical change* will be observed in the system, since from a practical point of view the regulators have not been 'moved' (the effect is damped by errors). It is seen that the effect of 'distorted' information may lead to an *erroneous* underlying conclusion like: "*no improvement of the flow situation is observed when shifting the initial setting of the regulator A by an amount $\Delta x$ in the direction $\delta \Rightarrow test the opposite direction p".* This is a typical *noise* problem.

In fact, the correct conclusion would be "*no observable improvement of the flow situation is noticed when shifting the initial setting of the regulator A by an amount $\Delta x$ in the direction $\delta \Rightarrow increase the value of $\Delta x$ and repeat measurement"*

The consequences of such a 'mistake' are serious in the real-time control process. In effect, the 'solution' strategy might be highly misleading. However, there are several solutions to this problem. They are mainly related to the idea of performing an initial, off-line *trial-measurement* in order to gather information about 'significant thresholds' for the control variables. In this way the magnitude of the actual perturbation of the control variable during the slope evaluation process will ensure that this problem of *scaling* is solved. The magnitude of the finite-differences interval is then expressed as a relative value of the magnitude of the trial-measurement.

In practice, the way in which these problems are *ranked* for their 'treatment' within the overall strategy in order to derive the quasi-optimal solution sought in real-time control of urban drainage networks will be discussed in section 5.1.

---

* *Although this effect is minimised to a certain extent in the Quasi-Newton algorithms, since they construct a 'quadratic fit' of the economic model which is based on overall gradient measurements.*
Examples of singular points in optimisation:
A, D: Extremes of the function.
B: Saddle point.
C: Sharp ridge.

Figure 18: Singular points in optimisation

3.7 A word about errors and floating point arithmetic

The purpose of this section is to introduce a few remarks about the role that the main kinds of errors might play in the performance of a numerical methodology for the search of a constrained 'quasi-optimum' in a function of many variables (hypersurface). As suggested in the above section there are cases in which, due to an improper treatment of errors, numerical algorithms for multi-variable search have faced severe problems of different kinds, including failure in application.

Errors arise when attempting to approximate an exact quantity by a computed value. An intuitive measure of error would be 'zero' if the approximations were 'exact', 'small' if the two quantities were 'close' and 'large' if the approximation were 'poor'. Often, a measurement of the absolute value of the error is not satisfactory because it may yield misleading criteria.
in cases where the error itself and the measured quantities are of the same order of magnitude. The so-called relative error also demonstrates certain disadvantages in those cases where the exact values involved tend to zero (see Gill et al, p. 7).

Therefore, in practice it is often convenient to use the following measure of error which combine desirable features of absolute and relative error:

\[ e = \frac{\xi - x}{1 + |x|} \]

(3.40)

where:

- \( x \) : the exact quantity
- \( \xi \) : the approximation

It is observed that this measure tends to behave like relative error when \(|x| >> 1\) and like absolute error when \(|x| << 1\).

The errors that appear when constructing an approximation in optimisation are:

- **truncation errors**, consisting of the neglected terms of the Taylor series expansion of the function \( f(x) \) around the point 'x' (equation 3.11). These errors tend to be 'smaller' when the finite difference interval is decreased. The use of the central finite difference formula helps to reduce this error.

- **condition errors** (also cancellation errors), which appear because of the inexact evaluation of the function \( f \) in points 'x', 'x + h' and 'x - h', according to the finite difference formula applied (Gill, et al, p. 128). This error may play an important role in some problems.

- **rounding errors**, arising during floating-point arithmetic operations on a discrete computer. These errors are usually small in relation to the other two, and they are often neglected.

In general, the magnitude of the truncation and condition errors varies inversely with the magnitude of the finite-difference interval. However, this is not always the case (section 3.6) because this dependency changes once the 'optimal' interval has been reached. This is illustrated in figure 19.

In the problem of real-time control of urban drainage networks the main errors come from the approximations to the derivatives. These approximations are constructed on the basis of gradient measurements on a hypersurface which is already only an approximate model of the real process, since it is provided by hydrodynamic simulations.

The choice of a quasi-Newton algorithm seems to be almost essential to this problem. In effect, these algorithms only require first derivatives. This provides a considerable reduction
in the time employed to construct the economic model for the process in relation to a second-order derivative method, albeit the elimination of a potential source of amplified errors (for the estimation of the second order derivative in other methods). Moreover, these methods provide robust algorithms when finite-difference approximations to the derivatives are required.

In our problem the magnitude of the errors coming from the approximation of the derivatives, largely dominate the other errors.

Although there are expressions to estimate the size of the finite-difference interval in ideal cases* (see Gill et al p. 341) they have proven to be irrelevant here, since the conditions under which they have been derived are much too restrictive for our problem. Therefore, the selection of the finite difference interval in the problem of real-time control of urban drainage systems must be done on the basis of practical (engineering) criteria.

For the sake of completeness we will argue that these expressions are derived on the basis of the available floating-point accuracy in the hardware platform as well as on setting empirical bounds to the larger sources of error.

An example of such formulas involving a forward difference approximation to the second derivative is:

\[ h_p = 2 \sqrt{\frac{\epsilon_A}{\Phi}}. \]  \hspace{1cm} (3.41)

where:

\( \epsilon_A \): number of significant digits according to the floating point accuracy of the computer (usually 10^{-4} for single precision and 10^{-12} in double precision).

\( \Phi \): An acceptable (non-zero) value of the finite-difference approximation of the derivative.

Thus, the procedure involves primarily the determination of a trial-interval \( h_n \) that may be used in order to compute an order of magnitude for the estimate of \( \Phi \) from the finite-difference formula applied.

* i.e., when the so-called objective function is largely sensitive to small changes in the control variable.
3.8 The scaling problem

Several problems are usually encountered during the process of development of algorithms for constrained multivariable search. As indicated in section 3.2 the characteristics of this kind of problems, namely, high non-linearities of the economic model and constraints*, interrelations among constraints and dependent variables make the solution of each particular problem a very difficult and complex task.

Although, some of these problems have been indicated in the above section, it is possible to state (Gill, et al, p. 324) that the most common problems encountered are:

a) Programming 'bugs'

b) Insufficient decrease in the objective function

c) Poor scaling

d) Overly stringent termination criteria

e) Inaccuracy in the finite-difference approximations.

The solution of a) is obvious. Item b) has been discussed in section 3.7. Item d) is often 'treatable' (see Gill et al, 1981, p. 327) and not very 'critical' to our real-time control problem. In general, item e) can be treated by reducing the length of the finite-difference interval down to the limit discussed in section 3.7.

Therefore, the critical remaining problem is the problem of scaling. This item is very controversial among authorities in this matter, to the extent that some authors even devote entire sections in their texts to this problem, while it is often ignored by others.

It is believed here that the problem of scaling is of central importance to the solution of the constrained problem. The term 'scaling' has often "been used in a vague sense to analyze numerical difficulties whose existence is universally acknowledged, but cannot be accurately described in general terms." (Gill et al, 1981, p. 273)

In general terms, 'scaling' is related to 'accuracy' in the problem formulation. Accuracy is understood here as the combination of many factors including the available digital floating-point representation, the accuracy in the function evaluation, the nature of approximations and so on. In general is described as the precision with which a computed function approximates the exact function value.

* We will subsequently refer to constraints in the most general way, i.e. as the set of physically restrictive conditions which must always be satisfied during the system operation. According to Schilling (1991), this set is divided into static constraints (eg. maximal rates, storage capacities, bounds) and dynamic constraints (eg. continuity and energy balances). Within the set of general constraints there is a particularly significant subset called the active constraints which directly restrict the solution space (defined by the control variables) according to each specific problem formulation.
There are several types of scaling that can be applied in order to improve the performance of an algorithm for constrained multivariable search. All the scaling procedures to be described require the availability of first-order derivatives. Among these are:

- **Scaling of the variables.** The idea of this kind of scaling is to make all the variables involved in the problem of the same order of magnitude in order to assign a similar 'weight' to these during the optimisation process. This may be useful in those cases where the independent variables involved have largely different order of magnitude.

- **Scaling of the derivatives.** This form is somehow related to the above. Sometimes 'bad' scaling occurs when the partial derivatives of a function with respect to a particular variable are not 'balanced'. The reference used is often the accuracy with which the function can be evaluated (εₙ). This kind of scaling has shown to be particularly relevant to our implementation, since the information about gradients is directly included in the formulas for the determination of the new - constrained - 'current point'.

- **Scaling of the finite-difference interval.** In some problems it is useful to restrict the effect of the choice on the finite-difference interval in the evaluation of the objective function. Several ‘formulas’ are available which allow this requirement to be fulfilled. This might be relevant in cases where 'over-sensitive' functions are present.

- **Scaling of the objective function.** The idea of this restriction is to restrict the response of the objective function to 'perturbations' in the control variable. However, some authors seem to believe that this form is not relevant since "...in theory the solution of a given problem is unaffected by any algebraic operation that we perform on \( F(x) \)." (Gill et al, p. 351).

- **Scaling of the constraints.** This kind of scaling has a considerable effect on the computation of the solution of the problem. It affects the limiting accuracy of the solution in the range-space of the active set. A well-scaled set of constraint functions should be well-conditioned with respect to perturbations of the independent variables. On the other hand, the constraint functions should have an 'equal weight' during the optimisation process.

The types of scaling to be applied in each case must be carefully chosen in relation to the characteristics of the problem to be solved. A poor scaling might have a considerable negative influence in the overall performance of an optimisation algorithm. Unfortunately, there is no general 'recipe' for choosing the most convenient way of scaling. Rather, it is almost entirely determined by the characteristics of the problem to be solved and the selected model of representation.
3.9 Closure

An overview of the state-of-the-art in numerical optimisation techniques has been presented in this chapter. The importance of the selection of a proper numerical methodology for the solution of the problem of on-line control of urban drainage systems has also been emphasized. This must be analyzed within the context of the characteristics that make this problem a 'special' optimisation problem.

The methods within numerical optimisation which can be used in order to solve this problem have been discussed. The derivation of the necessary and sufficient conditions for the existence of a extreme in a bounded region have also been performed.

In principle, one of the most general and accurate methods to tackle problems of dynamic nature is of course dynamic programming. However dynamic programming is generally too demanding on computational resources for real-time (see for example Tan, 1995). For the time being, dynamic programming seems to be practically restricted to 'off-line optimisation' and parallel computing. In the present situation, where the real-time control process is performed in on-line mode on a serial computer, discrete-continuous non-linear programming should produce a more compact and faster symbolic code vehicle. In effect, an out-of-date solution is irrelevant for real-time control regardless its accuracy.

The ways in which the phenomenon known as the 'curse of dimensionality' has been so far restricted to allow real-time implementation of this discrete-continuous, fully-non-linear optimisation methodology have then been discussed.

A discussion of the methods for unconstrained multivariable search has been opened. The inclusion of constraints and their consequences have then been analyzed. This was followed by an overview of the most useful methods in this problem (a direct method and an indirect method) for the search of a constrained optimum.

The use of finite-differences approximations to the derivatives is important in problems where exact analytical derivatives are not available. This introduce 'errors' which have to be taken into account in the solution of the optimisation problem. In this situation, it is useful to select an algorithm which provides a 'robust' implementation.

It has been shown that the choice of the finite-difference interval is not a trivial matter in our problem since it has a considerable effect in the overall performance of the process.

Finally, a few words have been devoted to the problem of scaling which is so controversial among authorities in this matter. This problem might be essential to good performance in some solutions and must be treated with care.

Although it has been already discussed, it must be stated again that the solution of each particular problem in this domain is a task involving considerable complexity, due to the factors listed above. The problem of optimisation of the real-time control of urban drainage networks is a highly non-linear problem with input vectors of a stochastic nature.
These techniques as and by themselves are not sufficient to solve the dynamic problem of real-time control of urban drainage systems. Therefore, the next chapter will be devoted to the description of an essentially different - and, we shall claim, complementary - way of approaching this problem: the logical optimisation.
Chapter 4  Logical Optimisation

4.1  Introduction

Reflecting upon the nature of our knowledge gives rise to a number of philosophical problems. These constitute the subject matter of the theory of knowledge or epistemology. Although most of them were already discussed by the ancient Greeks there is no total agreement even now as to how they should be solved.

The 'problem of truth' is one of the basic problems of the theory of knowledge (Chisholm, 1989, p. 4). This problem was already analyzed by Socrates and Plato who used the example of two men, one with 'real knowledge' and the other one with 'true opinion' in relation to 'the many sorts of knowledge'*. He suggested the following approach to the question 'What is the distinction between knowledge and true opinion?'. First, we must assume that if one man 'knows' and the other one has 'true opinion' but does not know. Thus, the first man has everything else that the second man has and also something else. After having made this assumption, he inquired: 'What is that which, when added to true opinion yields knowledge?'. Although this intuitive approach to the problem did not provide all the answers sought, it certainly brought the problem under discussion and spread some light upon the question about the many sorts of knowledge. The addition of 'adequate evidence' to true opinion does not necessarily produces 'knowledge'. There is a concept of probability associated to this. May we say then, that probability is that which, when added to true opinion yields knowledge?.

Probability is usually defined in a variety of senses. Of these, the most common are the statistical sense, the inductive sense and the 'absolute' sense. Whichever of these interpretations we adopt, we find that the concept of probability does not provide us with a satisfactory answer to Plato's question. Let us allow ourselves to say that if a man believes something, then what he believes is a proposition. Therefore, probable in its inductive sense "refers to a certain logical relation that holds between propositions". Unfortunately, however, the question "what is a good inductive argument?" is at least as difficult as the question "what is the distinction between knowledge and true opinion?"

The above would be equivalent to attempt to draw the 'line' between knowledge and true opinion by stating that "the subject s knows the proposition h to be true, provided that h is probable in relation to another proposition e", which leads to the questions "what other proposition e?" and "how does s knows e to be true?" and so on.

Perhaps a more 'successful' attempt at separating knowledge from true opinion can be performed from an ethical point of view.

* The italics are initially used to quote the original words that Plato used in his essay 'Theatetus' and, specially, in the Meno dialogue.
Therefore, 'to know that h is true will be, not only to have true opinion with respect to h, but also to have a certain right or duty with respect to h' (Chisholm, 1989, p. 12). Then, the success of this definition is conditioned by the definition of what is 'right' or what is 'duty', and also to what is to be considered as 'beliefs'. If these concepts are not properly defined, absurd situations may arise.

However, it is known that knowledge is based upon basic 'beliefs'. Without these, it would be essentially not possible to achieve a certain level of insight in most fields of science and technology. For example we believe in modelling in the same way that we believe in symbolic representation. It is a known fact that the success in modelling a given problem is highly influenced by the skills of the modeller, provided we have a 'good' model of reality. We 'know' that, provided the above mentioned conditions hold, the model's forecasts are very likely to be in accordance with the observations we make about an 'observable environment' and therefore we 'believe' in its predictions.

Let us use the words of Leendertzse (1981) - from Abbott (1991) - in order to describe the so-called modelling relation:

"The way a modeller derives a model for the system he is studying can best be described as an intuitive art. No fixed rule is given. The modeller must have the ability to analyze a problem, abstract its essential features, select and modify assumptions that characterise the system and subsequently extend and enrich it until a useful approximation is found ... Moreover ... the modeller is at least as important as the model which is used."

In this sense, some propositions are 'beyond reasonable doubt'. We may also say that it is reasonable for a person to believe them. These include those propositions for which he has 'adequate evidence' (in the sense discussed above). Presumably it is reasonable for somebody to believe any proposition that is more probable than not, in relation to the totality of what he knows. In other words, "induction is justified". The concept of evidence is also relevant to the discussion about the definition of knowledge.

We should at this point recall the definition used in hydroinformatics (Abbott, 1991), that "reality is the name that we give to the interface between our inner and outer world". Thus, 'reality' and any individual 'truth' can only be defined relative to a specific social, cultural and even religious environment.

The next sections will be devoted to defining the basic concepts of knowledge-based control in relation to the real-time operation of urban drainage networks. Due to its relevance to the present study, the analysis of the different classifications of knowledge (sections 4.3 and 4.4) will be done first in terms of diagnostic tasks and then generalized.

### 4.2 The concepts of knowledge-based, real-time control

The term 'knowledge-based control' (also intelligent control) has became extensively used in today's engineering practice. Intelligent control must involve both intelligence and control theory. It must be based upon a serious attempt to understand and replicate what we really
mean by *intelligence*: the generalized, flexible and adaptive capability that we observe in the human brain. Furthermore, it should be firmly rooted in control theory. In relation to our field of interest, our designs *must* often be intuitive in the early stages, but, once these designs are specified, we should at least do our best to understand them and to evaluate them in terms of the deepest possible mathematical theory.

From this point of view knowledge-based control is a vast subject. Formal proofs can be provided (see for example White and Sofge, 1992) that there is an underlying relationship between several topics in intelligent control. One of the implicit sub-objectives of this work is to demonstrate the need for the understanding of this statement.

Intelligent control embraces classical control theory, neural networks, fuzzy logic, Artificial Intelligence (AI) and a variety of search techniques (including for example, genetic algorithms) some of which have been reviewed in section 2.5. Figure 20 illustrates the relation between neural networks and control theory.

Neuro-control is a subset both of neural network research and control theory. It may also be observed that numerical optimisers are totally contained within the field of control theory.

The following definition of intelligent control which is implicit from the above mentioned concepts can be stated (Werbos *et al*, 1992):

"*Intelligent (knowledge-based) control is the use of general-purpose control systems, which learn over time how to optimise in complex, noisy, non-linear environments whose dynamics must ultimately be learned in real-time. This kind of control cannot be achieved by simple, incremental improvements over existing approaches ...""

As pointed out in section 1.4, in relation to the problem of real-time control of urban drainage systems, new concepts were required in order to develop efficient knowledge-based (in the widest sense) control systems. These concepts are now provided under the new perspective offered by hydroinformatics and fifth-generation modelling.

The approach employed in this work (the blend of numerical algorithms and logic) has been recognized (see Åström and McAvoy, 1992, p. 7) as a convenient one when approaching such problems. As a matter of fact, as will be apparent from this and subsequent references, there is a significant ongoing research activity aimed at exploring the potential of this combination.

Before proceeding further, it seems to be useful to provide a conceptual framework (from Åström and McAvoy, 1992) which could be used in order to view the development efforts in the field of knowledge-based control. This framework is schematized in figure 21.
The main underlying idea is that intelligent control can be viewed from the point of view of application development efforts as a multi-dimensional space with at least three axes: rules, objects and algorithms. Classic expert systems are grouped into the dimension of rules. One of the main shortcomings of this 'one-dimensional' approach is that for large problems it may lead to a symbolic code which is essentially unmaintainable, due to the large amount of rules of the kind .. if [condition] .. then [action]. Classical expert systems are also brittle in the sense that they show a sudden performance degradation when there are no available rules to cover the case at hand. Another recognized problem of these applications is the problem of providing 'weak explanations' to conclusions. The cause of this problem is found in the fact that in this case there may be major differences between an 'explanation' which is the path followed to find a decision, and a justification, which is a convincing and rational argument of why a decision has been taken.

The first limitation is often overcome by adding another dimension (objects) resulting in much more compact, maintainable, efficient and object-oriented symbolic code. One way of 'dampening' the second and third limitations is by introducing elements of fuzzy logic in order to generalize the applicability of the rules by incorporating valuable site-specific knowledge and therefore reinforcing their 'reasoning' capability. On the other hand, the applications encountered along in the algorithmic axis (numerical optimisers for example) are not, as and by themselves, flexible enough to tackle the problem of real-time control of urban drainage systems and therefore they need to be reinforced by elements taken from the other two axes. Therefore, it is possible to state that knowledge-based control refers to systems which essentially utilize the complete space shown in figure 21.
In order to complete the above discussion, we must now provide elements for a practical definition of knowledge with application to the problem that we are trying to solve. These definition must fulfil the conditions discussed in section 4.1 within the framework provided by the conceptual basis of knowledge-based control.

Speaking in general terms, we could state that domain-knowledge consists of the set of descriptions, relationships and procedures that hold within a certain universe of discourse, often regarded as the 'domain'. The French philosopher M. Foucault describes knowledge in general (Foucault, 1970), as divided into three main parts: taxonomy, mathematics and genesis. The taxonomy is the knowledge of the ordering and classification of things within a domain. This part of knowledge, which is declarative as it describes a set of facts, will be referred to as factual knowledge (Amdisen, 1992, p. 10). The mathematics is, in this context, the knowledge which allows reasoning in terms of different types of calculi (for example predicate calculus and propositional calculus). This part of the knowledge, which is procedural as it describes a set of general inference procedures, is part of the domain independent reasoning knowledge. Finally, the genesis, which in this context means the creation or acquisition of knowledge (learning), is generally related to empirical knowledge obtained from 'experience'.

Within the group of purely knowledge-based systems* there are several techniques with application to the problem of on-line control of urban drainage networks, some of which have been discussed in section 2.5. Among these are:

i) Heuristic rules
ii) Diagnosis
iii) Fuzzy logic
iv) Neural Networks
v) Learning control systems
vi) Adaptive critic methods
vii) Approximate dynamic programming

Due to the significance they have to the present work, our attention will be given to the techniques included in items i), ii) and iii). However, in view of the rapid development of research in the field of intelligent control it will be necessary in future to take into account the other issues, even though they are out of the scope of the present study.

*1 the phrase 'purely knowledge-based control systems' is used in this context to indicate that stand-alone methods grouped in the classical optimisation are excluded from the present analysis. However, as indicated in section 4.1, they must be included when speaking about knowledge-based approaches, if only because of the conceptual significance they have as foundations to the intelligent control theory.
4.3 The domain-independent reasoning knowledge

As explained through sections 4.1 and 4.2 an effective real-time control system must make use to a certain extent of the three main groups of knowledge defined in this study: the factual (taxonomical) knowledge, the domain-independent (mathematical and in general theoretical) reasoning knowledge and the empirical knowledge (also called practical, strategic or heuristic knowledge). They all seem to be important for the real-time control of urban drainage networks.

For methodological purposes we have chosen to start from the description of the domain independent reasoning knowledge.

According to Johnson and Keravnou (1988) the domain-independent reasoning knowledge can in turn be divided into two groups:

i) **Factual knowledge**, i.e., knowledge about general facts in the domain.

ii) **Reasoning knowledge**, i.e., knowledge about how to use the facts in order to reason about the domain.

In our problem, the domain-independent reasoning knowledge is conveniently described by a set of diagnostic tasks which form a general model of diagnostic reasoning. The three main types of inference, namely abduction, deduction and induction, are comprehensively described by Peirce (1956) and Amdisen (1992, p. 12). Pierce describes the first step of what he calls the 'scientific method' as an abduction task as follows:

".. Accepting the conclusion that an explanation is needed when facts contrary to what should be expected emerge, it follows that the explanation must be such a proposition as would lead to the prediction of the observed facts, either as necessary consequences or at least as very probable under the circumstances. A hypothesis then, has to be adopted which is both likely in itself and renders the facts likely. This step of adopting a hypothesis as being suggested (in accordance with) by the facts is what I call an abduction .."

Therefore the purpose of an abduction is to suggest a hypothesis which can explain the observations. In abduction it is important to infer all possible hypotheses. This process is illustrated (from Amdisen, 1991) in figure 22.

The observed effect A is matched with the effects described by the set of implications. The effect A is implied by the causes C and D, and they are inferred as possible causes of the effect A. Therefore, the causes C and D are in this way suggested as explanatory hypotheses of the effect A. Abductive inference has similarities with the inference system used in backward chaining but unlike this, abduction identifies multiple hypotheses for use in the succeeding inference steps.

After all hypotheses have been generated, the next step is to trace out all its necessary and probable experiential consequences. This step is called *deduction* (Peirce, 1965). Given the rule and the case, the result is inferred by *deduction*, which corresponds to *modus ponens* in
propositional logic (Amdisen, 1992, p. 14). It is observed that deductive inference is implemented in knowledge-based systems as *forward chaining*. Also in deduction it is important to identify all the expected effects when this deduction is based in a set of implications. This process is illustrated in figure 23.

The suggested hypotheses, C and D, imply the effects B and X respectively, which means that if C is the cause then the effect B must be observed and if D is the cause then the effect X must be present. An example is illustrated in figure 23 in which from matching with the actual observations, the effect B is found to be present but not X.

Peirce (1965), distinguishes further between two types of deductive inference, the necessary and the probable. In necessary deduction the cause will always produce the effect. On the other hand, probable deductions are those in which cause and effect are related by a probability. Thus, an effect will be produced from a cause with a specific probability.

When all the suggested hypothesis have been tested, the last step is to accept or reject the suggested hypothesis on the basis of an evaluation of the test results obtained during the deductive step. This step is known as *inductive reasoning*.

This means that, based on a relation between cause and effect, an implication can be inferred if the tests support the general relation. This is illustrated in figure 24.

Thus, given a result and a case, a general rule is inferred by *induction*. When induction is used to distinguish between a number of hypotheses, each implication from cause to effect is regarded as a general rule about the observed effects and the suggested hypothesis is regarded as a description of the actual case.
Some schemes of inductive inference deny that an event will ever occur on the basis that it has never so far occurred. However, probabilistic descriptions on the basis of experimental verification seem to be better suited to our problem. In this case, a diagnostic task is required in order to identify the cause of discrepancies observed between a desirable state (previously identified) and several 'current' states.*

All three kinds of inference have a well-defined purpose in diagnostic reasoning, and they form a general model of diagnosis. The new knowledge is introduced into the process during
the stages of *abduction* (where the hypotheses are suggested) and *induction* (where the hypotheses are concluded or rejected). This two stages process is described by both Peirce (1956) and Ayer (1987) as synthetical inference, i.e., an experience which can only be verified through experience. Deduction, on the contrary, is of the type 'analytical inference', i.e., it is known whether is true or not before it is experienced. The general model of such a process of diagnosis depicted in figure 25.

![Diagram of diagnostic reasoning](c:\phd\diagmod.wpg)

**Figure 25: The general model of diagnostic reasoning**

### 4.4 The domain-specific reasoning knowledge

As stated in section 4.2, the *domain independent reasoning knowledge* describes the *order* in which the different kinds of inference are applied in diagnostic reasoning. This is the reason why this knowledge can be applied to more than one specific domain.

On the other hand, the causal relations between the properties associated with the domain are described by the *factual knowledge*. The order in which the different causal relations are applied in the reasoning, is called the heuristic model (Steels, 1989, p. 5.33).

* i.e., those states corresponding to the application of extreme control strategies in the regulators in the network.
In general terms, it is possible to state that *domain-specific knowledge* is of two types: *factual knowledge*, i.e., knowledge about domain entities and their interrelationships and *reasoning knowledge*, i.e. knowledge of how to use factual knowledge to generate knowledgeable behaviour. Factual knowledge is essentially domain specific whilst a fraction of the reasoning knowledge can be applied to more than one domain.

Often, when dealing with ill-structured problems, use has to be made of domain-specific factual knowledge. *Classic* (brittle) *heuristic rules* are in general not suitable for representing this kind of practical, or strategic knowledge (Johnson and Keravnou, 1988, p. 186), because of their rigid semantic structure.

The purpose of the different diagnostic tasks is defined by the kind of inference it performs in relation to a particular phase of the diagnostic reasoning, e.g., abduction is generally related to structural aspects of the network (e.g., the location of a particular sensor). However, the dependencies between the tasks is not based on the kind of inference they perform, but must be established during the reasoning. This requires a complicated process called focusing which, alongside the selection of the right hypothesis, must also identify the next diagnostic task to be performed.

Here, the domain specific *reasoning knowledge* is regarded as procedural, and the dependencies between the diagnostic tasks are described in relation to the different kinds of inference used and the causal relations applied. Thus, the description of the heuristics of each of the tasks become explicit in relation to the domain independent reasoning knowledge.

In relation to the problem of real-time control of urban drainage systems, the domain specific reasoning knowledge seems to be far more relevant to the system’s operation than the domain independent reasoning knowledge. This domain specific reasoning knowledge is encoded in rules and objects containing information for a proper on-line operation of the system. The rules must necessarily contain fuzzy terms in order to transfer the perception of the information from the site-specific context to a more general description.

### 4.5 The factual knowledge

The purpose of this section is to establish a short discussion not about the conceptual aspects of the factual knowledge (they have been briefly exposed in sections 4.3 and 4.4) but about the ways in which the [domain-dependent] *factual knowledge* can be expressed both in terms of *site-specific knowledge* and in terms of *knowledge based on general properties* about the domain.

This has a considerable importance to knowledge-based control since the coding of knowledge based on general properties about the domain, together with the introduction of objects and elements of fuzzy logic in the rules, appears as one of the most effective ways to escape the *brittleness* inherent in traditional expert systems.
We will try to demonstrate here that an efficient implementation of knowledge-based control is generally related to the proper use of the above mentioned features.

The figure 26 illustrates the difference between a piece of factual knowledge based on site-specific knowledge (A) and the same piece of knowledge based on general properties about the domain (B).

A) The regulator A3 -> A1 must be open if the flowrate arriving to the branch OSTSTR to A3 exceeds 5.5 m³/s and there are less than 2.0 x 10³ m³ of spare capacity in the branch.

B) The commanding regulator of the branch must be open if the differential inflowing volumes in the branch are increasing too quickly and the present volume in the branch is close to the available storage in the branch and there is enough spare biological treatment capacity at the treatment plant.

Although the corresponding elements of knowledge (from previous experience) in operating these systems have been encoded in both cases, the second piece of code contains knowledge of a general range of applicability within the domain of urban drainage networks. It should be observed that the second piece of knowledge describes in general terms a control action to be taken on the basis of the on-line development of a flow event in the system.

In the case A the knowledge encoded is only valid for that specific network because it implicitly embodies network information, such as the location of regulator A3 in the branch, the storage capacity in the branch, the acceptable free capacity and so on.

It is to be expected that a system using this type of rules will be more easily subjected to degradation.

On the contrary, in the case B this process has to be carried out explicitly, i.e., the matrices containing information about the system's connectivity and reachability must be constructed beforehand for every site-specific problem (see Amdisen, 1992, p. 58). Elements of directed graph models are used for the construction of such matrices. They reflect the structural relation between the network's components. This approach is illustrated in figure 27.

Another example of knowledge based on general properties about the domain with, application to our problem and slightly more diagnostically-oriented, is illustrated in figure 28.

This introduces a diagnostic task since the control action is derived on the basis of reasoning about the causes of recorded discrepancies between 'observed' and 'desirable' states. From a practical point of view, this is equivalent to correcting the 'mistakes' observed in the
Figure 27: Representation model based on digraph theory (after Amdisen, 1991)

The 'target' regulator must be closed if the observed discrepancy in the system's state variable at each of its associated sensors can be regarded as 'large positive' or 'large negative' in relation to its corresponding threshold value.

*1 This is, the discrepancy between the current state of the system corresponding to the application of an extreme control strategy and a desired state of the system as required to satisfy the objectives imposed by the system's analyst.

Figure 28: Another example of coding knowledge based on general properties of the domain

application of the so-called extreme control strategies. In this sense, it is possible to identify an on-line 'learning' process of limited scope, which occurs within the intelligent agent in this implementation of the combined logical-numerical framework.
4.6 Object oriented representation

As stated in section 4.2, object orientation is an essential feature sought in today’s knowledge representations. The object-oriented encoded knowledge is usually maintainable, compact, general and robust, unlike the knowledge encoded in classic expert systems which is in this sense brittle (that is, it is exposed to sudden degradation when there is no knowledge immediately available to cover the case at hand).

The later versions of the main logical programming languages support the feature of object orientation. This feature has also been developed by most of the logical shells available to simplify application development efforts, resulting in much more efficient codes.

In this sense, NEXPERT OBJECT is not an exception. On the contrary, the later versions of this shell (SmartElements 2.0) not only provide a hybrid development tool but also a powerful and flexible graphical user interface (GUI). A hybrid tool integrates rules and objects in a proper fashion providing important commands to handle the resulting 'hybrid space'. This space is illustrated in figure 29.

As a matter of fact, objects themselves, are a powerful representational structure. They constitute the basic 'bricks' to represent entities in the domain. Objects have properties which describe their characteristics and slots which store information about specific objects.
Objects of similar characteristics are grouped into classes. There are also metaslots which describe how the slots behave. Properties can be inherited from a class or object to another class or object. Inheritance allows efficiency, as the particular attribute only needs to be declared in one place. It also provides consistency, as everything which inherits an attribute behaves in the same way, and it also provides generality.

One of the most powerful features to achieve generality is provided by dynamic objects. These dynamic objects allow us to model a world whose exact structure is not known beforehand (at the moment of coding).

They can for example, to be retrieved from databases containing all the site-specific information (e.g., structural relations in the network).

It is also possible to create dynamic links between objects and classes or objects and other objects in order to reflect changing relationships during the knowledge processing. Dynamic objects, are denoted by a plus (+) sign in front of the object’s name; see figure 30.

In this way it is possible to 'hardcode' only the main structures containing the knowledge based on general properties of the domain. The knowledge based on site-specific properties is either retrieved from user configured databases or learned through fully dynamic simulations. By these means, the user-defined information is complemented by the intelligent agent.

4.7 The rule-based approach

Experienced human operators have long been able to 'control' processes which could not be totally controlled automatically. This is often due to the fact that the behaviour of the process is not sufficiently understood to extract the knowledge which would allow a satisfactory automatic operation.

Many elements are responsible for the above mentioned features. Among these, is possible to identify the often unconscious feature of the knowledge possessed by experts, the natural reluctance that human experts have to being 'replaced' by 'machines' as well as the lack of optimal representational structures, and so on.

Information technology has helped turn information and knowledge into a valuable and recognised commodity. The use, transformation and enhancement of knowledge is now a prevalent feature of computer-based systems.

Knowledge bases have developed from roughly two main approaches (Schlumberger, 1994, p. 13). These are:

i) Expert Systems
ii) Knowledge-Based Applications (the later approach)
Figure 30: A fragment of the object network of the intelligent agent HYDNET.kb
Expert Systems take advantage of the possibility of formalising domain knowledge in a computer usable format. As discussed above, classic expert systems suffer from brittleness, i.e., a sudden degradation when there are no rules available to cover a specific case.

However, those systems which started with Macsyma (dealing with formal calculus) and Dendral (for chromatographic recognition) were remarkably successful in those domains where the knowledge was essentially well-structured (such as diagnostics or classification). The machine formalism used in classic expert system were heuristic rules encoding highly-structured expertise.

The development of such systems have given rise to the need for a new breed of consultants, knowledge engineers, in charge of the knowledge elicitation process. These knowledge engineers are, in principle, able to transform raw expertise into the required highly-structured machine formalism. Their intercession is necessary because the transformation of expertise into heuristic rules is not a trivial matter. Because of the initial success of Expert Systems, knowledge-based applications arose when the issue of the structuring of knowledge became significant. In effect, their rise was conditioned by the need to solve 'noisy' problems, in which knowledge is often ill-structured*.

As suggested in section 4.6, knowledge-based systems provide a representational structure which is roughly divided into two layers: one of objects and the other of rules which can be roughly be associated to the classical 'know-how' of experts. The rules together with the dynamic links among dynamic objects describe operations on those objects.

Since the structure of objects has been reviewed in section 4.6, we will concentrate now on the rules as (sign) vehicles for the main knowledge representation formalism. The general structure of the rules in a knowledge base is illustrated in figure 31.

4.7.1 Learning rulebases

An important and even perhaps defining attribute of an intelligent control system is its ability of improve its performance with time. The concept of learning is usually used to describe the process by which this is achieved. It has been stated that ".. a control system can be viewed as a mapping, from .. outputs and control objectives to actuation commands .. with learning as being the process of modifying this mapping to improve .. the system’s performance .." (Baker et al, 1992, p. 35).

A commonly held notion is that learning results in "an association between input stimuli and desired output actions". (Baker and Farrell, 1992, p. 46). A learning process is said to be complete if the knowledge is stored and increases with time for its use in subsequent events.

* As in the problem of real-time control of urban drainage networks. This problem can be characterised as highly non-linear, very dynamic, and noisy.
If \( \text{condition 1} \) and \( \text{condition 2} \) and \( \text{condition 1} \)  
\[ \Rightarrow \quad \text{Hypothesis} \]
Then \( \text{action 1} \) and \( \text{action 2} \) and \( \text{action 1} \)  
Else \( \text{action 1} \) and \( \text{action 2} \) and \( \text{action 1} \)

\[ c:\phd\rulebase.wpg \]

Figure 31: Typical structure of a heuristic rule

Artificial Neural Networks (ANN) provide a good example of the above statement. In effect, as the training set for the ANN increases, a better response of the application to real-life situations must be expected. These applications show what is called a graceful degradation with situations which are not included in the range covered by the ANN training set.

The basic idea underlying on-line learning arises from the observation that learning is facilitated in situations where a clear association can be made between a subset of the adjustable elements of the learning system and a localized region of the input-space. As a matter of fact, several desired features for such learning systems seem to rely on incremental gradient learning algorithms. Learning can be viewed as a mapping process with outputs \( M_i \). Then, it is possible to construct 'sensitivity functions' of the process in relation to the adjustable parameters \( p_j \) in order to define the desirable features of such a process (Baker and Farrell, 1992, p. 53). These functions are defined as:

\[ \zeta_{ij} = \frac{\partial M_i}{\partial p_j} \]  \hspace{1cm} \text{(4.1)}

Whereby, at each point \( x \) in the mapping domain the following desirable features must hold:

\[ i) \] For each output \( M_i \), there exists at least one adjustable parameter \( p_j \) such that the sensitivity function \( \zeta_{ij} \) is relatively large in the vicinity of \( x \).

\[ ii) \] For all \( M_i \) and \( p_j \), if the function \( \zeta_{ij} \) is relatively large in the vicinity of \( x \), then it must be relatively small elsewhere.
The second property (called *localization*) holds on the basis of the first property (called *coverage*). If both of them are satisfied then localized learning is achieved through the input domain of the mapping process. Thus, experience and therefore learning in one part of the input domain have only a marginal effect on the knowledge that has been gathered in other areas. Thus, problems due to the effect of conflicting objectives on the adjustable parameters are also reduced. In this class of problems, the operator $\mathcal{P}$ which makes possible the mapping process is provided by the hydrodynamic simulations.

There are situations, however, in which due to the presence of all kinds of constraints - including of course the real-time constraint -, it is not feasible to design a complete learning process in the sense explained above. In these cases, the learning process, although effective, must have a scope which is limited to the duration of the event and therefore has to be reset every time an event is controlled in real-time. This is called an 'on-line learning process'. This approach which seems to be quite advantageous for effective on-line control, is described in the work on intelligent control of Werbos et al., discussed in section 4.2.

Several techniques can be used in general (Khonkder, 1995; Wilson, 1995) in order to perform the so-called learning process for the control of urban drainage networks.

The main idea is that a certain kind of internal 'relationship' is established by mapping from system's state to cost predictions. This relationship is constantly evaluated according to a predefined rewarding criteria. Then, after the learning process is completed the so-called internal relationship can be used to recognize 'good' from 'bad' scenarios.

### 4.8 Fuzzy logic

"When fuzzy logic was conceived, it was expected that most of its applications would be in the realm of those knowledge-based systems in which the resident information is both imprecise and uncertain. Contrary to this expectation, most of the successful applications of fuzzy logic at this juncture relate to control and systems analysis in which there is imprecision but no uncertainty." Lofti A. Zadeh (1992).

When applied to control, fuzzy logic is known as fuzzy linguistic control or FLC (Langari et al, 1992, p. 93). The FLC technique can in principle be used when a sufficiently accurate and yet not unreasonably complex model of the system to be controlled is unavailable or when a precise measure of performance is not very meaningful or practical. In these cases the control problem, instead of being posed within a strictly analytical framework, is essentially based on empirically acquired knowledge regarding the operation of the process.

In fuzzy linguistic control systems, the control scheme is implemented in two parts: a dynamic filter (the intelligent data processor) and the rule processor which constitutes the core of such a system. The rule processor draws on a rulebase essentially encoding empirical knowledge about the operation of the system. This knowledge is stated in terms of the components of the process input and output vectors.
The inference engine computes (according to a certain defined operational strategy) the appropriate control action according to condition → action rules (see figure 4.12, section 4.7).

As explained in section 4.5, one of the main features of these rules is that they embody *linguistic terms* or labels that are used in order to transfer the perception of the information from the site-specific context to a more general description.

In fuzzy logic, elements in a given 'set' receive overlapping coverage from the elements of the neighbouring sets. This is a natural and desirable property of fuzzy sets since these elements are intended to be partial members of sets which do not have the sharp, well-defined boundaries normally associated, for example, with numerical techniques.

Following the above notion, one could develop a unified representation for a given term set containing labels such as *high, low and medium*. This is illustrated in figure 32.

![Figure 32: Assignment of memberships of the fuzzy set](c:\phd\fuzzy_1.wpg)

It is apparent from the above discussion that the idea of 'linguistic' fuzzy rules replacing mathematical formulae or some other analytical exact description provides an effective representational tool to tackle problems where the domain-knowledge is not very well structured.

This is the reason why these techniques are in principle so attractive to designers of real-time control systems for urban drainage networks. It must be pointed out, however, that fuzzy logic,
much like classical techniques is based on a rigorous mathematical foundation (see, for example, Zadeh, 1965, p. 339), and in particular that of set theory. In other words, in fuzzy control the rule set constitutes a symbolic representation of the control algorithm, while control computation takes place at the level of the underlying algebraic formulations and is done in a precise and formally tractable manner.

We will explicitly claim now that symbolic representation offers a number of advantages over non-symbolic conventional approaches to less-well structured problems as the one which occupy us here. In order to design a fuzzy linguistic control system we need (using the rationale of fuzzy control; see Langari et al, 1992, p. 101):

i) Prior and usually direct knowledge about the operation of the process.

ii) The ability to articulate this knowledge in a linguistic form.

iii) The means to represent this knowledge in a meaningful and yet quantitatively sound manner.

The outcome of such a design process - which is a knowledge engineering problem -, is a control algorithm that ideally mimics the control function of an expert human operator of the given process.

In even less-well structured situations, one is generally unable to translate the external functional specifications of performance into a set of viable control design objectives as described above. The difficult stems from the fact that in these situations such specifications of performance are stated at a level which is too far removed from the level of operation of the process.

This situation, coupled with the difficulties (and associated inaccuracies) in modelling the physics of the process itself, creates uncertainties in the very structure of knowledge. Therefore these problems must be treated "as they appear", i.e., by making use of all sources of information and knowledge as this arises from the actual (ongoing) situation.

The above situation gives rise to the need of rules containing imprecise (fuzzy) terms as illustrated in figure 4.9 (section 4.5). These fuzzy terms will be evaluated in relation to the development of the on-line situation.

Let us now devote a few words to practicalities regarding design methodologies for fuzzy control since this is a point of central importance in the system's development. Following the above discussion, we will state that, in the design of a fuzzy controller, one must identify the main control variables and determine a linguistic set which provides the right level of accuracy (resolution) for describing the action.

For example, a linguistic term set containing three elements, say \{small, medium, large\} may not be satisfactory within a certain domain, which may instead require the use of a five-term set such as \{very small, small, medium, large and very large\}. Moreover, different types of fuzzy membership functions can be used. The selection of the fuzzy membership function affects the type of reasoning to be effected by the inference mechanism (Langari R. et al, 1992, p. 105). The most commonly used types of fuzzy membership functions are illustrated
in figure 33. A complete and motivating discussion about the representational mechanisms in fuzzy control can be found in Langari and Berenji (1992, pp. 93 - 133).

Figure 33: Common fuzzy membership functions

Membership functions

a).- Linear monotonic
b).- Triangular
c).- Trapezoidal
d).- Bell shaped

c:\phd\fuzzy_2
In relation to the problem of real-time control of urban drainage networks, the selection of the methodology for the design of the fuzzy controller may have an overall influence in the effectiveness of the reasoning process. On the other hand, if the set of linguistic terms is under or overdesigned, the fuzzy subsets may not be properly represented through the universe of discourse. From a conceptual standpoint, a wrong fuzzy partitioning limits the capability of the inference engine to interpolate the membership across the boundary of application.

It is useful to remember now that a fuzzy subset is distinguished from a classical or crisp set by its membership or characteristic function which can take any value in the unitary or definition interval \([0, 1]\).

Unfortunately, there is no generalized 'recipe' for the selection of the fuzzy membership function or the size of the fuzzy linguistic sets. They are rather defined by the characteristics of the different kinds of domain knowledge.

4.9 Closure

The need for representing less-well structured knowledge arises in various analyses such as model calibration, real-time control, etc. In the case of real-time control this problem is often linked to a comparison between 'observed' and 'desired' (or even expected) situations. In a general description of these problems, the result of such a comparison is not an exact numerical value but rather a linguistic description such as "small" or "large" or "critical", and so on.

The fuzzy theory provides elements for a general description of less-well structured problems by providing methods for the case specific interpretation of these 'general' linguistic labels. These methods, which are deeply rooted in the most rigorous mathematical theory, have a high degree of generality and flexibility.

The results of the application of such methods seem to be greatly influenced by the selection of both: the kind of fuzzy membership function and the size of the fuzzy linguistic set. This last factor is conceptually related to the effectiveness of the representation in the universe of discourse.

For hydrodynamic systems, the results of the application of the fuzzy set theory on the rules encoding knowledge based on general properties for the domain (the general class which groups a set of such systems) can then be based on general cause/effect relationships in the domain which are accurately described by a diagnostic task (see sections 4.1 to 4.4). This framework allows for an effective representation of the different kinds of domain knowledge required for the real-time operation of hydrodynamic networks.

On the other hand, the dynamics and complexity of these systems give rise to the need for performing on-line learning as one of the most effective ways to collect valuable domain knowledge based on site-specific properties of the given network. These properties vary from network to network and therefore are difficult to encode. Complete learning systems are those which improve their performance with time on the basis of their previous experiences.
However, when performing real-time control of urban drainage systems it is often necessary to restrict the extension of the learning process to the time span of the event horizon.

The on-line learning process is then interpreted as a powerful complement to the representational mechanisms provided by fuzzy set theory. It is clear now that intelligent (which in this case is a synonym of effective) control is the proper means to superpose all of these approaches.

The next chapter will be devoted to describing the realisation of such a prototype: a hybrid system which for the first time combines all of these approaches under the new perspective offered by hydroinformatics.

But at the same time we state explicitly that this is not the end (but probably just the beginning) of the efforts aimed at effective control. We view this work as one more step in the ascendent process leading to this goal, perhaps not so far away now: the intelligent control of hydrodynamic networks.
Chapter 5  The Combined Logical-Numerical Framework

5.1  Introduction

It is not unusual in research to arrive in unfamiliar and unexplored places. Although the greatest moments in research are unquestionably the moments of discovery, there is something very special about the research activity in itself. This is in essence a learning process in which the relevant - advantageous - features of the problem must be classified, analyzed and only then properly employed in order to proceed to solve the problem at hand. As every learning process, research is fascinating. However, research must be orientated in one way or the other to problem solving. We normally experience little motivation in researching for the sake of researching without a definite goal.

In research activity ".. everybody proceeds along his own path and carrying his own baggage .." (Abbott, 1991). At first, some concepts are interiorised intuitively and then generalized, expanded and reformulated again. There are difficult moments where the efforts seem to lead nowhere, but then, again, reasons to proceed are often found in the discovery yet to come, in the learning that it embodies, in the benefits of the new technology for human society, in the challenge in itself, and, in general, in the wish which characterises many persons to build a better future. Last but not least, these reasons are often found in the support of valuable people around us.

Every computer code is ".. a mapping of some part of the mind, including the state of mind, of its nominal creator .." (Abbott, 1994, p. 5). There must be an intention too: the intention in creating the code in order to provide an effective solution to the problem studied. An object comes to presence. But we must not be concerned about the object in itself. Indeed, ".. we must be concerned much more with what the object does and much less with what it is .." (Abbott, 1994, p. 6). The orientation of the object is what really counts.

Just as in every mapping process, in developing computer code it is necessary to follow a strategy. The output of this mapping process contains an underlying set of influence functions (Baker and Farrell, 1992, p. 55) which contain information about the way in which the mapping process is done. Although this underlying 'set' is in general not visible from outside, it considerably influences the performance of the process. That is why it is so important that this, otherwise invisible 'set' of operations is described somewhere.

Therefore, a part of this chapter is devoted to the description of such a 'set'. It is written with the intention of making it as comprehensible as possible in terms of the so-called 'discourse of average intelligibility' as introduced originally by Heidegger (Being and Time, 1927). The object created which in this case is certainly oriented to problem solving has been called HYDNET_RTC.
5.2 HYDNET_RTC. Overall description

As explained above, HYDNET_RTC is the name which has been given to the combined logical-numerical framework. This framework consists of the superposition of an intelligent agent (declarative and procedural) upon a primarily algorithmic, fully non-linear, numerical optimisation methodology.

Among other relevant functions, this intelligent agent has a guiding function, selecting only the most promising scenarios for numerical evaluation using MOUSE hydrodynamic models.

An overview of the structure and operational features of the combined logical-numerical framework is presented in figure 34.

Figure 34: An overview of the structure and operational features of the combined logical-numerical framework

Figure 35 offers a more detailed illustration of the modular structure of the combined logical-numerical framework as well as the interrelation among modules.

The importance of the guiding function performed by the intelligent agent within the combined logical-numerical framework must be outlined. This interaction, which has proven to
Figure 35: Modules integrating the combined logical-numerical framework
be essential for real-time implementation, often ensures the convergence to a 'good enough' point in the process of non-linear optimisation. As indicated, the intelligent agent derives valuable factual knowledge based on site-specific features of the network by performing an on-line learning process with the help of the fully dynamic models of MOUSE. Within this framework, the optimisation process is performed in successive stages of increasing level of accuracy. The number of stages triggered, depends on the time available to perform the optimisation process. This process starts with a very fast, on-line, logical optimisation step (usually require few minutes). The arrangement of factors is important here because a fairly good control strategy must be ready well in advance in relation to the critical moments of the system loading from precipitation. Since it is not feasible in practice to design a regulation system which controls 100% percent of the flows in an urban drainage network, we might speak of quasi-optimisation. In this sense, we are not concerned with any kind of 'global optimum' of the kind found in mathematical texts, but rather with achieving an effective and intelligent enhancement of the system's performance. As suggested in section 2.7, the combined logical-numerical optimisation process shows itself to be asymmetrical. What this means in practice is that we do not obtain the same results if we modify the order of the 'elements': the arrangement of a first numerical optimisation step followed by a logical optimisation step appears to be inefficient from the [present] point of view of real-time control. In effect, the numerical optimiser would most likely start its search from a 'poor' initial vector and it would initially employ precious time to derive a better strategy. In this hypothetical case we would risk arriving at the critical moments of the event without a reasonably good control strategy to apply to the system. Such a circumstance would most likely reveal undesirable consequences for the on-line control process.

Let us now however analyze the opposite case. By simply modifying the arrangement of factors, the disadvantages of the first scheme become advantages for the second. In this case we initially perform a very fast logical optimisation step and therefore provide a 'good enough' control strategy well in advance of the critical moments of the rainfall event. In this situation the numerical optimiser can effectively show all of its potential: it can perform a considerable fine-tuning of a 'fairly good' control strategy taking advantage of the progressive reduction of the forecasted horizon as it approaches the final stages of the rain event.

It must be explicitly stated that the situation described above corresponds only to the present computational state-of-the-art (1995) in real time control of urban drainage networks. This picture may be expected to change as we introduce more computer power into it. It radically changes with the insertion of parallel processing capabilities in the picture. In such situations the arrangement of the factors could very well be changed. At such a stage of development the natural non-commutativity (asymmetry) of the process can be overruled by hardware factors. In other words, time - and hardware development - are factors which will considerably strengthen the present implementation. However, this does not mean that this is a 'better' approach, whatever that may mean. The constant development and improvement of existing applications is a general law of technological progress.
5.3 The process of dynamic mapping of the non-linear economic model in HYDNET_RTC

As explained in section 2.9, the power of non-linear optimisation arises from the fact that these algorithms can in principle handle cost functionals of 'any shape'. If we stick to certain basic rules (which are not very restrictive in our problem), we can construct a set of cost functions which represent in a rather accurate way the complex process which takes place in these systems. In addition, we can in principle define different cost functions for each of the keypoints in the network in order to attain multiple objectives. This is known as multi-objective optimisation.

From a computational point of view this means that it is not feasible to utilise a set of equations - even complex equations would be much too restrictive regarding multiple objectives - in order to define the so-called economic model for the process. Therefore, in the most general way, the economic model must be obtained through a process of mapping.

Although there seem to be certain common features, the characteristics of the mapping process vary from implementation to implementation. In this framework, the economic model for the process must be obtained through dynamic mapping. The word dynamic has been included to call attention on the fact that this mapping process has to be performed every time a new control strategy is being applied during the optimisation process. The mapping process in this framework effects a contraction in the representation hyperspace initially defined by the set of system state variables at each of the m costpoints to the solution hyperspace composed by the set of control variables at each of the n flow regulators. In general m >= n.

This contraction - also called here a hyperspace conversion process - is obtained through the operator \( \mathcal{P} \) provided by the hydrodynamic simulations. The mapping process can then be represented as:

\[
\mathcal{R} \times \mathcal{P} - \Omega
\]  

(5.1)

where:

\( \mathcal{R} \) : Representation hyperspace of m dependent variables (axis).
\( \mathcal{P} \) : Mapping operator provided by the hydrodynamic simulations.
\( \Omega \) : Solution hyperspace of n independent variables (axis).

The process of dynamic mapping which takes place in the combined logical-numerical framework is schematized in figure 36.

In this framework the m components describing the rewarding functions as function of each of the system state variables are filtered through the time series of their corresponding systems state variable to produce temporary profit time series associated to a given control strategy. These profit time series are then mapped through the [hyper]space of the input domain to project the economic model on the solution space of the problem. This is the hyperspace where the search process takes place. Let us employ a simple example in order to illustrate
Figure 36: The hyperspace conversion process which occurs in the combined logical-numerical framework

the way in which the last step of this process can be computationally implemented:
void Integrate (Xf, Yf, Np)
{
    float Res, dif_height, Arect, Atr;
    int i;
    for (i = 0; i < Np-1; i++) {
        Res = Xf[i+1] - Xf[i]; /* adjust, in time series with timestep & saverate */
        if (Yf[i] <= Yf[i+1]) {
            slice_curve (Xf, Yf, 'up');
            dif_height = Yf[i+1] - Yf[i];
            Arect = Yf[i] * Res;
        } /* end of if */
        else {
            slice_curve (Xf, Yf, 'down');
            dif_height = Yf[i] - Yf[i+1];
            Arect = Yf[i+1] * Res;
        } /* end of else */
        Atr = (Res * dif) / 2;
        Atot = Atot + (Arect + Atr); /* accumulate */
    } /* end of for i */
    Compile_Costs (Atot);
} /* End of Integrate */

5.4 Features oriented to the real-time implementation in the fully non-linear numerical optimiser

So far we have described the attempts at controlling the computational load which characterises the fully non-linear techniques from outside the numerical optimiser. These are basically related to the use of several levels of knowledge about the domain in order to reduce the search space and in order to steer the numerical optimisation process.
But as a matter of fact certain features have been incorporated within the numerical optimiser in order to minimise this problem. The problem of the curse of dimensionality must be attacked from all possible sides. Only then can a feasible real-time solution be achieved.

In order to fully understand the role these features play in the numerical optimisation methodology, we refer the reader to the description of the modus operandi of these methods offered in Chapter 3. Indeed, the features which are now going to be described, allow a significant reduction in the computational time employed in performing the numerical optimisation step.

These features which are built into the numerical optimiser in order to fulfil the real-time constraint are:

A) The 'free-start'.
B) The 'sequential search link'.
C) The 'dynamic constraint' function
D) The fully user-controlled line search
E) The 'slope hotstart'

The feature A) and E) are totally concerned with the slope evaluation process. Features B) and D) are aimed at controlling the single-line search process. However, the feature C) affects both the slope evaluation and the single-line search process. We will introduce each of these features and their role in the numerical optimisation framework. The names given to these newly developed features are original to this work; they do not occur in the available references.

A) The 'free-start'

During the slope evaluation process, it is often necessary to 'measure' the gradient when taking a step of finite length from the current point. This gradient will then be used to construct the finite-difference approximations to the derivatives (section 3.6).

In the problem of real-time control of urban drainage networks, regardless of the control variable we choose, it is usually necessary to perform this 'measurement' twice: the first time in the direction of the positive sense of the perturbation (p+), and the second time in the negative sense (p-). This is illustrated in figure 37.

This process must be repeated for each of the $nc \times ns$ components of the solution matrix, where $nc$ is the number of independent variables (number of regulators) and $ns$ is the number of stages used for the discretisation of the time-dependent 'policy vectors'. Therefore $2nc \times ns$ function evaluations (hydrodynamic simulations) are usually required during the slope evaluation process.

It is common to find in smooth optimisation that the gradient measured in the direction of p+ is nearly the 'mirror image' of the gradient measured in the direction of p- (for a perturbation step of constant length), i.e., they are nearly of the same magnitude but with opposite signs. The only significant exception to this 'rule' is found in the sharp ridges which are fortunately
uncommon in so-called smooth optimisation (section 3.2).

One of the most remarkable characteristics of the Quasi-Newton methods is that they are capable of maintaining the descent property as well as the their superlinear rate of convergence in the presence of inexact line searches, i.e., in the case that the line search is not performed with exact parameters. Even in this case, the matrix update system can maintain the self-orientation property.

The above observation leads to the notion of constructing a relatively simple function by which it is possible to reduce by nc x ns the number of function evaluations during the slope evaluation process. The main idea can be summarized as follows: If when taking a finite perturbation step in the one direction, say $p^+$, the result of the 'experiment' is not successful, then the algorithm must not perform the 'measurement' in the other direction, but it must assign its 'mirror image' to the matrix cell containing the information about the gradient corresponding to the current element in the direction $p^-$. On the contrary, if the experiment is successful, this value will be assigned directly. This assignment must be done in a careful way, according to the current position.

In those cases in which the initial assumption of a steepest descent direction is rather inaccurate (eg, where the slope evaluation process is started from a stationary point), the task of restoring the right vectors is left to the line search procedure.

We will subsequently argue that the 'free start' feature contributes to reducing the risk of being 'trapped' in a local optimum since it provides the capability of exploring directions from the current point, which may be - initially - not those of steepest descent, but then they may lead to better vectors.

B) The sequential search link

The notion behind this feature is again simple. In general, sequential search plans require placing two experiments at the time during the sectioning phase (section 3.4). This makes possible the choice between intervals. In this way the interval which does not contains the optimum is systematically discarded.

In principle the location of each experiment involves one function evaluation (hydrodynamic simulation), so that in order to perform one sectioning step two function evaluations are required. This can be easily reduced by storing the previous successful experiment (pse). This means that, in each sectioning step, only one experiment is needed if the pse is used as a reference.
The reduction achieved in the number of function evaluations by using this system is of order $cy \times ne$, where $cy$ is the number of main updating cycles and $ne$ is the number of Fibonacci experiments needed to locate the extremum during the single-line search. This feature is only possible in this kind of implementation because the number of experiments is totally user-specified.

**C) The dynamic constraint function**

In most real-life cases, a fully dynamic pipeflow model is required in order to model the complex phenomena which characterise the flow in hydrodynamic networks. It has been observed that approximately 95% of the time employed by the optimisation process is spent in performing the fully dynamic simulations.

In this situation it is of central importance to reduce as much as possible the number of hydrodynamic simulations that need to be triggered. This is part of what we call the 'steering' of the numerical optimisation process. Under these circumstances it is logical to attempt to avoid simulations which are not physically relevant, that is, those simulations that do not affect the physical state of the system. These simulations arise as a natural part of the process of search for the optimum.

In effect, one could think of situations in which the regulators move without interfering with the flow: for example, weirs moving above the free surface of the flow. We could also think of an attempt to set a regulator out of its practical range of operation. In these cases it is not necessary to simulate. This could economise time, which is a precious resource in real-time applications.

The dynamic constraint function must observe the time series of the system state variable at each of the flow regulators and compare these values to the intended setpoints. It must be observed that this gives rise to the need to specify to the pipeflow model those points where a given system state variable must be 'monitored' in the search for physical constraints.

Computationally speaking this function could (for example) be realised in the following way:

**D) The fully user-controlled line search**

The only possibility of achieving a 'total control' of the time spent by the numerical optimiser is by controlling the exact number of function evaluations to be performed during the process. An important number of these simulations are performed during the single-line search process.

Therefore an effective numerical search methodology for optimisation must provide a full user control of the number of experiments made during the line search. This is related to the design of the software. In effect, the algorithms must be consciously developed to achieve this feature.

These features have been implemented in the present logical-numerical framework. We have the possibility of knowing in advance the exact number of function evaluations which is going to be performed during the whole process. This number can be computed according to the
The Combined Logical-Numerical Framework

float Phys_Constraint (Keypl, ..., KeyPn)
char Keypl[STDL], ..., KeyPn[STDL];
{
    char filename[STDL];
    unsigned int i, j;

    Run_Pipeflow_Model (KeyP1, KeyP2, ..., KeyPn);
    Write_pipe_file (filename);
    Set_Flags_On_Regulator_TSSeries (*FlagToKP1, ..., FlagToKPn);

    for (i=0; i<nreg; i++)
        for (j=0; j<ntimes; j++)
            if (Setp[i][j] > Max_SSV[i][ks])
                Setp[i][j]=Max_SSV[i][ks];
/* end of for of j */
/* end of for of i */
/* end of Phys_Constraint */

following expressions:

In those cases where quadratic termination is possible and desired:

\[ N_e = 1 + n_i \left( n_j \times n_s \right) + n_f + 2 \] \hspace{1cm} (5.2)

In those cases where quadratic termination is not possible (or not desired):

\[ N_e = 1 + n_m \left( n_j \times n_s \right) + n_f + 2 \] \hspace{1cm} (5.3)

where:

- \( N_e \) : number of function evaluations
- \( n_i \) : number of independent variables in the problem
- \( n_s \) : number of stages used for the discretisation of the time-dependent vectors
- \( n_f \) : number of experiments to be performed during the line search
- \( n_m \) : maximum number of main updating cycles allowed.

E) The 'slope hotstart'

The 'hotstart' feature incorporated in MOUSE allows to effect a significant reduction of the time required in order to carry out the slope evaluation process. In this process a perturbation \((\Delta x)\) is produced in each of the \((n \times m)\) components of the solution matrix. This feature takes
advantage of the fact that an extra simulation is only required starting from the 'time' in which the perturbation is produced.

This simulation is then attached to the time series of the results of the pipeflow simulation corresponding to the default strategies.

5.5 Levels of parametric optimisation

As suggested in chapters 3 and 4, the real-time control of urban drainage networks is achieved in this framework through a process composed of consecutive 'stages' of increasing levels of accuracy. The optimisation at each stage is performed on the basis of the results of the previous stage.

This scheme introduces several advantages in relation to an 'all or nothing' strategy. These are:

i) The optimisation process can be interrupted at any moment, depending upon the time available. Regardless of which stages have to be discarded, we will always obtain results with are more or less accurate, that is, within the present framework the 'optimal' control strategy is only refined as time elapses.

ii) Due to a proper arrangement of factors, it is possible to furnish a fairly good control strategy well in advance of the critical moments of the hydraulic load from precipitation (from the logical optimisation step).

In order to describe the levels of parametric optimisation achieved in each of the stages, it is necessary to understand what is done in each of the main steps of the combined logical-numerical framework.

In brief, it is possible to state that, in the logical optimisation step, knowledge about the system is utilised (or extracted from all available sources) in order to derive a better control strategy. This control strategy is none the less still only an approximation: it cannot be regarded as an 'optimal control policy' in the classical sense of this phrase. As is generally acknowledged (Gonwa et al, 1989), logical methods provide many advantages for real time control but they are not exact, in the sense of providing the 'feasible-constrained global optimum' for the process.

As a matter of fact, due to the presence of every kind of physical constraints in the system, not even the dynamic or non-linear optimisation techniques based on a rigorous mathematical theory can approach the absolute global optimum for this process. In this sense we should speak of the 'constrained global optimum' or the 'feasible global optimum' in the real-time control of urban drainage networks.
On the other hand, in the numerical optimisation step, the elements of the classical theory of non-linear optimisation (see Chapter 3) are applied in order to further refine the results obtained in the logical optimisation step. Given that we have been capable of providing a good economic model for the problem, these results will be further refined: we can guarantee that - provided the above mentioned conditions are fulfilled - the solution obtained at this stage represents one of the constrained local optima for the process.

If the optimisation process has been properly tuned from the beginning and good forecasts are provided, the results from the complete cycle of combined optimisation (the logical optimisation step plus the first cycle of the numerical optimisation step) in this framework may correspond to the 'feasible-constrained global optimum' mentioned above.

The critical point in this process seems to be related to the rainfall forecasts (section 2.8). In effect, the accuracy of the surface runoff and pipeflow forecasts is conditioned to the accuracy of the rainfall forecasts. If the rainfall forecasts are good, so will be the runoff and pipeflow forecasts. It must be remembered now that the present optimisation framework is based on a scheme of dynamic control which equally relies on on-line measurements and forecasts.

This is necessary in order to avoid partial decisions, i.e., decisions which represent a good control decision for the time being while they might have adverse consequences in the future stages of the event.

5.6 Detailed description of the combined framework

The next few sections will be devoted to the description of the main modules of the combined logical-numerical framework. This description will be provided from several points of view: the structured programming point of view, the functional point of view and the integrational point of view. This is necessary because each of these modules are to be understood in the general context: the combined logical-numerical framework developed for optimisation in the real-time control of urban drainage networks.

5.6.1 QNEWT: the constrained quasi-Newton optimiser

The program QNEWT is a structured piece of computer code developed using Hewlett-Packard C for workstations within the UNIX environment. This program constitutes the core of the fully non-linear numerical optimiser provided in the combined logical-numerical framework. It is primarily oriented to the real-time control of urban drainage networks, although it could be extended to the control of hydrodynamic networks in general.

The conceptual basis of this program have been explained in Chapter 3. As indicated by its name, this program implements a Quasi-Newton methodology for the search of a constrained minimum in a non-linear function of many variables.

The solution is given by this program in the form of a set of strategy vectors, i.e., series of control decisions which cover the whole forecasted horizon. This program requires the use of
hydrodynamic models: runoff and pipeflow models. The forecasting capability is very important here since the numerical optimiser implements a scheme of dynamic control (section 2.2) which means that the control decision is not only taken on the basis of the existing situation in the system, but also on the forecasted one.

Among other parameters - which will be discussed in detail in sections 6.1 and 6.2 -, the program QNEWT requires the specification of initial (constrained) search vectors in order to start the search. These initial vectors can be located at any point in the constrained search space. This program will then find the local optimum located in the neighbourhood of the starting point. The better the initial vectors, the closer the solution vector will be to the 'feasible-constrained global optimum' discussed in section 5.5.

Several features (section 5.4) aimed at real-time implementation have been developed in program QNEWT. This program also implements the most effective numerical methods so far available for non-linear multivariable search. These are:

i) A steepest-descent Quasi-Newton algorithm for multivariable search which makes use of the matrix updating system given by the well-known Broyden-Fletcher-Goldfarb-Shanno rank-two formula.

ii) A Fibonacci methodology for the search of a constrained optimum in the single-line search procedure.

iii) A feasible-directions algorithm for constraining the search.

One of the main characteristics of this program is that it provides a full user control of all aspects of the search. As explained in section 5.4, this is a feature of central importance in fully non-linear optimisers intended for the [highly dynamic] problem of real-time control of urban drainage networks. The order of the steps in the generalised non-linear optimisation methodology can be schematised for structured programming purposes as illustrated in figure 38.

Computationally speaking the sequence of the main steps to be taken by the constrained Quasi-Newton optimiser are:

5.6.2 CFGEN: the cost function generator/editor/analyzer

The construction of the economic model is an issue of central importance for the successful performance of all numerical optimisation systems. As explained in section 3.2, the results of the numerical optimisation process can only be as good as the economic model is a good representation of the complex processes occurring in the network during storm events.

In effect, a good economic model has to be capable of describing accurately the existing relation between the system state variable and the cost which is assigned to these values.
EVALUATE INITIAL STRATEGY VECTORS
Construct first approximation to the Hessian matrix

LOOP ON ALL MAIN UPDATING CYCLES

PERFORM SLOPE EVALUATION PROCESS
- Decrease perturbation through all points in solution array.
- Evaluate cost of new situation
- Construct gradient vectors

UPDATE APPROXIMATION TO HESSIAN MATRIX
- Assemble search vector $s$
- Determine the steepest descent direction

CONSTRAIN THE SINGLE-LINE SEARCH
- Find distance between current point (CP) and the boundary which is pointed by the search vector (arrow).
- BfMcketting phase
- Securing phase

PERFORM THE SINGLE-LINE SEARCH
- BfMcketting phase
- Securing phase

DERIVE NEW STRATEGY VECTORS
- Check 'projection' against constraint

CHANGE CURRENT POSITION TO NEWLY DETERMINED VECTOR
- Take this vector as the new current point (CP)

CHECK STOP CRITERIA:
- Convergence criteria
- Criteria of maximum number of iterations

Figure 38: Structured programming representation of the program Qnewt

Therefore we could use the following statement to describe this process: the smarter the construction of the cost functions the more successful the results of the numerical optimisation process.

There are several rules which must be observed in order to construct a 'good enough' economic model for this process. These are:

* The objectives to be achieved with the system's optimization (elimination or maximal reduction of potential damages of pollution and flooding to the surrounding environment) must be clearly mirrored in each of the cost functions.

* The relative importance of one factor in relation to the other is not predefined. It depends on the specific place where the analysis is being performed.

* The costpoints in the system must be located in strategic places, that is, places in which the damages can be easily evaluated (observed) according to accurate and well defined quantitative criteria.
void main()
{

/* COMPUTE ORDER OF MATRICES */
lim = nregul * npstrat;

/* COMPUTE COST OF INITIAL CONTROL STRATEGIES */
write_strat_file (0, x stratfilename, 0);
CSSaved = 1; /* This simulation is to be saved for next 'slope hotstarts' */
INDIC = 1;

/* Transfer global hotstart parameters */
TotCost = EvCost (MD, timestep, saverate, nregul, pCP, INDIC, HiSwt, hox_ref, hox_ref, netfilename, stratfilename, nnetSen,
CSSaved);
count ++;

/* FIRST SLOPE EVALUATION */
slope (TotCost, 0);

/* ESTIMATE FIRST APPROXIMATION TO THE SERIES OF MATRICES IN BFGS FORMULA */
Identity_Matrix (x, /* Ensure positive definiteness for minimisation */

/* LOOP UNTIL TERMINATION CRITERIA ARE SATISFIED */
while (toler > eps) {
    OldCost = TotCost;
    iter ++; /* increase iter after storing old cost */
    /* CONSTRUCT NEXT APPROXIMATION TO THE HESSIAN MATRIX */
    Hessian_Matrix (0); /* vector s[] computed */

    /* PROJECT VECTORS FOR CONSTRAINED SINGLE LINE SEARCH */
    upper = find_upper_limit (0);

    /* PERFORM THE SINGLE LINE SEARCH */
    alfa = fibon (upper); /* single-line search, projecting vectors */
    New_Settings (); /* and compute new TotCost */

    toler = fabs (TotCost - OldCost);
    if (toler == eps) { break; } /* end of if */

    /* CHECK MAX. NUMBER OF ITERATIONS */
    if (iter >= nmax) { break; } /* end of if */

    /* END OF BFGS CYCLE */
    / END OF SLOPE EVALUATION */
    /* SLOPE EVALUATION -> CONSTRUCT GRADIENT VECTORS FOR NEXT CYCLE. */
    slope (TotCost, 1);

    /* UPDATE BFGS MATRIX FOR NEXT SEARCH */
    BFGS_Update (0);
}
/* end of main */
* A smooth mathematical cost function must always be preferred before an irregular one. The 'key' values of the system state (for example the level of the street in a given town) can be taken as points of change of slope on the curve.

* The cost function must assign a cost to every possible value of the system state variable in the system (that is the cost hypersurface must be 'large enough' so as to avoid the situation in which the optimiser cannot find the cost associated to a given system state variable).

* A balance should be achieved between the situation of open regulators and closed regulators, in such a way that the optimal situation preferably points to intermediate settings for the regulators. If required, this objective can be achieved by constructing the cost functions in such a way that the consequences of extreme strategies are highly penalized.

In principle, the main objectives to be achieved with the optimization can be summarized in the following maxim: 'keep the overflows as small as possible while prevent the streets from flooding'. Further extensions of these objectives might include the minimization of electric energy consumption for the system operation, optimization of the working load on the treatment plant, and flow equalisation. In order to provide effective help in meeting some of these criteria, the program CFGEN has been constructed. This program is equipped with several facilities, such as the analytical generation of well-known mathematical functions, file handling, graphic visualization and edition of curves, generation of ASCII tables with coordinates pairs, equations, main fitting parameters, etc. Program CFGEN provides the following menu of analytical cost functions:

1). \( y = ax + b \) (straight line: with possibility of \( n \) segments) \( (5.4) \)

2). \( y = ax^2 + bx + c \) (quadratic parabola) \( (5.5) \)

3). \( y = ax^3 + bx^2 + cx + d \) (cubic parabola) \( (5.6) \)

4). \( y = ax^4 + bx^3 + cx^2 + dx + e \) (4th degree polynomial) \( (5.7) \)

5). \( y = ax^n \) (power function) \( (5.8) \)

6). \( y = e^{ax} \) (exponential function) \( (5.9) \)

7). \( y = \frac{a e^{(ax)}}{e} \) (modified exponential) \( (5.10) \)

where:
\[ a = \frac{(x - x_{\text{min}})}{(x_{\text{max}} - x_{\text{min}})} \]  

(5.10.a)

\[ y = \log(x) \]  

(5.11)

All the coefficients are user-specified. The range of variation of the independent variable \( x \) between the limits minimum \( x_{\text{min}} \) and maximum \( x_{\text{max}} \) is also user specified. In the equation of the modified exponential function (7) the cost function is expressed in relative terms. The functions (1) (with number of segments \( > 1 \)), (5), (6), (7) and (8) are especially well-suited to problems of this kind. A valuable graphic visualisation/edition facility has been incorporated into the program CFGEN, with the objective of allowing the user to perform a fine tuning of the constructed cost functions and an identification of potential inconsistencies in these functions.

This is achieved by performing a call to one of DHI’s standard XWindows applications (DHIGED) equipped with the above mentioned edition facilities. In this way the user can edit the points which were analytically generated, drag them individually over the graph (with coordinate tracing), form a satisfactory file and save the modifications to an ascii file with extension .CFG. This graphic visualisation/edition facility incorporated into the program CFGEN is often used in order to edit the time variation curves describing the variation of the global cost with time (ascii files with extension .VTS). This feature has proven to be a useful representational tool in certain cases in which it is necessary to indicate that additional penalty terms have to be added to the cost at certain stages of the event horizon.

For example this feature could be useful to indicate to the combined optimisation system that it is necessary to release water towards the end of the event horizon in order to maintain a certain free capacity which could be used in case another rain event occur rapidly after the current event.

It must be pointed out that, since the cost evaluation system used by this framework is based on a cost integration over the event horizon, the definition of [non-unitary] time variation curves will affect the global cost associated to a given strategy at all stages of the optimisation process. This situation is illustrated in figure 39.

As explained above, as the global cost of a strategy at each of the costpoints is obtained through time-integration, the 'shape' of the time variation function is taken into account as a whole.

The time variation curves define cost multiplication factors (CMF) to be applied on the cost functions. Usually the values of these CMFs are taken between zero and two.

The default values of the time variation function is taken as 1.0. This corresponds to a horizontal straight line in the lower graph in figure 39.
However, even if these values are meant to be considered as unity, they must be specified in the .VTS files.

The way of specifying these parameters for an optimal performance of the combined logical-numerical framework will be further discussed in Chapter 6.

5.6.3 CFCALC: the cost function calculator/evaluator/integrator

So far we have explained the process of generation of cost functions. These functions in combination with the time variation curves will generate - when filtered through the dynamic time series of the system state variables at each of the keypoints - the economic model for the process. In turn, this model allows the definition of the solution space through the hypersurface conversion process described in section 5.3.

The assignment of an exact value of cost to every cost point in the network involves a complex process composed of hypersurface conversion and time integration. In order to explain this process from the programming point of view, it is convenient to concentrate on what happens at a single costpoint.

Initially, at this level it is necessary to construct a temporary 'cost time series'. This means that it is necessary to 'filter' the cost function at the given costpoint, say A, through the dynamically obtained time series of the system state variable at the same costpoint A. In other words, given a certain control strategy, the system must be able to 'know' the exact cost of this strategy at point A at each of the stages defined for time-discretisation.

Once this 'cost time series' has been constructed, the code is prepared to perform the time integration. This step involves the definition of a 'time resolution' in order to divide the cost time series into multiple 'slices'. By computing the area under the curve in each of these slices, we obtain, by integration, the area under the curve of cost versus time at each costpoint. This area represents the total time-integrated cost associated with the given control strategy at costpoint A.

Now by repeating the same process at each of the costpoint we can compute the global time-integrated cost of the given control strategy in the system. The optimisation process is aimed at finding the strategy which minimises this global time-integrated cost.

All of these tasks (including the call to the MOUSE hydrodynamic model to generate the time series as a result of the given control strategy) are performed by the program CFCALC. This program needs to be supplied with the following parameters:
- Name of the input file (text file) containing the curve of water levels (h) against cost associated to the costpoint being analyzed (static, generated by CFGEN, \( \rightarrow .CFG \) file).

- Name of the input file (text file) containing the time series of the system state variable at the costpoint. This time series has of course been generated previously by MOUSE hydrodynamic model and taken from a temporary array (shared memory or .TMP file) to a dynamically generated ascii file. This task is performed within the communication centre LEAD (dynamic, generated by MOUSE, \( \rightarrow .TSF \) file).

- Name of the input file (text file) containing the curve describing the variation in time of the cost function. In those cases where this variation is not significant, a horizontal straight line which intersect the Y axis at \( y = 1 \), must be supplied. (semi-dynamic, generated by CFGEN, \( \rightarrow .VTS \) file)

- An operator describing the process to be applied on the input curves (valid operators 1 to 5). The following operations are supported by CFCALC, according to the specified operator (\( S \) stands for system state variable):

1. Find combination \([S_{\text{max}}, t, \text{Cost (S_{max})}]\), by using information from the three input files (.CFG, .TSF, .VTS).

2. Find combination \([S_{\text{min}}, t, \text{Cost (S_{min})}]\), by using information from the three input files (.CFG, .TSF, .VTS).

3. Find combination \([S_{\text{avg}}, t, \text{Cost (S_{avg})}]\), by using information from the 3 input files (.CFG, .TSF, .VTS).

4. Find the combination \([S, t, \text{Cost (S)}]\) for all points in the time series (according to the time resolution with which it was calculated by MOUSE), form the 'cost time series', and then integrate it in order to find the 'total' cost associated with the costpoint (It requires the use of the information contained in the 3 x j cost files: .CFG, .TSF, .VTS).

5. Apply all processes 'simultaneously'.

The operator commonly employed in the numerical optimisation sub-process is the number 4. The other operators may offer useful information during the process of 'calibration' of the economic model for the urban drainage network.

It is also necessary to supply CFCALC with the name of the output file where the results of the cost calculation/evaluation are going to be stored (extension .CFC) and also with a keyword to confirm that all input files are ready to be processed.

The call to CFCALC is shown here with one example (The option -help is available):

cfcalc -confirm y -nrcostpoints 7 -costfile CstCP1 -tserfile tserCP1 -timfile tmultCP1 -operator 4 -outfile CostEv1
It should be observed that the program CFCALC must be called within a loop of \( j \) costpoints with changing parameters. Fortunately, the process is fully automatized, and the user does not perceive any of these rigorous transfer of parameters. This job is performed by the communication centre (program LEAD). CFCALC is a stand-alone (separate executable) module due to its internal complexity.

5.6.4 TUNING: the module to specify optimal parameters for the single-line search

As mentioned before, there is agreement among authorities in the field of classic optimization about the importance of providing the search algorithm with an efficient procedure for the single-line search subproblem.

It was also explained that the present fully non-linear optimization system works with a Fibonacci line search procedure, in which the number of experiments must be specified in advance. Some other parameters must also be specified in advance, for this Fibonacci line search procedure, such as the fractional resolution based on the initial interval of uncertainty (\( \varepsilon \)).

There are several ways of implementing the Fibonacci line search procedure. There is, for example, the possibility of specifying the length of the final interval of uncertainty desired from a given initial interval of uncertainty. The program will then determine the number of Fibonacci experiments to be performed in order to arrive at this final interval. Actually, this was the first variant implemented in the numerical optimisation framework.

However, this is not effective enough for solving our problem, and especially so, if we think of the possibility of real-time implementation with our present computers. The number of function evaluations must be totally user controlled for the sake of real-time implementation.

Accordingly, we had already at this stage an additional problem to solve. The problem was to correlate such parameters as the initial interval of uncertainty for the Fibonacci search (internally computed by the program as explained before), the final uncertainty interval, and the fractional resolution based on this final interval.

In effect, we had to ask how we could know in advance if the number of experiments specified would be insufficient, excessive or just enough to reach the final interval of uncertainty from the starting interval, given a value of the fractional resolution (\( \varepsilon \)).

The answer is found in equation (3.24). If we rework this equation to express \( \varepsilon \) in terms of \( I_0 \) and \( I_n \) and make the substitution \( n = n_e \), we obtain:

\[
\varepsilon = \left( \frac{F_{n+1}}{I_{n}} \right) \frac{I_{n}}{F_{n+1}} - 1
\]

\[(5.12)\]
Which provides an iterative procedure in which, given an initial interval \((I_0)\), a desired final interval \((I_{ne})\) - which can also be specified as interval reduction from a relative unitary initial interval of uncertainty - and a number of experiments \((n_e)\), we can compute an interval resolution \((\epsilon)\).

The 'optimal' number of experiments is that, which allows us to reach (approximately) the final interval of uncertainty from the initial one with the least possible number of experiments, that is, with the smaller non-negative value of \(\epsilon\). The procedure is implemented in program TUNING. The input data needed are: \(I_0\), \(I_{ne}\) and several values of \(n_e\) (number of experiments for the single-line search).

Then the program TUNING generates a series of \(n_e + 1\) fibonacci numbers, and will evaluate equation (5.12) for all values of \(n_e\), choosing the 'optimal' value of the fractional resolution corresponding to the given number of experiments during the line search and to the final interval of uncertainty (also user specified). These parameters should be specified as input data in the file 'exper.inp'.

It is important to notice that the way in which this algorithm converges to the optimum also depends on the specification of this relation as it will determine the initial parameters for the single-line search. In effect, the value of the interval resolution \(\epsilon\) (which is user-specified) combined with the value of the initial interval of uncertainty \(I_0\) (internally calculated by the optimiser), determine the position of the first two Fibonacci experiments, as can be seen from equation (3.24). Only if the number of Fibonacci experiments (also user specified) is large enough, will the search be terminated in the same way, but this would probably be achieved at a relatively high computational cost. Therefore, a 'reasonable' specification of the relation between \(n\) and \(\epsilon\), is highly recommendable in order to achieve the desired results. Once the ratio \((I_0/I_{ne})\) is decided according to practical criteria, this relation can be specified by using the TUNING computational module.

5.6.5 LEAD: the communications centre

Due to the relatively large number of programs and libraries involved in the combined logical-numerical optimisation framework, it has been necessary to develop a 'main communication centre', i.e., a module capable of 'scheduling' the order of events in the optimisation process.

This module is also responsible for the heavy transference of parameters between the main components of this system. In the same vein, program LEAD is in charge of the transference of dynamic (changing) parameters to the MOUSE hydrodynamic models. For example, this program selects those scenarios requiring 'special hotstart', i.e., the hotstart required from the function 'slope' (in charge of the slope evaluation process) involving 'time-shifted' control strategies. As can be observed from figure 5.2, it is this program that makes possible the interaction among all the modules integrating the numerical optimisation sub-system. An example of a typical call to the MOUSE hydrodynamic pipeflow models would be:
This program must also 'keep track' of the way in which the cost evaluation process takes place, and the selection of the kind of hotstart to be used at a particular stage of the process. The final result of such a cost evaluation process is a global time-integrated cost corresponding to a given control strategy, which is returned from the program LEAD to the fully non-linear Quasi-Newton optimiser.

Finally this program contains the 'dynamic constraint function' described in section 5.4 which avoids those simulations which are physically trivial for the given system. The program "LEAD" has the following algorithmic structure:

```c
float Lead (dur, timestep, saverate, nreg, nCP, FLAG, CShot, rrftime, prftime, netname, stratname, nns, CSS)
{
    /* Compute basic parameters for cost integration */
    resol=(timestep * saverate) / 60; /* time resolution (minutes) */
    n=int(1+MD/resol);
    /* With current_strategy do */
    run_runoff_model (..., ...,);
    run_pipenow_model (..., ...,);
    Check_Dynamic_Constraints ()
    /* Loop through all costpoints performing cost integration */
    for (i=0; i<nCP; i++)
    {
        Perform_Time_Integration ();
    } /* end of for */
    } /* end of Lead */
```
5.6.6  PIPFLW: The interface with MOUSE ONLINE

Due to the complexity of the modelling processes which take place in the MOUSE package as well as the relatively large number of possibilities involved, it is necessary to provide one more stage in the integration scheme of the combined logical-numerical optimisation system.

This intermediate stage is the program PIPFLW. This is a standard DHI working routine which has been developed outside of the framework of this study. However, a description of its main characteristics is necessary here due to the role it plays in the process and also for the sake of completeness.

This program represents the MOUSE ONLINE way of communicating with the internal functions of the MOUSE standard modelling package. As an example, PIPFLW issues calls to the functions openmouse() and runmouse() (MOUSE pipeflow model) in charge of performing the initialization of parameters and the modelling of the pipeflow processes, respectively.

There are a number of parameters supported by program PIPFLW which allow the activation or deactivation of options in the standard MOUSE modelling packages. These are:

- `netfil <NET_file_name>`
- `regfil <CTR_file_name>`
- `swffil <SWF_file_name>`
- `rffil <RFF_file_name>`
- `pwoffil <PWF_file_name>`
- `bwoffil <BSF_file_name>`
- `prffil <PRF_file_name>`
- `hotfil <HOTSTART_file_name>`
- `duration[min] <event_horizon [float]>`
- `timestep[s] <timestep [integer]>`
- `saverate <saverate [integer]>`
- `filhot <switch_hotstart_yes_or_not>`
- `debug <trace_run>`
- `help <user_help>`

Of course, the indicated extensions refer to the same standard notation employed by the MOUSE modelling system (see MOUSE documentation, DHI, 1992)

5.7  The double blackboard architecture

As a system which is primarily intended to deal with the operation of urban drainage networks, the MOUSE ONLINE system is basically composed of three main sub-systems which constitute white-box models (Nielsen et al, 1991). These sub-systems rely on the fully dynamic pipeflow models of the standard MOUSE package. These subsystems are:

MOUSE PILOT, used for the evaluation of a large number of control strategies based on long-term deterministic simulations. This system is also used for the assessment of the potential of the system for applying real-time control techniques.
MOUSE ONLINE operator's system, which is used as the 'operational control unit' in the urban drainage network operated with real-time control techniques. It includes the optimisation modules and several user interfaces. This sub-system interacts with the standard SCADA system.

MOUSE SIMULATOR which is used for planning and design of the real-time control system in the given network, as well as for the testing of the final software configuration.

The exchange of information among these modules must be done in a safe and fast way. This is essential to real-time control. One of the most convenient ways of doing this is by using a double blackboard architecture (within the UNIX environment for workstations), as shown in figure 40.

The different modules integrating the MOUSE package share and exchange information by writing to and reading from a common data area which is called the internal blackboard. Another blackboard - the external blackboard - is used for exchanging data between the MOUSE ONLINE system and the standard SCADA system.

This architecture provides the MOUSE ONLINE system with high flexibility. In this scheme, modules can be substituted or added rather easily. The requirement that they all must fulfil is that they have to be capable of reading and writing to the blackboards according to the common specifications.

The double blackboard architecture also makes it simple to exchange data with 'external' programs. For example, different forecasts of the expected flows towards the treatment plant can be transferred to the local control system of the treatment plant. In the other direction, it is possible to receive and use information regarding the variations in the capacity of the treatment plant if such information is available.

This architecture includes several options for optimisation in the real-time strategy selection process (real-time control). At the moment it is intended to make use of the present combined logical-numerical optimisation framework.

5.8 The intelligent agent. Approaches used for decision making

As described in Chapter 4, the intelligent agent built into the present combined logical-numerical framework makes use of several knowledge-based-related techniques in order to approach the optimal strategy.

These techniques are those of diagnosis, heuristics and fuzzy logic which are all incorporated into the resulting knowledge base. This intelligent agent also performs an 'on-line learning process' of limited scope.

So far we have described the conceptual aspects of these techniques, but now the description will be given in terms of their practical implementation. We are now concerned with the description of the modules which perform such tasks.
Figure 40: The double blackboard architecture of MOUSE ONLINE
5.8.1 HYDNET: The knowledge base

Intelligent behaviour is related to the knowledge that an entity possesses about its environment. Usually an important part of this knowledge is descriptive and therefore it can be expressed in declarative form.

In order to 'model' its environment, a knowledge base requires basic 'bricks' or 'cells' of information, since this is also the way that the Real World is built. These 'bricks' are meant to contain essential information, including encapsulated knowledge, about units in the environment. This scheme of representation is called *object oriented representation*.

As explained in section 4.6 objects are indeed interrelated through very diverse links. They are generally grouped in classes and subclasses. Sometimes* it is necessary to encode knowledge whose structure is unknown a priori and can only be determined in real-time.

The present framework incorporates a knowledge base - called HYDNET.kb - which has been designed as a general tool. As explained in Chapter 4, several techniques have been employed in order to make this general representation possible. This knowledge base is not intended for site-specific application, but has been designed in such a way that only knowledge based on general properties of the domain has been encoded in its rules, using fuzzy logic sets. The knowledge based on site-specific properties is acquired in real-time through the above mentioned 'on-line learning process'.

A description of the representational scheme employed by HYDNET.kb must therefore be included. The logic programming structures used by the knowledge base are summarized in the table 5.1.

---

* especially in less well structured problems, where many of the features of the problem are not known beforehand.
Table 5.1: Logical representational structure in the knowledge base HYDNET.kb

<table>
<thead>
<tr>
<th>Classes</th>
<th>Sub-Classes</th>
<th>Objects</th>
<th>Properties and Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>[*] Branches</td>
<td>None</td>
<td>[*] Dynamic objects</td>
<td>Inherited for dynamic objects</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(as many objects as branches exist in the real-world network).</td>
<td>* FreeInCapac &lt;float&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[*] Balance</td>
<td>[*] FlowBranch_A .. FlowBranch_Z</td>
<td>[*] Dynamic objects</td>
<td>Inherited for dynamic objects</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[*] Inflows</td>
<td>[*] InflowBranch_A .. InflowBranch_Z</td>
<td>[*] Dynamic objects</td>
<td>Inherited for dynamic objects</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[*] Outflows</td>
<td>[*] OutflowBranch_A .. OutflowBranch_Z</td>
<td>[*] Dynamic objects</td>
<td>Inherited for dynamic objects</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[*] PhysNetwork</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[*] Regulators</td>
<td>None</td>
<td>[*] Dynamic objects</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(as many objects as regulators exist in the real-world network).</td>
<td>* link_pos1 &lt;string&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[*] Sensors</td>
<td>None</td>
<td>[*] Dynamic objects</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(as many objects as sensors exist in the real-world network).</td>
<td>* link_pos1 &lt;string&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[*] Strategies</td>
<td>[*] Strat_R1 .. Strat_RZ</td>
<td>[*] Dynamic objects</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(as many objects as regulators exist in the real-world network).</td>
<td>* range &lt;float&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[*] TimeSeries</td>
<td>[*] TSer_Nod1 .. TSer_NodZ</td>
<td>[*] Dynamic objects</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(as many objects as regulators exist in the real-world network).</td>
<td>* SetPA &lt;float&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.8.2 LEARN: the 'learning' section

This section describes the development and implementation of an 'on-line learning section' - of limited scope - within the knowledge base HYDNET.kb. This 'on-line learning process' is considered as incomplete and spatially localized. The main objective of this process is to identify the reasons for 'potential undesirable consequences' arising from the application of extreme (known) control strategies in the system. This information is essential for real-time control.

This learning is said to be incomplete due to the fact that its knowledge does not increase continuously with time: its scope is limited to the duration of the event horizon. Therefore this knowledge cannot be used in subsequent events. It is reset when the controlled event is finished. On the other hand, it is spatially localized because the associations are establish between subsets of the learning parameters and spatially localized regions of the input-space (section 4.7.1, p. 4-18). For example, the 'corrections' to the extreme control strategies for a given regulator - in those cases where they produce 'undesirable effects' - are computed on the basis of the analysis of the consequences in the set of keypoints linked to the regulator.

This process can obviously only be approximated, although it is general. The generality is understood here as the capability to handle any flow situation. This is because this section operates on the basis of the existing flow situation as described by the information provided by on-line sensors as well as the flow forecasts.

As suggested above, the main notion behind this implementation is to establish an individual - implicit - relationship between input stimuli (known control strategies at a given regulator) and outputs of the system expressed as time series of the system state variable at the set of its associated costpoints. In general, this is a 1 : n relationship.

Therefore, the default control strategy (taken as a time-dependent vector for dynamic control) in the regulator, say X, is supposed to be appropriate as long as the constraints imposed for the system's operation are satisfied at the set of costpoints associated to the regulator. If one of these constraints is violated, an adjustment of the 'learning parameters' \( \eta(t) \) is required. It should be observed that these learning parameters have been represented as a function of time. In effect, they are imposed as a time function supposed to satisfy a certain predefined implicit relationship. Only those elements of the array of learning parameters \( \{ \eta(t) \} \) where the constraint is violated, are adjusted.

The adjustment consists of imposing the limiting condition (that is, the system state variable is forced to follow the constraint violated) and using the learned implicit relationship in order to obtain the corresponding modifications to the control strategy, in the opposite direction. The source utilised in order to construct such a relationship is of course a finite (and indeed a relatively small) number of fully-dynamic MOUSE simulations.

The result of such a process is only approximate. Indeed, this is the objective. The learning process is just one of several approximations to a 'quasi-optimal' control strategy performed within the knowledge base HYDNET.kb. They are further refined by both, the intelligent agent and the fully non-linear numerical optimiser.
5.8.3 UINFO: The module for post-processing of the user-supplied information

As explained in section 4.5, it is often necessary, in order to provide an effective representational tool for the factual knowledge based on general properties of the domain, to explicitly derive the site-specific network information at 'runtime', that is, when performing the real-time control of the actual event on a given hydrodynamic network.

This process involves the construction of a number of matrices (figure 27, section 4.5) containing factual knowledge based on site-specific properties of the network. This information is related to the connectivity of the nodes in the network, the reachability of a set of nodes from a reference node A, the sub-units in the main network and so on.

In order to make this process flexible, accurate and very fast it is necessary to provide a general representational framework with which the user can provide a realistic Real-World model. This representational framework is very important, since a correct information about the way that a system must be operated is only possible on the basis of an accurate input information.

The method employed by the intelligent agent in order to allow the features described above is based in the decomposition of the main hydrodynamic network into a finite number of sub-networks, which constitute functional entities. That is, the decomposition is done in such a way that the processes which occur in one of these entities do not affect any of the other entities.

These entities are physically related to natural reservoirs which are formed by the set of physically linked pipes, manholes and structures upstream of the main (active) flow regulators.

The intelligent agent uses a user specified frame-recognition system in order to identify these operational units. This situation is illustrated in figure 41. All that it is required is to supply a 'coordinate frame (box)' by specifying the coordinates of the lower-left and the upper-right corners of each functional sub-network. The information provided by this representational framework is then used in order to construct the connectivity and reachability matrices described above. This task is performed by the function UINFO. As indicated, UINFO is not a stand-alone piece of computational code but a function incorporated in the library of C functions of the intelligent agent.

There are internal 'rules' to resolve those cases where the same nodal point is included in more than one recognition frame. However, if the recognition frames are properly defined, these cases can be considerably reduced.

5.8.4 MASSBAL: the mass balance in the system's sub-networks

Once the hydrodynamic network has been 'classified' and the intelligent agent gathers the factual knowledge based on site specific properties, a number of approximations to a quasi
Figure 41: The frame recognition system employed in HYDNET.kb
Combined Logical Numerical Enhancement of Real-Time Control

-optimal control strategy are constructed by this agent.

As described in Chapters 3 and 4 the intelligent agent performs several tasks in order to enhance the optimisation capability of the fully non-linear numerical optimiser. These tasks are basically those of providing good initial vectors for the numerical search as well as providing a first approximation to the Hessian matrix. The intelligent agent has also the important task of steering the numerical optimisation process.

The approach utilised in order to provide such a first estimation of the Hessian matrix is based on the idea of performing a mass-balance in each of the system’s subnetworks for all of the times defined in the discretisation of the time-dependent vectors (including strategy vectors). We must remember that these sub-networks are identified by module UINFO on the basis of the user-supplied information. Then, by using fuzzy heuristic rules of the kind described in section 4.5, it is possible to define correct ‘policies’ for the operation of the main flow regulators. This task is performed by the program MASSBAL (interfacing NEXPERT OBJECT and C).

The results of the program MASSBAL are promising 'expected gradients' from a default position (the final strategy derived by the intelligent agent) for each of the points defined in the solution matrix*.

The construction of this approximated matrix involves a considerable number of computations which are performed on the basis of the time series obtained from the fully dynamic simulation of the final control strategy on the urban drainage network.

The program MASSBAL computes - for each time defined in the discretisation of the control strategies - the inflow volume to the branch, the outflow volume from the branch and the actual rate of change of inflows, and it compares this last value with the recommendable fuzzy parameters from the rules. The favourable gradient is then determined from the results of such a comparison. The knowledge for deciding upon the proper control action is encoded in fuzzy heuristic rules as described earlier.

We must observe that in order to determine the inflowing and outflowing volumes, it is required to have previously identified all overflow points and catchments belonging to each of the subnetworks (also called here 'branches').

This task is also performed by program MASSBAL on the basis of reading the information contained in the standard MOUSE .SWF file. The results of such a classification task are sent to an ascii file called OVFW.STR which has the following structure:

* The solution matrix is defined in this dynamic non-linear implementation as the matrix of \( n_{cont} \times n_{pstrat} \) containing the control strategies for each of the flow regulators (ncont) covering all stages (nstrat) within the so-called event horizon.
The Combined Logical-Numerical Framework

<table>
<thead>
<tr>
<th>Branch Nr (BRC)</th>
<th>Overflow weirs in branch (OWIB)</th>
<th>Name of overflow weirs in branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>NAM1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>****</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>NAM10N, NAM10S</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>NAM17</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>NAM12, NAM13BP</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>NAM16IG</td>
</tr>
</tbody>
</table>

The program MASSBAL sends run-time information about active processes to the user screen. An example of the way that this information is summarized is given below:

* Reading relevant network data from file: HYDNET.DAT ...
* Reading catchment & pipe data from file: SW.F.TXT ...
* Writing available storage volumes per branches to file: AVL_STOR.VOL ...
* Performing key runoff simulation ...
* Reading time series of inflows from file: nam.dbg ...
* Writing time series of inflowing volumes per branches to file: INFLOWS.VOL ...
* Performing key pipeflow simulation ...
* Reading time series of outflows from file: pipe.dbg ...
* Writing time series of outflowing volumes per branches to file: OUTFLOWS.VOL ...
* Writing overflow weirs per branches to file: OVFW.STR

5.8.5 SHAPE: Resemblance of the natural behaviour of the system

As described, several approximations are performed within the intelligent agent in order to approach the quasi-optimal vectors (control strategies). The resemblance to the natural behaviour of the system introduces an interesting idea, which arises from the observation that ".. most quasi-optimal control strategies in a given regulator, somehow resemble the shape of the 'natural time series' of the system state variable in the neighbourhood of the flow regulator itself .." (Martin, 1994).

The natural time series are understood here as the time series obtained in a system without regulation, that is, when the usage of the storage capacity is a minimum. The solution of this problem is related to the derivation of 'shape coefficients' to be applied on the default strategies.

If properly employed, this approach provides a respectable contribution to the final control strategy conformed by the intelligent agent as initial vectors for the numerical search. We observe that this statement summarizes empirical (practical) knowledge obtained from the solution of similar problems by making exhaustive use of non-linear numerical search techniques.

The way in which the quasi-optimal control strategies resemble the so-called 'natural time series' depends on the type of the regulator and also on the system state variable observed at
the set of associated costpoints. Sometimes this resemblance is achieved by 'mirroring' these natural time series and on some other occasions it is identified by direct resemblance. It is important to notice that the above mentioned statement presumes an underlying 1 : n association between regulators and costpoints. The way of revealing these links is done through an 'off-line calibration process' which will be described in detail in section 6.2.

The proper specification of these internal links as well as the definition of priorities for the regulators are essential for the successful performance of this approximation.

5.8.6 INTERP: Convergence to a desired behaviour in the network

Once the consequences of extreme control strategies are known, it should be possible to 'correct' the potential undesirable consequences of those control strategies (by sweeping across the so-called information sub-network) between desired and actual setpoints in each of the information nodes (sensors). In order to translate the observed flow differences in the system into modifications of the control strategies for the flow regulators, it would (again) be necessary to use a previous identification of the links between regulators and sensors. It must be observed that this could be a 1 to n relationship since several sensors might be linked to the same regulator. In this way the main idea could be stated as follows: every sensor supplies a certain request to its associated regulator. These requests (modifications to the regulator's settings) are aimed at an improvement of the flow situation around the sensor itself. All requests to a given regulator are accumulated and a final modification of the control strategy is selected at every time level. We must observe that this variant is applicable to both schemes, the static and the dynamic control.

An accurate identification of the links between sensors and regulators as well as the definition of proper weighing factors for sensors requesting regulators are essential to the success of this variant. This information is of course site-specific and can only be obtained after calibrating the system with known (recorded) events.

The 'desired behaviour' of the system involves a high utilization of existing storage capacity, minimisation of combined sewer overflows, minimisation of ground flooding in certain keypoints in the system, reasonable utilization of electrical energy for the system's operation as well as smooth working conditions at the treatment plant(s). This program works directly on the output information of the learning process.

The output information from this program is also sent to the user. This allows a clear tracing of the process, and contributes to its transparency. An example of such 'messages' are:

| * Observed setpoints (from ECS) at costpoints A1 requires change of settings for regulator A3 at time t₁, ... |
| * Observed setpoints (from ECS) at costpoints NAM9 requires change of settings for regulator A3 at time t₂, ... |
| ... |
| and so on ... |
The translation of the suggested modifications into concrete control decisions (in terms of setpoints) can be done provided a \textit{proper relationship} input $\rightarrow$ output is available. This relationship is provided by the module LEARN. A good agent must also provide \textit{rules for resolving conflicting requests from sensors to regulators}. For example, if a regulator A is associated to points B and C, the situation may arise in which the situation observed in point 'B' requires the regulator A to 'close' while the situation at point 'C' requires regulator A to 'open'. In these cases the requests have to be 'weighed' (and 'accumulated' for a resulting control action) according to a \textit{system of priorities}.

It is important to notice that a number of strategic fully-dynamic runoff and pipeflow simulations are an important source of \textit{knowledge} - of various levels - about the system. In other words, there is a considerable amount of knowledge encapsulated in information form, which can be used for the real-time control of the system which is 'hidden' and can be revealed by a proper analysis of the consequences of the application of extreme control strategies on the system.

The ways in which this knowledge can be extracted and utilised (according to the ideas exposed in this section) is an issue of major importance in knowledge-based on-line control of urban drainage networks.

5.9 \textbf{SmartElements-C working interface}

The knowledge base HYDNET.kb has been developed within the development environment provided by \textit{SmartElements} (the later version of NEXPERT OBJECT). This is an efficient and flexible interface (which includes a graphical user interface) which has been provided by NEURON DATA -, the company which developed the \textit{SmartElements} package -.

Indeed, one of the main sources of power of the later versions of NEXPERT OBJECT is provided by its powerful interface with the C language and the libraries that NEXPERT OBJECT provides to facilitate such an interaction. The next few sections will be devoted to the description of the practical use made of this interface in developing the object HYDNET_RTC (the intelligent agent).

5.9.1 \textbf{KBRTC : the logical optimisation centre}

The program KBRTC (C code) has been developed to function as the \textit{main control centre} of the intelligent agent HYDNET_RTC. Program KBRTC performs the following tasks:

\begin{itemize}
  \item[i)] Recognize situations requiring real-time control (by trigger values)
  \item[ii)] Update \texttt{.nxx} files to be used by the intelligent agent
  \item[iii)] Load the knowledge base HYDNET.kb and initialize interfacing parameters
  \item[iv)] Set handlers for external C functions used by the knowledge base
  \item[v)] Trigger execution of fuzzy heuristic rules in HYDNET.kb by volunteering the 'start' hypothesis
\end{itemize}
vi) Supervise (control) the interfacing process
vii) Load - on user’s choice - the Graphical User Interface of NEXPERT OBJECT
viii) Once the process is finished, it transfer the control to the numerical optimisation system if the chaining switch (global) has been activated

The use of the hybrid SmartElements-C interface system introduces several technical advantages in relation to the development of a system which is only supported by SmartElements. The main advantages are related to speed, flexibility and maintainability of the resulting code. There are tasks which can only be performed from the logic shell, but there are other tasks (those of a rather algorithmic structure or involving heavy mathematical computations) in which the C language is more efficient. Once more, we have taken advantage of the best sides of each technique.

As is well known (NEXPERT OBJECT Application Programming Interface manual, preface) it is possible by using the Application Programming Interface (API) to extend the processing capabilities of NEXPERT OBJECT. This scheme is particularly advantageous for real-time processes (for example regarding the link with the data acquisition system).

For the sake of a complete illustration of the way that the overall control process can be designed by using this tools we shall provide a fragment of one of the possible variants of source code for the main function in the KBRTC.c control centre.

As suggested above, this is just one among several possible variants. This does not necessarily means that this is the variant implemented (many technical factors influence the final selection), but this variant has a clear structure which lends itself to methodological investigations. The specific implementation is influenced by the low-level design* of the overall control process, hardware factors and so on.

The interaction between the intelligent agent and the fully non-linear, numerical optimisation methodology occurs in multiple ways. When required, functions of the numerical optimisation framework are called from within the knowledge base in order to perform a certain rather algorithmic sequence of operations. In this way, the order of some of the operations in this framework depends (to a certain extent) upon the group of specific rules which are triggered in a given real-time control problem. The above can be illustrated by the following (logical) fragment of sub-symbolic code:

* In this context, the term 'low-level design' is used to refer to the process design at the level of the foundations of the system, i.e., at the direct interface with the SCADA system as well as the routines to access the shared memory region. It has nothing to do with complexity. Indeed, this could constitute one of the most complex levels of design.
void main (argc, argv)
  
  unsigned int argc;
  char **argv;

  
  
  AtomId Slotld; /* id of atom to be transferred */
  char ***kb_name="HYDNET.kb"; /* default kb name */
  KBId **KBptr; /* pointer to the kb */

  /* Perform command line arguments recognition */
  CommandLineArgs (...);

  /* Real-time control */
  RetrieveStateSwitch (...); /* data acquisition system */
  if (StateSwitch < triggers) {
    
    
  MonitoringLoop (...);
  }

  /* Select proper .nxd files for the version */
  system ("VersionX");
  system ("clear");
  NXP_Control(NXP_CTRL_INIT);

  /* Load the knowledge base */
  printf ("Loading Knowledge Base "%s", kb_name);
  NXP_LoadKB(kb_name, KBptr);

  /* Volunteer hyp. assigning a start value to the slot */
  NXP_GetAtomId("RainThreshold", &Slotld, NXP_ATYPE_SLOT);
  NXP_Volunteer(Slotld, NXP_DESC_STR, "TRUE", NXP_VSTRAT_VOLFWRD);

  /* Set handlers for C functions & start knowledge processing */
  NXP_SetHandler(NXP_PROC_EXECUTE, RTC, "RTC");
  NXP_SetHandler(NXP_PROC_EXECUTE, ANL_SUBNETW, "ANL_SUBNETW");
  NXP_Control(NXP_CTRL_KNOWCESS);

  /* Initialize Graphic User Interface (GUI) engine */
  if (UserSwitch == 1) {
    
    
  /* only for tracing purposes */
    printf ("nvNEXPERT OBJECT ...
    printf ("LOADING THE GUI OF NEXPERT OBJECT ...
    printf ("nvNEXPERT OBJECT ...
    NXPFX_Control(NXPFX_CTRL_INIT);
    NXPFX_Control(NXPFX_CTRL_START);
  }

  NXP_Control(NXP_CTRL_EXIT);
  InternalFunctionCalls (...);

  InterfaceOperations (...);

  ExitProcessFunction (...);

  } /* end of main function */
5.9.2 CNVFRM: the format convertors

We would like to conclude this chapter by devoting a few words to the role played by the format convertors. Due to the need for designing a hybrid intelligent control process (in the sense explained above), it became necessary to provide intermediate elements to facilitate the communication between different kinds of files or databases formats.
These intermediate elements are the format convertors. They have been named here as Cnv_frm. The two underscores are actually substituted by letters which indicate the particular kind of conversion they perform. For example, the file CnvTSfrm (TS stands for time series) is intended to perform the conversion (C → NEXPERT OBJECT) of the files containing time series of the system state variables at all costpoints to one of the NEXPERT database formats. This is necessary because this information (coming from MOUSE) needs to be used by the knowledge base.

The format convertors ensure the total automation of the control process and, although they are not 'visible' to the user in the sense of being major steps in this combined logical-numerical enhancement of real-time control, the overall performance of the process is conditioned by the performance of these format convertors. In this framework they make possible the communication between C-supported, standard ascii files and NEXPERT OBJECT databases.
Chapter 6 The Real-Time Control Process with the Combined Logical-Numerical Optimiser

6.1 Introduction

Now that we have described from a practical standpoint the most relevant characteristics of the modules which integrate the combined logical-numerical framework, we consider useful to provide a functional description of the optimisation process itself. That is, we are concerned in this Chapter with the description of the parameters which must be specified to the optimisation system in order that this process can take place successfully.

The optimisation process includes the construction of the economic model for the physical processes involved. We will cover this item in section 6.2 (Data specification), which will be structured from the functional point of view, i.e., in the same order as the user is supposed to specify these parameters so as to use the full potential of this hybrid optimisation framework.

The description provided here about the input and output of the combined optimisation process will be expressed in hydroinformatic terms. Therefore, the contents of the next few sections are closely related to the concepts discussed in Chapter 4. In general terms, this representation involves two processes (Abbott, 1993, with additional material from Amdisen, 1994 - :

a) The representation of knowledge as information in order to encapsulate or embody this knowledge.

b) The perception of information in order to acquire new knowledge.

The encoded knowledge takes the form of an understanding of the nature of the problem. On the other hand, the new knowledge is the knowledge obtained through a perception of the result of the processed information and should provide a solution to a given problem.

In general terms, the information provided to the combined logical-numerical framework - viewed here as the combination of a computational and a reasoning core - can be divided into two main groups: the static information of the problem and the dynamic information of the problem. The dynamic information is essentially based on site specific features. An important part of the dynamic information in this problem, is obtained through fully-dynamic simulations. This information is often processed by fuzzy heuristic rules or another equivalent knowledge-based technique*.

* Such as - for example - the learning related techniques.
6.2 Data specification. Trigger values

6.2.1 Specification of the dynamic information for the intelligent agent

As described in section 5.9, the combined logical-numerical, real-time optimisation process begins when the main monitoring loop sends a signal to program KBRTC indicating that the specified threshold values have been exceeded. In such a case, the program KBRTC activates - through the Application Programming Interface - the knowledge base HYDNET and starts the knowledge processing through networks of rules which trigger one-another in order to fulfil certain goals (hypotheses).

One of the first steps performed by the knowledge base HYDNET.kb is related to the retrieval of the input information contained in a number of NEXPERT OBJECT databases: the .nxp[d] files.

The main .nxp[d] files required by the knowledge base are:

a) 'network.nxp[d]'  
b) 'regulators.nxp[d]'  
c) 'timestr.nxp[d]'  
d) 'thresholds.nxp[d]'  
e) 'sensors.nxp[d]'

A detailed explanation about each of these NEXPERT OBJECT flat file databases is given below.

a) The file 'network.nxp[d]'

The file 'network.nxp[d]' contains relevant dynamic information about the global network's configuration. This information is utilized internally in order to specify the number of dynamic objects which will be generated in each class. An example of the typical structure of this file is:

<table>
<thead>
<tr>
<th>name</th>
<th>node_nr</th>
<th>pipe_nr</th>
<th>regul_nr</th>
<th>sensor_nr</th>
<th>ev_hor</th>
<th>timestep</th>
<th>saverate</th>
<th>ntimes</th>
<th>DefStrat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goteborg</td>
<td>78</td>
<td>72</td>
<td>4</td>
<td>8</td>
<td>720</td>
<td>60</td>
<td>30</td>
<td>6</td>
<td>OPEN</td>
</tr>
</tbody>
</table>

b) The file 'regulators.nxp[d]'

The file 'regulators.nxp[d]' contains relevant information about the main flow regulators in the network as well as the physical links among them. These links must be revealed through an off-line calibration process. This process will be explained in detail in section 6.3. We will just state now that this information allows the knowledge base to take into account such phenomena as the 'spatial interactions between regulators' which have been described in Chapter 2. The typical structure of this file is:
The Real-Time Control Process

<table>
<thead>
<tr>
<th>Nr</th>
<th>RegName</th>
<th>RegType</th>
<th>RegLB</th>
<th>RegUB</th>
<th>RPrior</th>
<th>LkReg1</th>
<th>RelPos1</th>
<th>LkReg2</th>
<th>RelPos2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A3</td>
<td>Gate</td>
<td>-1.37</td>
<td>-0.37</td>
<td>0</td>
<td>None</td>
<td>Neutral</td>
<td>None</td>
<td>Neutral</td>
</tr>
<tr>
<td>2</td>
<td>NORSTR</td>
<td>Weir</td>
<td>4.25</td>
<td>5.00</td>
<td>1</td>
<td>None</td>
<td>Neutral</td>
<td>None</td>
<td>Neutral</td>
</tr>
<tr>
<td>3</td>
<td>OSTSTR</td>
<td>Weir</td>
<td>11.65</td>
<td>15.00</td>
<td>0</td>
<td>None</td>
<td>Neutral</td>
<td>None</td>
<td>Neutral</td>
</tr>
<tr>
<td>4</td>
<td>SYDSTR</td>
<td>Weir</td>
<td>7.96</td>
<td>10.00</td>
<td>1</td>
<td>None</td>
<td>Neutral</td>
<td>None</td>
<td>Neutral</td>
</tr>
</tbody>
</table>

c) The file 'timestr.nxpd'

The file 'timestr.nxpd' contains relevant information about the structure of the strategy employed for the discretisation of the time-dependent vectors (for example, the control strategies covering the event horizon). The information contained in this file allows the reproduction of the time structure of the dynamic control, that is, this information allows the creation of dynamic objects to contain relevant control parameters at different stages in the real-time control process. The last timelabel in this file must correspond to the estimated duration of the event horizon. An example of the typical structure of this file is:

<table>
<thead>
<tr>
<th>Nr</th>
<th>timelabel</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>120.00</td>
</tr>
<tr>
<td>3</td>
<td>300.00</td>
</tr>
<tr>
<td>4</td>
<td>420.00</td>
</tr>
<tr>
<td>5</td>
<td>600.00</td>
</tr>
<tr>
<td>6</td>
<td>720.00</td>
</tr>
</tbody>
</table>

d) The file 'thresholds.nxpd'

The file 'thresholds.nxpd' contains information about the main threshold values which characterise the main elements of the network. These 'thresholds' are used by the fuzzy heuristic rules in order to provide decision making criteria which are used by the intelligent agent. An example of the typical structure of this file is:

<table>
<thead>
<tr>
<th>intens_thr (R)</th>
<th>time_thr (R)</th>
<th>BTC (TP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.50</td>
<td>30.00</td>
<td>6.00</td>
</tr>
</tbody>
</table>

e) The file 'sensors.nxpd'

The file 'sensors.nxpd' contains relevant information about the main sensors composing the real-time control system. It must be observed that this presumes an implicit - underlying - association between sensors and keypoints: for the sake of the effectiveness of the real-time control, the sensors must - at least - be installed in those points selected as keypoints in the combined optimisation system. In other words, it is recommendable to choose as costpoints those nodes where sensors are installed. This ensures a reliable decision-making process.
This information is indeed essential for optimal representation: the success of the intelligent control process is conditioned by the quality of the information in this file. Notice that the flow regulators must also be 'declared' as sensors in this file. An example of the typical structure of this file is:

<table>
<thead>
<tr>
<th>NR</th>
<th>SNAME</th>
<th>STYPE</th>
<th>SLB</th>
<th>SUB</th>
<th>SDSP</th>
<th>LR1</th>
<th>RPOS1</th>
<th>LR2</th>
<th>RPOS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A1</td>
<td>Level</td>
<td>-2.50</td>
<td>2.00</td>
<td>1.01</td>
<td>A3</td>
<td>Neutral</td>
<td>None</td>
<td>Neutral</td>
</tr>
<tr>
<td>2</td>
<td>Nam9</td>
<td>Level</td>
<td>8.00</td>
<td>15.00</td>
<td>10.05</td>
<td>A3</td>
<td>Upstrm</td>
<td>None</td>
<td>Neutral</td>
</tr>
<tr>
<td>3</td>
<td>Nam10n</td>
<td>Level</td>
<td>5.00</td>
<td>15.00</td>
<td>9.50</td>
<td>Sydstr</td>
<td>Upstrm</td>
<td>A3</td>
<td>Dwnstr</td>
</tr>
<tr>
<td>4</td>
<td>Nam11</td>
<td>Level</td>
<td>12.25</td>
<td>15.00</td>
<td>13.20</td>
<td>Sydstr</td>
<td>Dwnstr</td>
<td>None</td>
<td>Neutral</td>
</tr>
<tr>
<td>5</td>
<td>Ryaps</td>
<td>Flow</td>
<td>0.00</td>
<td>20.00</td>
<td>5.00</td>
<td>A3</td>
<td>Upstrm</td>
<td>None</td>
<td>Neutral</td>
</tr>
<tr>
<td>6</td>
<td>Nm13bp</td>
<td>Flow</td>
<td>0.00</td>
<td>20.00</td>
<td>0.00</td>
<td>Oststr</td>
<td>Upstrm</td>
<td>A3</td>
<td>Neutral</td>
</tr>
<tr>
<td>7</td>
<td>Regikod</td>
<td>Flow</td>
<td>0.00</td>
<td>10.00</td>
<td>0.00</td>
<td>Norrstr</td>
<td>Upstrm</td>
<td>None</td>
<td>Neutral</td>
</tr>
<tr>
<td>8</td>
<td>Nm16jg</td>
<td>Flow</td>
<td>0.00</td>
<td>3.00</td>
<td>0.00</td>
<td>Oststr</td>
<td>Dwnstr</td>
<td>None</td>
<td>Neutral</td>
</tr>
<tr>
<td>9</td>
<td>A3</td>
<td>Level</td>
<td>-1.37</td>
<td>-0.37</td>
<td>-1.00</td>
<td>A3</td>
<td>Neutral</td>
<td>None</td>
<td>Neutral</td>
</tr>
<tr>
<td>10</td>
<td>Norrstr</td>
<td>Level</td>
<td>4.25</td>
<td>5.00</td>
<td>4.45</td>
<td>Norrstr</td>
<td>Neutral</td>
<td>A3</td>
<td>Dwnstr</td>
</tr>
<tr>
<td>11</td>
<td>Oststr</td>
<td>Level</td>
<td>11.65</td>
<td>15.00</td>
<td>13.50</td>
<td>Oststr</td>
<td>Neutral</td>
<td>A3</td>
<td>Dwnstr</td>
</tr>
<tr>
<td>12</td>
<td>Sydstr</td>
<td>Level</td>
<td>7.96</td>
<td>10.00</td>
<td>9.50</td>
<td>Sydstr</td>
<td>Neutral</td>
<td>A3</td>
<td>Dwnstr</td>
</tr>
</tbody>
</table>

The general data specifying the parameters for the 'key' hydrodynamic simulations performed during the stage of logical optimisation must be included in the file 'KBRTC.DAT' (standard ascii file). An example of the typical structure of this file is:

```
GOTEBOE.\GENT  AVL.nxpd
720.00         60.00    30     1
720.00         0.00     4      15
5              6       6

0.00          120.00   300.00  420.00  600.00  720.00
```

Description of parameters in the file 'KBRTC.DAT':

A.- netfilename, B.- nxpd file to contain available volumes per branches, C.- event horizon (min), D.- timestep (sec), E.- saverate, F.- hotstart switch (1 → Yes, 0 → No), G.- hotstart time in .PRF file (min), H.- hotstart time in .RRF file (min), I.- number of regulators, J.- number of main branches in the network, K.- number of stages for discretisation of time-dependent vectors, and L.- timelabels of each of the stages of discretisation (allows strategy refinements).
The information about main branches in the system which allows the construction of the reachability and connectivity matrices according to the criteria discussed in section 5.8.3 must be specified in the file 'HYDNET.DAT'. The information in this file is of central importance to the work performed by the intelligent agent and therefore must be carefully specified. An example of the typical structure of this file is:

<table>
<thead>
<tr>
<th>Bnr</th>
<th>Fr_nod</th>
<th>To_nod</th>
<th>minX</th>
<th>maxX</th>
<th>minY</th>
<th>maxY</th>
<th>Init_Vol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NAM1</td>
<td>NORSTR</td>
<td>39956.5</td>
<td>40451.4</td>
<td>32184.8</td>
<td>43131.8</td>
<td>84523.0</td>
</tr>
<tr>
<td>2</td>
<td>NAM11</td>
<td>SYDSTR</td>
<td>39141.6</td>
<td>43154.2</td>
<td>17876.0</td>
<td>20523.6</td>
<td>25496.0</td>
</tr>
<tr>
<td>3</td>
<td>SYDSTR</td>
<td>A4</td>
<td>36500.0</td>
<td>39141.6</td>
<td>20523.6</td>
<td>25100.7</td>
<td>84390.0</td>
</tr>
<tr>
<td>4</td>
<td>NAM17</td>
<td>OSTSTR</td>
<td>42562.9</td>
<td>51825.8</td>
<td>23902.1</td>
<td>25736.4</td>
<td>78156.0</td>
</tr>
<tr>
<td>5</td>
<td>OSTSTR</td>
<td>A3</td>
<td>35780.8</td>
<td>42562.9</td>
<td>24295.0</td>
<td>28000.0</td>
<td>108736.0</td>
</tr>
<tr>
<td>6</td>
<td>NM16JG</td>
<td>OSTSTR</td>
<td>42000.0</td>
<td>43000.0</td>
<td>23000.0</td>
<td>24500.0</td>
<td>15350.0</td>
</tr>
</tbody>
</table>

Name of text_version_of_SWF_file: SWF.TXT

Description of parameters in the file 'HYDNET.DAT':

A - Number of branches, B - Total number of sensors for the runoff simulation, C - number of level sensors for pipeflow computations (in accordance with the '.NET' file), D - number of discharge sensors for pipeflow computations (in accordance with the '.NET' file), E - number of regulators, name of pipeflow ascii result file (extension '.dbg', usually 'pipe.dbg'), F - name of runoff ascii result file (extension '.dbg'), and [for all main branches in the system] -> G - specify parameters as indicated in figure above.

Further remarks

i) If the simulation is 'hotstarted', the data about initial volumes in the main branches of the network can be obtained from the summary facility of MOUSE corresponding to those scenarios which will be used for the hotstart.

ii) The values of 'minX', 'maxX', 'minY' and 'maxY' correspond to the 'frame recognition system' described in section 5.8.3.

iii) Care should be taken that the nodes specified in 'Fr_nod' and 'To_nod' in the definition of the main storage branches of the network do not overlap, i.e., that they are not repeated in the declaration. It must be remembered that the main storage branches in the network are only present upstream of the main flow regulators.

iv) The text (ascii) version of the '.SWF' file can be quickly and easily obtained by using the 'Goodies' facility in the catchment and pipe data section of MOUSE (menu 'A.8.2')
On the other hand, the fully non-linear, numerical optimisation system requires the specification of certain parameters for the main stages of the process. These parameters concern both the numerical optimisation process and the hydrodynamic simulations of MOUSE.

The main data files for the numerical optimisation methodology are:

i) The file 'exper.inp'
ii) The set of '.CFG' files
iii) The set of '.VTS' files
iv) The '.NET' file
v) the '.CTR' file

**Description of each input file**

i) The file 'exper.inp'

As explained above, this file contains important parameters for the main stages of the non-linear numerical optimisation process, namely the slope evaluation process and the single-line search. An example of the typical structure of this file is:

```
4 8 12 6 2 4 0.05 2.50 720.00
60.00 30 1 0.00 720.00
GOTEBORG.NET  GOTEBORG.CTR
-1.37 -0.37
4.25  5.00
11.65 15.00
7.96  10.00
```

Description of parameters in 'exper.inp':

A) - number of active flow regulators, B) - number of costpoints, C) - number of '.NET' sensors, D) - number of stages used for discretisation of the time-dependent vectors, E) - maximum number of optimisation cycles allowed, F) - number of 'experiments' to be located during the single-line search, G) - interval resolution (program TUNING), H) - A stopping criteria on the cost evaluation used to determine convergence (Cost Units), I) - duration of MOUSE simulations (min), J) - timestep (sec), K) - saverate, L) - hotstart switch (1 → Yes, 0 → No), M) - hotstart time in .RRF file (min), N) - hotstart time in .PRF file (min) and O) - operational boundaries (settings) for each of the active flow regulators.
ii) The set of '.CFG' files

The '.CFG' files are the files used for the definition of the economic model of the system. As argued in section 3.2, the results of the overall control process can only be as good as the economic model is a realistic representation of the complex phenomena which occur in these systems.

In this framework, the '.CFG' files are unique for each of the costpoints in the system. This means that the information contained in each of these files have a local scope: the overall economic model is obtained through mapping on the basis of fully-dynamic simulations. In principle these can be defined by functionals of any shape. They can even express the cost as a function of a different system state variable at each costpoint. This is one of the sources of generality and flexibility in this framework.

The '.CFG' files (one for each costpoint) are generated by the program CFGEN. As described in section 5.6.2, this program provides a menu of analytical functions for cost function generation which can be 'fine tuned' by the graphic visualisation/edition facility of CFGEN. This program supports two .CFG output formats: the detailed format with useful information about the parameters used in the analytical equation of the curve and the ready format which is directly supported in the optimisation framework.

A typical example of the files obtained after using program CFGEN is given below:

a) The ready output format of program CFGEN (used for cost evaluation):

In the first row we specify the number of coordinate pairs in the file and in the remaining rows we specify the pairs of values (SSV, C) where SSV represents the system state variable measured at the given costpoint and C represents its associated cost in the corresponding Cost Units (CU).

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1.200</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>1.600</td>
<td>0.030</td>
</tr>
<tr>
<td></td>
<td>2.000</td>
<td>0.066</td>
</tr>
<tr>
<td></td>
<td>2.400</td>
<td>0.108</td>
</tr>
<tr>
<td></td>
<td>2.800</td>
<td>0.155</td>
</tr>
<tr>
<td></td>
<td>3.200</td>
<td>0.209</td>
</tr>
<tr>
<td></td>
<td>3.600</td>
<td>0.268</td>
</tr>
<tr>
<td></td>
<td>4.000</td>
<td>0.333</td>
</tr>
<tr>
<td></td>
<td>4.399</td>
<td>0.406</td>
</tr>
<tr>
<td></td>
<td>4.799</td>
<td>0.485</td>
</tr>
<tr>
<td></td>
<td>5.199</td>
<td>0.571</td>
</tr>
<tr>
<td></td>
<td>5.599</td>
<td>0.666</td>
</tr>
<tr>
<td></td>
<td>5.999</td>
<td>0.768</td>
</tr>
<tr>
<td></td>
<td>6.399</td>
<td>0.880</td>
</tr>
<tr>
<td></td>
<td>6.799</td>
<td>1.000</td>
</tr>
</tbody>
</table>

The type of system state variable (h, Q, etc) is identified by this framework according to a keyword declared in the '.NET' file. For example, if the costpoint coincides with an overflow weir where we measure the overflow discharge, then we must include the keyword 'WeirFlow' in the definition of this costpoint (sensor) in the above mentioned '.NET' file.

Finally it is necessary to observe that the name of this ready format '.CFG' file indicates the costpoint where the cost function should be applied.
For example if the cost function name is 'CostCPI' the framework will assume that the given cost function must be used for costpoint number one. The order of the costpoints is also given in the 'sensors section' of the '.NET' file.

b) The detailed output format of program CFGEN

As explained above, the 'detailed output format' is necessary because it provides important information about the set of analytical parameters used for the generation of the cost curve. These define a first approximation to the cost function which can be used as an important reference and therefore should be stored. The typical structure of the detailed output format of program CFGEN is given below:

```
Cost Function Table Generated by -> Program CFGEN.

Output File Name -> TEST_L.CFG
Equation -> y = (a.e^(a*b))/(e)
Number of segments used for generation: 1
Number of Points (All segments): 15
Coordinates of Points -> (X, Y):
1.200  0.000
1.600  0.030
2.000  0.066
2.400  0.108
2.800  0.155
3.200  0.209
3.600  0.268
4.000  0.333
4.399  0.406
4.799  0.485
5.199  0.571
5.599  0.666
5.999  0.768
6.399  0.880
6.799  1.000

[WARNING : If nseg > 1 the data below corresponds to the last segment].

Coefficients for the equation (a, b, c, d, f, n) [N.U -> Not Used] :
a: Comp.  b: 0.75  c: N.U  d: N.U  e: N.U  f: N.U

Range for the independent variable (xmin, xmax):
1.2      6.799

Resolution:
0.399929
```
This file is saved by program CFGEN with the *same name* as that of the file in ready `.CFG` format plus the characters '_L'. For example if the filename specified to program CFGEN is 'CostCP1.CFG' the detailed output format will be sent to the file 'CstCP1_L.CFG'.

*iii) The set of '.VTS' files*

The '.VTS' files express the variation of the cost function with time. This is an important feature which can be used to provide a more complete representation of the complex processes occurring during real-time control of urban drainage systems. These files could be used - for example - to indicate that the water stored in the system must be released towards the end of the rain event in order to provide free capacity when a rain event is expected rapidly after the event that is currently being controlled. As explained in section 5.6.3, they provide a powerful representation model. The use of 'non-default' '.VTS' files affects the cost integration process over the event horizon.

In this file we simply define *time-multipliers* in different stages of the real-time control process. *By using time variation functions we include an imaginary 'time axis' in the hyperspace upon which the economic model is mapped.* An example of the typical structure of the .VTS files is:

<table>
<thead>
<tr>
<th>Time</th>
<th>Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1.000</td>
</tr>
<tr>
<td>120.00</td>
<td>1.000</td>
</tr>
<tr>
<td>300.00</td>
<td>1.000</td>
</tr>
<tr>
<td>420.00</td>
<td>1.150</td>
</tr>
<tr>
<td>600.00</td>
<td>1.700</td>
</tr>
<tr>
<td>720.00</td>
<td>2.100</td>
</tr>
</tbody>
</table>

The maximum number of stages which is allowed in the '.VTS' files corresponds to the number of stages defined in the discretisation of the control strategies.

Similarly to the '.CFG' files, the association to a given costpoint is indicated by its name: for example the file 'tmultCP5.VTS' will be associated to the fifth sensor encountered in the sensors and regulators section of the '.NET' file.

*iv) The '.NET' file*

The '.NET' file comprises relevant information about the system's network. This information is used by module PIPFLW in order to define control strategies to be used in the MOUSE HD simulations and also in order to extract time series from the '.PRF' file of MOUSE. Among other functions, the information contained in the '.NET' file indicates to the optimiser the points in the staggered grid whose time series must be stored in the '.dbg' file. This file is also used by the MOUSE SIMULATOR'S USER INTERFACE in order to perform a plotting of the combined network. In general, the '.NET' file defines the following parameters:

* The structure of the cost network in the system.
* Order of costpoints and regulators.
* Type and association of sensors to costpoints
Whatever represents input to MOUSE (rain, control strategies) is considered as regulators by the ".NET" file. Therefore, the rain gauges are also to be regarded as regulators. The information in the ".NET" file is divided into the following sections:

- Information about nodes (nr, coordinates, type, name).
- Information about pipes (nr, from -> to, nr of grid points, type).
- Information about pumps (nr, node associated, from -> to).
- Information about overflow weirs (nr, node assoc., from -> to).
- Information about sensors (nr, coordinates, min. value, max. value, type, from -> to, grid nr., nr. in set).
- Information about regulators (nr, coordinates, min. value, max. value, type, from -> to, grid nr).

Due to its considerable extent we will only offer a fragment of the ".NET" file: the 'sensors' and the 'regulators' sections by way of examples.

```
# Start of ROSA data set ...

#   #   Xc   Yc   MinV   MaxV   Keyword  From   -> To   GN   SN

Sensors
1  35780.80  25994.70  -1.37  15.00  NodeLevel A1
2  33628.80  25001.60   8.00  15.00  NodeLevel NAM9
3  36920.00  21093.70   0.00  15.00  NodeLevel NAM10N
4  43154.20  17876.00  12.00  20.00  NodeLevel NAM11
5  34931.70  27069.60   0.00  20.00  PumpFlow  RYAPS -> RYABF 0 1
6  37617.70  27330.70   0.00  30.00  WeirFlow NAM13BP -> 0 0 1
7  39890.60  30382.80   0.00  10.00  WeirFlow RELKOD -> 0 0 1
8  42476.90  23228.00   0.00   5.00  WeirFlow NAM16JG -> 0 0 1
9  35780.80  25994.70  -1.37  -0.37  WeirLevel A3 -> A1 0 1
10 38956.50  32184.80   4.25   5.00  WeirLevel NORRSTR -> R5 0 1
11 42562.90  24295.00  11.65  15.00  WeirLevel SYDSTR -> K0 0 1
12 39141.60  20523.60   7.96  10.00  WeirLevel SYDSTR -> KNAPEG 0 1

Regulators
1  34871.70  27089.60   0.00  50.00  RainGauge  RYARV
2  35780.80  25994.70  -1.37  -0.37  WeirPosit A3 -> A1 1
3  38956.50  32184.80   4.25   5.00  WeirPosit NORRSTR -> R5 1
4  42562.90  24295.00  11.65  15.00  WeirPosit SYDSTR -> K0 1
5  39141.60  20523.60   7.96  10.00  WeirPosit SYDSTR -> KNAPEG 1
```

Description of parameters in the ".NET" file:

- **Xc** : X coordinate of the node
- **Yc** : Y coordinate of the node
- **MinV** : Minimum setpoint (in the case of regulators this is specified as settings)
- **MaxV** : Maximum setpoint (in the case of regulators this is specified as settings)
- **Keyword** : Indicates the type of sensor (also the system state variable observed)
- **GN** : Grid number
- **SN** : Number in set (MOUSE)
v) the '.CTR' file

The '.CTR' file contains the definition of the initial control strategies for the main flow regulators in the network. These control strategies must cover the whole event horizon. The '.CTR' file defines a time resolution (also called here a discretisation pattern) for the control strategies. This means that the modifications to such strategies can only be performed at these stages. The number of stages specified here directly determines (together with the number of flow regulators) the size of the solution array assembled by the Quasi-Newton optimiser. These are square matrices of order \( n_r \times n_r \), where \( n_r \) represents the number of flow regulators and \( n_r \) the number of stages used in the discretisation of the time dependent vectors.

An example of the typical structure of the '.CTR' files is:

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>5.0</th>
<th>6.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0</td>
<td>0.00</td>
<td>-0.93</td>
</tr>
<tr>
<td>120.0</td>
<td>0.00</td>
<td>-0.87</td>
</tr>
<tr>
<td>300.0</td>
<td>0.00</td>
<td>-1.02</td>
</tr>
<tr>
<td>420.0</td>
<td>0.00</td>
<td>-0.85</td>
</tr>
<tr>
<td>600.0</td>
<td>0.00</td>
<td>-0.78</td>
</tr>
<tr>
<td>720.0</td>
<td>0.00</td>
<td>-0.76</td>
</tr>
<tr>
<td>12.77</td>
<td>4.38</td>
<td>7.96</td>
</tr>
<tr>
<td>12.77</td>
<td>4.38</td>
<td>8.17</td>
</tr>
<tr>
<td>12.77</td>
<td>4.38</td>
<td>8.08</td>
</tr>
<tr>
<td>12.77</td>
<td>4.38</td>
<td>8.17</td>
</tr>
<tr>
<td>12.77</td>
<td>4.38</td>
<td>8.18</td>
</tr>
</tbody>
</table>

### 6.2.2 The set of underlying relationship functions

The combined logical-numerical framework requires a special kind of 'information' about the domain which can only be obtained through model-based practice. This 'information' is understood as the informational representation of that part of the knowledge about the system which cannot be encoded and indeed cannot even be user-specified: it can only be acquired through modelling based on fully dynamic simulations performed with 'key control strategies' (model-based control).

The contents of this section are closely related to the issues discussed in sections 4.3, 4.4, 4.7 and 5.8. Although it has not been explicitly mentioned, it is apparent that the classification between 'dynamic' (which includes the set of underlying relationship functions) and 'static' information is performed from the level of the domain. In this study, 'static' information is regarded as that information which is essentially based on general properties for the domain.

Fuzzy heuristic/diagnostic rules provide an effective 'environment' to represent this knowledge in informational form. In effect, these rules encode fuzzy linguistic terms which are interpreted on the basis of reasoning about the ongoing situation, and therefore they possess a general range within the domain.

As suggested above, the knowledge base supplied in this combined framework, provides a number of key control strategies which are tested in order to correct potential undesirable effects. The quasi-optimal control strategy is then approximated on the basis of reasoning
about the causes of the system's misbehaviour in a particular scenario. This process has been
designed to be performed in the minimum possible time.

6.3 Revealing internal links

The purpose of this section is to outline the importance of a proper specification of internal
links in the system controlled in real-time. It will subsequently be argued that an effective
knowledge-based control of urban drainage networks can only be achieved through a proper
understanding of these internal physical links. These links are a characteristic of the network:
they are always 'present' although they may or may not become evident in a particular real-
life situation.

However, the actual understanding of these links can be realised in different ways. Indeed, this
depends on the particular technique which is used for problem solving. In the most general
terms, the understanding of these relationships is achieved through modelling.

An important part of the complexity of the problem of real-time control of urban drainage
networks originates from the fact that the input to these systems possesses a stochastic nature.
This is essentially 'unpredictable' and can only be represented with a greater or lesser
accuracy.

In the present (combined) optimisation framework, the basic understanding of the functioning
of the hydrodynamic network occur in two phases which are related to an 'off-line calibration
process' (performed by the system analyst) and the 'on-line learning process' described in
section 5.8.2.

In this section we will depict the 'off-line calibration process' since the 'on-line learning
process' has already been discussed. This 'calibration' process is performed on the basis of
recorded (earlier) events, preferably occurring in the area where the hydrodynamic network
is located. This process must be performed off-line, that is, by using events occurred in the
past without the restrictions imposed by the real-time constraint.

The main objectives of such a process are the construction of the economic model for the
problem and the specification of internal links in the system such as for example the
interaction among flow regulators or between sensors and regulators. This information is
stored in the '.nxpd' files.

This 'off-line calibration process' consists of a testing of the system with several events and
control strategies. The results of such an analysis - based on fully dynamic simulations - are
empirical descriptions of the underlying relationships between the nodes of the so-called
'information subnetwork'.

From this analysis we can observe (and describe), for example, the level of influence of a
certain regulator at the nodes of the information sub-network around it. We can also describe
interactions among regulators and the relative importance of each regulator in relation to the
increase in the storage capacity it causes when it is operated. This descriptions are depicted in the files 'regulators.nxpd' and 'sensors.nxpd'.

6.4 The choice of the 'kind' of optimisation: the real-time constraint

It is the purpose of this section to provide criteria for the selection of the 'scope' of the combined logical-numerical optimisation process in relation to the given real-time control problem that has to be solved and the time available to effect this control.

Real-time control is performed if the data about the ongoing process is utilised in order to improve its performance in terms of well define criteria. We have argued that effective real-time control is related not only to the use of on-line information about the state of the system, but also to the proper use of flow forecasts. As stated by Tan et al (1990), a forecast of the flow situation improves the control since the control action is taken not only on the basis of the actual situation but also taking into account the expected situation.

One of the advantageous features of the present hybrid framework is the possibility of choosing among several kinds of control 'schemes'. This means that we can specify the characteristics of the control process in relation to the characteristics of the system and the system's input (the stochastic rain event).

Basically, the kinds of control 'schemes' supported by this framework are explicitly related to its main functional elements. Therefore, the process to be performed can be expressed in terms of:

i) A strictly logical kind of optimisation.

ii) A strictly numerical, non-linear kind of optimisation*

iii) A combined logical-numerical kind of optimisation

The most effective type of optimisation is unquestionably iii). However, there may be situations in which it might be necessary (or feasible, depending on the case) to perform one or both of the other two types of optimisation.

Just to mention two possible scenarios, let us imagine the situation of a large network with a considerable number of flow regulators which must be continuously controlled over a long period of time.

* This is possible in those systems where the number of flow controllers is relatively small or where only the more important regulators - in relation to their influence on the storage capacity in the system - are globally controlled by the numerical optimiser. In the later case the less important regulators are locally controlled by making use of a proportional integral-derivative (PID) - or similar - type of control.
In this case a strictly numerical, non-linear kind of optimisation, or even a combined logical-numerical kind of optimisation could be computationally too demanding in real-time. In this case it would probably be necessary to perform a strictly logical kind of optimisation. An alternative way of approaching this problem is described by Wilson (1995). The opposite example is found in a medium-to-small scale network with a relatively small number of (globally controlled) main flow regulators in which it would be feasible to perform the types of optimisation i) and ii).

Of course, as discussed in section 5.5, the rapid development of the hardware which is taking place nowadays make it possible to predict that within a relatively short period of time, it will be possible to perform the combined logical-numerical type of optimisation (or even exhaustive numerical search) in the great majority of cases. Parallel and Vector computers (already available in certain configurations) are a good example of the above development.

Even within the present computational environment, however, due to the features incorporated in the present combined logical-numerical framework, our goal of implementing hybrid optimisation in a relatively large number of real life situations is already within reach.

6.5 The chaining process

The purpose of this section is to discuss the technical aspects of the combined logical-numerical kind of optimisation. The more convenient arrangement of factors in this hybrid system was found to be a).- A logical optimisation step and b).- A non-linear, numerical optimisation step.

As happens during the development of most hybrid systems with these characteristics, additional technical problems must be solved in order to facilitate the 'chaining process' between two essentially different approaches so as to suit each of them to the solution of the combined problem. This section will address the way of specifying parameters so that this process can take place in a successful manner.

The default mode in the combined logical-numerical optimiser is the 'detached mode' in which the optimisation process starts with a logical optimisation step which is triggered by a signal from the data acquisition system (SCADA).

This causes a call to the main logical optimisation centre (program KBRTC) which takes care of the whole logical optimisation step. If the input data have been properly specified, the intelligent agent rapidly produces a series of better vectors in relation to the default situation in the network (no control).

In default mode, these resulting vectors (contained in a number of ready format files) must be supplied by the user to the numerical optimiser. This module is activated by typing the command 'exper' from the command line.

However, the automated way of performing the 'chaining' process is also supported by the combined logical-numerical framework. For the sake of flexibility, this is done by using a
UNIX system 'batch' file specifying the parameters to transfer to the elements which integrate
the hybrid framework. This 'batch' file is available in the directory containing the executable
files of the optimisation framework.

6.6 Output results

The output of the combined framework can be again separated - at least for methodological
purposes - into two main groups:

i) The output of the logical optimisation step

ii) The output of the numerical optimisation step

As the main modules of the combined logical-numerical optimiser are being 'triggered',
messages are sent to the screen to inform the user about the processes which are taking place.
Due to the large number of modules involved, in the case of the intelligent agent it is rather
difficult to summarize the sequence of steps taken. However, they correspond to the sequence
described in section 5.6, 5.8 and 5.9. In this section we are rather concerned with the
description of the output results supplied in each of the main steps of this process.

i) The output of the logical optimisation step

As explained in section 5.2, one of the main functions of the intelligent agent is to provide
the initial search vectors for the fully non-linear optimisation system. In this way, the
intelligent agent effects a restriction of the search space which have proven to be practically
essential for real-time implementation.

This is achieved by performing a series of approximations to the quasi-optimal control
strategies which are contained in the following ready format files (they can be used directly
by the numerical optimisation system):

a) 'INTERPOL_RD.CTR'
b) 'SHAPED_RD.CTR'
c) 'KBSTR_RD.CTR'

The file 'INTERPOL_RD.CTR' contains a first 'intelligent' approximation to the quasi-
optimal control strategies for the flow regulators. This approximation is constructed on the
basis of the idea of the mass balance in the system's subnetworks explained in section 5.8.4.
On the other hand, the file 'SHAPED_RD.CTR' provides a second 'intelligent' approximation
to the quasi-optimal control strategies for the flow regulators. This approximation is
constructed on the basis of the idea of the convergence to a desired behaviour of the system,
as explained in section 5.8.5.

Then, a final approximation to the quasi-optimal control strategies is performed on the basis
of the previous approximations, which are then rechecked by applying the results of the 'on-
line learning process' described in sections 4.7.1 and 5.8.2. This final approximation is
contained in the file 'KBSTR_RD.CTR'.
An example of the typical structure of these '.CTR' files, as derived by the knowledge base, is given below:

```
5 6
0.0 0.00 -0.93 4.38 12.77 7.96
120.0 0.00 -0.87 4.38 12.77 8.17
300.0 0.00 -1.02 4.31 12.77 7.99
420.0 0.00 -0.85 4.38 12.77 8.08
600.0 0.00 -0.78 4.38 12.77 8.17
720.0 0.00 -0.76 4.38 12.77 8.18
```

Another main task performed by the intelligent agent is to provide a first estimate of the matrix of second order partial derivatives (the Hessian matrix). This matrix contains information about the curvature of the search space in the neighbourhood of the initial vectors that are supplied. This information is provided by the intelligent agent in what is really a qualitative way, that is, in the way of linguistic labels indicating the most likely descent directions expected from the initial vectors. This information is contained in the file 'REQUESTS.NXP' (of which due to its extent only a fragment is supplied).

As observed, there are again three main possibilities from a given (default) position:

a) The search is directed upwards, meaning that a better setpoint is expected between the current position and the corresponding upper boundary.

b) The search is directed downwards, meaning that a better setpoint is expected between the current position and the corresponding lower boundary.

c) The current position is maintained.

It is important to explain that the present estimate of the gradient matrix does not replace the numerical slope evaluation process, but it indicates the direction in which the perturbation should be taken in order to obtain a suitable descent direction.

ii) The output of the numerical optimisation step

*The characters '_RD' attached to the '.CTR' file name indicate that the file is in ready format. The distinction is necessary because a detailed format of these files is also provided by the knowledge base.*
The output of the *numerical* optimisation step is stored in a certain number of formatted ASCII files. The more important are:

1. **The n 'PIPEnCYC.DBG' files**
2. **The n x m '.TSF' files**
3. **The n x m '.CFC' files**
4. **The n 'SOLnCYC.CTR' files**
5. **The n 'COSTnCYC' files**
6. **The n 'PRFnCYC.PRF' files**

where *n* represents the number of optimisation cycles and *m* the number of costpoints.

The time used for the computations is stored in an ASCII file called 'opt_time.TIM'. It must be pointed that more than 95% of this time is employed by the HD simulations performed.

1. **The n 'PIPEnCYC.DBG' files**

The file 'pipe.dbg' contains the time series of the system state variable (h or q) specified at each of the so-called '.NET' sensors. This information is stored in the '.NET' file.

Although this has been already explained, it should be emphasized that the concept of a '.NET' sensors is used to designate whatever represent 'output' from the MOUSE ONLINE model, while the '.NET' regulators indicate everything that represents input for the MOUSE ONLINE system (including the rain gauges, for example).
The combined optimisation framework stores the relevant '.DBG' files under the name 'PIPECYC.DBG', where n stands for the number of the cycle. Observe that, in this case a number of 6 '.NET' sensors was specified. These are:

1. 'B4.1500' (Level Sensor).
2. 'B4.1490' (Level Sensor).
3. 'B4.1491' (Level Sensor).
4. 'B4.1480' (Flow Sensor).
5. 'B4.1500' (Regulator #1).
6. 'B4.1490' (Regulator #2).

A typical example of the structure of the '.DBG' files is given below:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>16.310</td>
<td>16.170</td>
<td>16.350</td>
<td>0.000</td>
<td>16.290</td>
<td>16.150</td>
</tr>
<tr>
<td>2.000</td>
<td>16.310</td>
<td>16.170</td>
<td>16.350</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
<tr>
<td>4.000</td>
<td>16.370</td>
<td>16.350</td>
<td>16.350</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
<tr>
<td>6.000</td>
<td>16.470</td>
<td>16.540</td>
<td>16.350</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
<tr>
<td>8.000</td>
<td>16.540</td>
<td>16.710</td>
<td>16.350</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
<tr>
<td>10.000</td>
<td>16.700</td>
<td>16.850</td>
<td>16.350</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
<tr>
<td>12.000</td>
<td>16.970</td>
<td>16.970</td>
<td>16.367</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
<tr>
<td>14.000</td>
<td>17.233</td>
<td>17.100</td>
<td>16.530</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
<tr>
<td>16.000</td>
<td>18.480</td>
<td>17.230</td>
<td>16.730</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
<tr>
<td>18.000</td>
<td>19.090</td>
<td>17.340</td>
<td>16.920</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
<tr>
<td>20.000</td>
<td>19.340</td>
<td>17.370</td>
<td>17.073</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
<tr>
<td>22.000</td>
<td>19.410</td>
<td>17.340</td>
<td>17.180</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
<tr>
<td>24.000</td>
<td>19.400</td>
<td>17.300</td>
<td>17.230</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
<tr>
<td>26.000</td>
<td>19.410</td>
<td>17.370</td>
<td>17.238</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
<tr>
<td>28.000</td>
<td>20.080</td>
<td>17.480</td>
<td>17.680</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
<tr>
<td>30.000</td>
<td>20.180</td>
<td>19.720</td>
<td>18.760</td>
<td>0.000</td>
<td>16.300</td>
<td>16.160</td>
</tr>
</tbody>
</table>

In the case of 'B4.1490' and 'B4.1500' there is superposition of a sensor and a regulator at the same node. This is perfectly possible. The cases are distinguished by a keyword in the '.NET' file.

2. The n x m '.TSF' files.

The '.TSF' files represent an intermediate step taken by the numerical optimiser, starting out from the results of the pipeflow model of MOUSE in order to realise the process of cost evaluation, calculation and integration. These files are also used to realise the hotstart facility used by the numerical optimiser. They basically contain the same information as the 'pipe.dbg' file, but now classified and separated by costpoints. The information is ready at this stage to be supplied to the program CFCALC, where it will be processed together with the information from '.CFG' and '.VTS' files. The n '.TSF' files are written by the program LEAD.
3. **The n x m '.CFC' files**

The '.CFC' files summarize the results of the cost evaluation process at every costpoint. These are ASCII formatted files with enough information about the process to make it self-explanatory. Each of the '.CFC' files contains a 'cost time series' associated with the costpoint. This cost time series is the result of the *hypersurface conversion process*, explained in section 5.3. They are internal files used by the combined framework during the optimisation process.

4. **The n 'SOLnCYC.CTR' files**

The structure of the files containing the quasi-optimal control strategies ('.CTR' files) for the flow regulators have been discussed earlier in this section.

5. **The n 'COSTnCYC' files**

The cost file summarizes only the global results of the process of cost evaluation, calculation and integration. This is, the global cost obtained after integration of the cost time series over the whole forecasted horizon associated with each of the costpoints. Also the total cost of the control strategy for the system, during the whole period of covered by the simulations, can be represented in this way.

This file is created after every step taken by the mathematical optimizer during the process of slope evaluation or during every experiment located in the single-line search. An example of such a 'COSTFILE' is given in figure 5. It must be remembered that the number which identifies each of the costpoints corresponds to the order specified in the '.NET' file.

```
Cost_of_the_control_strategy_at_CP#1 : 4.082E+06
Cost_of_the_control_strategy_at_CP#2 : 2.737E+08
Cost_of_the_control_strategy_at_CP#3 : 1.707E+09
Cost_of_the_control_strategy_at_CP#4 : 4.515E+06
Cost_of_the_control_strategy_at_CP#5 : 5.932E+08
Cost_of_the_control_strategy_at_CP#6 : 4.344E+08
Cost_of_the_control_strategy_at_CP#7 : 7.229E+08
Cost_of_the_control_strategy_at_CP#8 : 3.911E+06
TOTAL_COST_OF_THE_CONTROL_STRATEGY : 3.744E+09
```

6. **The n 'PIPEnCYC.PRF' files**

The '.PRF' files are the most detailed source of information provided during the optimisation process. These are binary files created by the MOUSE pipeflow models containing time series of the system state variables at all nodes in the network.

The most relevant '.PRF' files - those simulations corresponding to the quasi-optimal control strategies at the end of each main numerical optimisation cycle - are stored by the optimisation
framework under the name 'PRFnCYC.PRF', where \( n \) indicates the number identification of the cycle. These files are written by the MOUSE pipeflow models in binary format. For further details about the structure of the '.PRF' files, reference is made to the MOUSE documentation (1992).

These files provide one of the most complete sources for evaluating the performance of the numerical optimiser in terms of well-defined flow criteria.

As indicated above, the user is informed about the ongoing processes during the numerical optimisation process. During the stage of numerical optimisation, a summary of this list of processes is sent to the file 'SUMMARY'. This improves the traceability of the numerical optimisation process. An example of the typical structure of this file is given below:

```
+------------------------------------------
| Quasi_Newton search terminated because of | Nr ITERATIONS.
| Number of completed updating cycles      | 1
| No. of cost function evaluations         | 31
| Time series of h/Q stored in files       | PIPEnCYC.DBG
| Relevant .PRF files (pipeflow) stored in | PRFnCYC.PRF
| Optimal control strategies in files      | SOLnCYC.CTR
| Results of cost evaluations in files     | COSTnCYC
| Initial control strategy in file         | FIRST.CTR
| Cost of the initial CS in file           | COST_FST
| Time used for the optimization in file   | opt_time.TIM

NOTE: \( n \) stands for the number of the cycle.
```

So far we have provided the main criteria which must be observed when specifying the data during the combined logical-numerical optimisation process.

Due to its extent, it is not feasible to provide coverage of all the details of the process. However, what we have described appears to cover the most important aspects that must be taken into account in order to ensure the successful performance of the present framework.
Chapter 7    Applications

7.1    Introduction

The present version of the Combined Logical-Numerical Framework (CLNF) has been so far tested in relation to the criteria listed in Chapter 1 with three urban drainage system models including the model of the real world urban drainage network of the city of Gothenburg, Sweden. It is believed that these tests are sufficient to demonstrate the potential of the present framework to deal with real-time control in relation to the criteria listed in section 1.3 within the hydroinformatic point of view.

The three examples chosen have different characteristics. Indeed, a certain number of distinguishing (or rather special) features was sought in each of these. The first two have a rather academic purpose: they were developed in the initial stages of the research when only the fully non-linear, numerical optimisation system was available. The main conclusion which may be extracted from these is that they demonstrate that the present optimisation system is actually capable of performing a 'quasi-optimisation' in those cases where the volume of regulated flows represents a rather significant percentage of the total volume of water which circulates through the system for a given rain event. In the real world these situations often arise for rain events of relatively short return periods, although these are not frequently controlled.

Let us think of those cases of systems with relatively large storage capacity in which the real-time control is performed on rain events of relatively short return periods. In these cases, the volumes of non-regulated flows can actually be small since they are basically contained by the network. Of course, in these cases the volumes of combined sewer overflows and surface flooding are usually small (if present at all), but then again it is in principle possible to perform control aimed at flow equalisation for better working conditions at the treatment plant - just to mention an example.

It is clear that at the moment the two main objectives sought with real-time control are the reduction of combined sewer overflows and/or the reduction of surface flooding. These objectives are essentially conflicting in those situations where they appear simultaneously, as explained in section 2.1.

7.2    EXAMPLE Nr. 1: "A LOOPED SYSTEM"

The main objective of this example was - in addition to what was explained in section 7.1 - to test the performance of the combined logical-numerical optimisation framework when dealing with systems where hydraulically complex phenomena could appear. In this case the phenomenon we are referring to is the 'loop effect' described in section 2.1. For this example an extreme synthetic rain event was selected in order to create a scenario in which both of the aforementioned
undesirable effects (surface flooding and combined sewer overflows) would appear simultaneously for the situation of no control. This was done as well as for testing the robustness of the numerical methods employed. The scenario created with this extreme rain, however, might well appear in real-world applications with relatively small events for those cases in which the system contains a high initial volume of water (for example, two consecutive rain events, when there is no time for implementing a previous drawing down of the water levels as a part of the optimisation process). The "LOOPED" urban drainage system has a surface of approximately 7.5 km² (2.5 by 3 Km). The diameter of the pipes increases (towards its downstream side) from 1 to 1.5 m. When aggregated over all pipes, manholes and structures, the available storage capacity is approximately 6000 m³. The rain enters the system at four points (where the rain sensors are installed). These points are associated with:

1. Catchment in 'B4.1500', with a surface of 2.1 ha, 110 inhabitants/ha, and 60% impervious area.
2. Catchment in 'B4.1502', with a surface of 8.54 ha, 60 inhabitants/ha, and 50% impervious area.
3. Catchment in 'B4.1490', with a surface of 10.20 ha, 140 inhabitants/ha, and 75% impervious area.
4. Catchment in 'B4.1511', with a surface of 7.345 ha, 100 inhabitants/ha, and 70% impervious area. The total area contributing to surface runoff is then 18.32 ha.

These again have been chosen to provide an extreme (although barely realistic) situation. The rain hydrograph used for real-time control in this system is illustrated in figure 42.

![Figure 42: The rain event used on the 'LOOPED' system](image)

This simplified system which was used as the starting point for testing the framework, is composed of twelve nodes, two overflow weirs, nine pipes and eight sensors (composed of four rain sensors, three level sensors and one flow sensor). There are three structures (at the nodes 'B4.1491', 'B4.1480' and 'B4.1510'). Two pumps are installed in the system, one pumping from
'B4.1510' to 'B4.1511' and the other one pumping from the structure at 'B4.1480' to the treatment plant. When the inflows arriving to 'B4.1480' exceed the capacity of this pump untreated overflows start to be produced through the outlet situated at 'A0.0327'. The magnitude of this flow is measured by the flow sensor located in the pipe from 'B4.1480' to 'A0.0327'.

Due to its small size, only four costpoints were selected for representation. These costpoints are located at the following nodes:

1. 'B4.1500' (Coinciding with level sensor # 1).
2. 'B4.1490' (Coinciding with level sensor # 2).
3. 'B4.1491' (Coinciding with level sensor # 3).
4. 'B4.1480' (Coinciding with flow sensor # 1).

There are also two regulators (underflow gates) situated at the nodes 'B4.1500' and 'B4.1490'. It should be observed that no spatial coincidence is required between costpoints and regulators. As often occurs, the number of costpoints is greater than the number of regulators. A plan view (schematic representation) of the 'LOOPED' system is provided in figure 43.

![Figure 43: A plan view of the 'LOOPED' system](image)

The operational ranges (physical boundaries) for the flow regulators are:

a) 'B4.1500' £ Between 16.29 m and 17.54 m.
b) 'B4.1490' £ Between 16.15 m and 17.65 m.
It should be observed that this range coincides with the diameter of the corresponding pipe (1.25 m and 1.50 m, respectively).

The cost functions connected to each of the costpoints (in the order stated before) are shown in figure 44. In these graphs we find: Water Levels (WL) or Discharges (Q) \( x \) axis against Cost (in Cost Units) \( y \) axis.

As explained in section 2.9, the construction of appropriate cost functions for the problem is a factor of central importance for the success of the optimization process. The selection of the appropriate number and location of the costpoints is also a factor to be taken into account.

The criteria used for the construction of the economic model in this example were provided by the objectives of the control. In effect, these cost functions were constructed with the purpose of 'driving' the optimiser to reduce water levels (W.L.) at the costpoints below the ground level (in order to avoid street overflows in these points) and at the same time to maintain these levels 'high enough' to prevent excessive untreated overflows through 'A0.0327'. This means that the storage capacity should be utilized as much as possible.

In other words, these cost functions introduce a conflict that has to be resolved by the optimizer. It is necessary to remember that we are dealing with conflicting objectives. This contradiction can be expressed as: excessive water levels will cause surface flooding and therefore must be avoided, but low water levels will cause excessive untreated overflows and must be avoided too. The preferred situation would be one in which the system is as full as possible without causing surface flooding.

The cost functions corresponding to each of the four costpoints selected for representation in the 'LOOPED' system were defined in the following order:

1. 'B4.1500' \([X\ \text{axis} \ \&\ \text{W.L. (m)}; \ Y\ \text{axis} \ \&\ \text{Reward (CU)}]\)
2. 'B4.1490' \([X\ \text{axis} \ \&\ \text{W.L. (m)}; \ Y\ \text{axis} \ \&\ \text{Reward (CU)}]\)
3. 'B4.1491' \([X\ \text{axis} \ \&\ \text{W.L. (m)}; \ Y\ \text{axis} \ \&\ \text{Reward (CU)}]\)
4. 'B4.1480' \([X\ \text{axis} \ \&\ Q (m^3/s); \ Y\ \text{axis} \ \&\ \text{Reward (CU)}]\)

where 'CU' stands for surrogate Cost Units. Three-segment, straight cost functions were selected for the sake of simplicity. They were found to provide an effective representation of the operational costs in this system.

The 'decision matrix' was composed in this case of (eight x eight) decision variables. A four-stage, time-shifted control strategy is to be evaluated for two main flow regulators, as described above.
Figure 44: Cost functions utilised for the real-time control of the 'LOOPED' system

7.2.1 Results and comments.

All the simulations were performed by using the runoff and pipeflow models of MOUSE without the representation of surface effects (i.e. surface flooding). The forecasted horizon (FH) was set to 80 minutes. This time is enough for a primarily complete representation of the advance of the inflow hydrograph through the system caused by the rain event described above, which lasted 44 minutes. A time step equal to ten seconds was used for the computations. The number of time steps between the saving of results was selected as twelve, in order to obtain a convenient time resolution (TR) for writing the results to the shared memory. A value of 0.058 was used as the fractional resolution based on the initial interval of uncertainty for the Fibonacci search.

A shared memory table (or temporary array) was displayed on the screen after every simulation. This array contained dynamic information about sensors (rain, runoff, level and flow sensors, in this order) and also the information about regulators (applied strategies).

The initial control strategy vectors for the numerical optimiser were defined according to the time resolution (two minutes) as a four-points-time-shifted control strategy (in principle it can be any number, although it must be taken into account that the size of the matrixes to be constructed is equal to the product of the number of regulators and the number of points in the strategy). The rest of the values were linearly interpolated by the optimizer. A total of \((\text{FH} / \text{TR}) + 1\) values were generated. The initial strategy for the optimization was defined - although only in this case - as regulators completely closed and pumps working according to a local default pumping strategy. According to the already defined surrogate cost functions the cost of initial control strategy would be:
In the reference situation - the non-regulated situation - the cost according to the above defined cost functions would be:

\[
\begin{align*}
\text{Cost of the control strategy at CP#1:} & \quad 1.620E+05 \\
\text{Cost of the control strategy at CP#2:} & \quad 0.006E+03 \\
\text{Cost of the control strategy at CP#3:} & \quad 1.657E+05 \\
\text{Cost of the control strategy at CP#4:} & \quad 0.000E+00 (*)
\end{align*}
\]

TOTAL COST OF THE CONTROL STRATEGY: 3.357E+05 C.U.

(*) No combined sewer overflows are produced in this system when all regulators are closed.

After one main updating cycle of the inverse of the Hessian matrix (one search for the sliding direction plus one single-line search), the optimisation framework constructed the vectors illustrated in Figure 45.

\[
\begin{align*}
\text{Cost of the control strategy at CP#1:} & \quad 0.870E+02 \\
\text{Cost of the control strategy at CP#2:} & \quad 0.881E+02 \\
\text{Cost of the control strategy at CP#3:} & \quad 0.911E+02 \\
\text{Cost of the control strategy at CP#4:} & \quad 5.183E+05
\end{align*}
\]

TOTAL COST OF THE CONTROL STRATEGY: 5.186E+05 C.U.

Figure 45: Vectors found as a solution after one main updating cycle of the Hessian matrix

According to the cost functions defined, the time-integrated global surrogate cost corresponding to this solution strategy is:
The quasi-optimal control vectors obtained by the optimisation framework after two main updating cycles to the inverse of the Hessian matrix are illustrated in Figure 46.

![Figure 46: Vectors found as a solution after two main updating cycles to the Hessian matrix containing information about the curvature of the search space](image)

The total surrogate cost associated with these vectors is only 7,609 x 10^4 Cost Units. The plots of the time series corresponding to the situations of regulators completely closed (initial strategy for this optimization process), regulators completely open (no regulation), and regulators controlled according to the quasi-optimal strategies obtained after one and two main updating cycles to the Hessian matrix are given below. They show the improvement of the flow situation not only in terms of cost, but also according to well-defined hydraulic criteria.
Standard MOUSE notation for significant thresholds in graphical output:

- Indicates the level of the bottom of structure in general (pipe, manhole, detention pond)
- Indicates crest level in general (weir crest, for example)
- Indicates the level of the top of structure
- Indicates the level "zero" (0.00 m)
- Indicates an internal connection in the system

**B4.1490**: Considerable volumes of CSO's are produced in the neighbourhood of this point in this reference situation. Slight surface flooding is simultaneously observed starting at t=25 min.

**B4.1491**: Considerable amounts of CSO's observed at this point between t=25-50 min. The highest peaks of this overflow cannot be given even primary treatment.

**B4.1500**: Surface flooding observed at this point between t=15-70 min.

**B4.1480**: Time series at the cost points in the 'LOOPED' system, corresponding to the application of 'default' control strategies on the system (Q - Regulator 'closed')

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Pipe Model</th>
<th>Datafile</th>
<th>Result File</th>
</tr>
</thead>
<tbody>
<tr>
<td>27-Jan-1994</td>
<td>11:44</td>
<td>DynWave</td>
<td>Example1.SWF</td>
<td>Closed.PRF</td>
</tr>
</tbody>
</table>
B4.1500: Relatively small volumes of CSO's are observed in the neighbourhood of this point between t=30–40 min.
A0.0327: Considerable amounts of CSO's are produced through this point between t=25–80 min, which continue to occur towards the end of the simulated event. Due to the large volume of this overflows the higher peaks cannot even be provided with primary treatment.

Time series at the costpoints in the 'LOOPED' system, corresponding to the application of 'default' control strategies on the system (b. – Regulators 'open').

Sample: EXAMP1.SWF
Result file: NO_CONT.FRF
Pipe model: DYN.WAVE
B4.1490: Relatively small volumes of CSO's are now produced with this 'quasi-optimal' strategy in the neighbourhood of nodes B4.1490 and B4.1491. Due to this fact they can now be given at least primary treatment.

B4.1490: A similar situation to that described above is observed at this point.

B4.1480: The overflows through this point are considerably less if compared to default strategies (those strategies in which the main flow regulators are not operated in time, but rather they are set to intermediate positions or they are opened). The comparison with other default strategies shows in this case similar results to the comparison with the regulators open. Therefore, they are not included.

Time series at the costpoints in the 'LOOPED' system, corresponding to the 'quasi-optimal' vectors after one main updating cycle.
A very similar situation to that described in the time series obtained from the application of the quasi-optimal control strategies is observed here. The differences are produced by 'refinements' or 'fine tuning'. It is observed that the CSO's through A0.0327 are further reduced.

Time series at the costpoints in the 'LOOPED' system, corresponding to the 'quasi-optimal' vectors after two main updating cycles.
7.2.2 Preliminary conclusions. Test Case 1

* The street flooding at the nodes 'B4.1500', 'B4.1501' and 'B4.1491', which was present most of the time during the situation in which the regulators were closed (and which was introduced in order to establish a reference situation with theoretical than rather practical meaning) was completely prevented by the quasi-optimal control strategy found by the optimizer.

* The magnitude of the untreated overflows through 'A0.0327' was considerably reduced by this quasi-optimal strategy if compared to the situation in which no control was effected (regulators completely open). In effect, this figure reaches approximately 3000 m$^3$ of partially treated or untreated overflows through the node A0.0327 in the non-controlled situation and only approximately 300 m$^3$ (about only one tenth of the reference figure) with the strategy obtained after the numerical optimisation process.

* For this simplified prototype of an urban drainage system, four costpoints appear to be enough in order to allow a satisfactory economic representation of the control process. However, more costpoints could also be included in order to provide a more effective cost-representation.

* The three-point cost function with straight segments, seems to be a simple but effective model for the representation of the cost functions in this system. This observation is made on the basis of observing the search patterns of the fully non-linear optimisation methodology in this as well as other trial examples. In effect, the algorithm showed here a sustained orientation (non-hemstitching pattern) towards the feasible constrained optimum from a starting point situated in the neighbourhood of such a singular point. Mathematically speaking, this is a clear indication that the quasi-Newton algorithm (robust in this kind of situations unlike many other search algorithms) is properly extracting the relevant information about the curvature of the search space in this complex region, which is essentially determined by the kind of curve defined in the reward functions (in this case three-point function with straight segments).

* As observed from the results of some of the tests performed, only the first and second cycles of updating of the Hessian matrix (for this implementation of the Broyden-Fletcher-Goldfarb-Shanno algorithm) yield significant improvements from the initial situation. This is particularly true for the first updating cycle (performed with the identity matrix).

* After the first updating cycle to the inverse of the Hessian matrix, the total surrogate cost associated to the solution strategy was only $25.6\%$ (cost reduction $= 74.4\%$) of the total cost of effecting no regulation in the system. After the second main updating cycle, this value was only $22.7\%$ (cost reduction $= 77.3\%$) in relation to the cost of the same reference situation. These figures are of course only valid for this particular and rather special example.

7.3 EXAMPLE 2 - "A SIMPLE SERIAL LAYOUT OF REGULATORS"

As suggested by its title, besides testing the non-linear optimizer with a new, but also simplified setup, the main objective of this exercise was to test the ability of this algorithm to handle serial layouts of regulators. In effect, this sewer system include two underflow gates (at nodes
Applications

'JUNC_01' and 'NODE_03' which due to their location in the system show a serial connection between them.

The combined sewer system for this example has a surface of approximately 3.75 Km² (2.5 by 1.5 Km), with pipes whose diameters vary between 0.75 m and 1.25 m. When aggregated over pipes and manholes, the storage capacity of the system is approximately equal to 4000 m³. The rain is entering the system through four catchments (where the rain sensors are installed) and is supposed to be evenly distributed over the system. These four catchments are connected to the following nodes:

1. 'NODE_01', with a surface of 20.00 ha, no inhabitants, and 60% impervious area.
2. 'NODE_02', with a surface of 9.00 ha, no inhabitants, and 60% impervious area.
3. 'NODE_03', with a surface of 10.00 ha, no inhabitants, and 60% impervious area.
4. 'NODE_04', with a surface of 12.00 ha, no inhabitants, and 60% impervious area. Thus, the total area directly contributing to surface runoff is 30.6 ha. The rain hydrograph employed in this example (once again an extreme rain event) is schematized in Figure 47.

![Rain Hydrograph](image)

Figure 47: Representation of the rain hydrograph used for RTC in example 2

This simple system is composed of seven nodes, one overflow weir (at 'BASIN_1'), six pipes and nine sensors (comprising four rain sensors located at each of the catchments and five level sensors coinciding with the five costpoints).

In addition to the overflow weir, there is a pump installed at 'BASIN_1' with the purpose of pumping water at a constant rate of 0.5 m³/s towards the treatment plant after the water level in the structure has reached a certain 'starting' level. When the magnitude of the inflows arriving to the structure in 'BASIN_1' exceed the capacity of the pump, untreated overflows will be produced.
through the weir. The magnitude of these overflows is related to the water level registered by the level sensor connected to this node.

As mentioned above, five costpoints were defined in order to allow an accurate representation of the economic model to describe the combined sewer system to the mathematical optimizer. These all were associated to level sensors. These sensors were located at the following nodes: 'NODE_01', 'JUNC_01', 'NODE_02', 'NODE_03', 'BASIN_1'.

The order of the costpoints is defined in the '.NET' file which also contains information about sensors and regulators in the system. The input parameters for the numerical optimizer are contained in the file 'exper.inp'. A plan view (schematic representation) of the urban drainage network of this example is shown in figure 48.

![Plan view of the urban drainage network for example 2](image)

Figure 48: Plan view of the urban drainage network for example 2

There are two main flow regulators in this system (underflow gates) situated at 'NODE_03' and 'JUNC_01'. As indicated, they are serially connected. The operational range for the regulators is:

1. Gate at 'JUNC_01' between 10.00 m and 11.25 m.
2. Gate at 'NODE_03' between 12.00 m and 13.00 m.

The reward functions related to each of the costpoints (in the order stated before) are shown in Figure 49. These cost functions were again constructed in order to reward an optimal use of the storage, transport and treatment capacities in the system. An possible further test would be to revert the reward functions assigned to water levels and discharges in order to test the stability of the numerical optimiser.
The initial situation in the system

The default control strategy - for the flow regulators - in the on-line control of urban drainage networks is usually understood as one in which weirs and gates are open and pumps work in an automatic mode with locally specified pumping strategies. A significant volume of untreated combined sewer overflows was observed through the weir in 'BASIN_1' (when this default strategy was applied). The total volume of these overflows was observed to be approximately 4050 m³. No surface flooding was observed at the manholes in the system.

On the other hand, if the regulators were closed, a considerable surface flooding was produced at the following nodes: 'NODE_01', 'NODE_02', 'JUNC_01' and 'NODE_03'. However, in this situation no combined sewer overflows are observed at the weir in 'BASIN_1'.

The reward balance was established in such a way that an optimal solution 'pointed' to intermediate positions or settings for the regulators, because the consequences of both extremes (open or closed) were highly penalized. The 'decision matrix' in this problem was composed of (ten x ten) decision variables, since a five-stage, time-shifted control strategy is to be evaluated for two main flow regulators.

Results

The results were specified to the fully non-linear numerical optimizer after one and two main updating cycles to the hessian matrix, respectively. The event horizon (duration of the
hydrodynamic simulations) was taken as 120 minutes. This is, in this case, enough for a complete representation of the inflow hydrograph produced by the rain event described above, which lasts approximately 60 minutes. A time step equal to ten seconds was used for the simulations. The saving rate was set to twelve in order to obtain a convenient resolution of the results.

Four Fibonacci experiments were specified for the single-line search, which, together with a value of 0.070 for the fractional resolution, allowed a final interval of uncertainty of 0.228 m to be achieved, from the initial unitary interval of uncertainty. A five-point, time-shifted control strategy was used in order to describe the settings for the regulators over the forecasted horizon.

The initial control vectors supplied to the numerical optimizer in this example can be described as one in which the gates were closed, weirs were open and pumps were locally controlled in order to ensure that every displacement of the regulators would cause an effective change in the flow situation during the process of slope evaluation. Therefore, the specified initial strategies were:

For the gate at 'JUNC_01':

<table>
<thead>
<tr>
<th>time (min)</th>
<th>0.00</th>
<th>44.00</th>
<th>60.00</th>
<th>86.00</th>
<th>120.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stt (m)</td>
<td>10.00</td>
<td>10.00</td>
<td>10.00</td>
<td>10.00</td>
<td>10.00</td>
</tr>
</tbody>
</table>

For the gate at 'NODE_03':

<table>
<thead>
<tr>
<th>time (min)</th>
<th>0.00</th>
<th>44.00</th>
<th>60.00</th>
<th>86.00</th>
<th>120.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stt (m)</td>
<td>12.00</td>
<td>12.00</td>
<td>12.00</td>
<td>12.00</td>
<td>12.00</td>
</tr>
</tbody>
</table>

The runoff forecasts were performed by using the detailed runoff model of the MOUSE system (see section 2.8.2). Fully dynamic simulations were performed with the hydrodynamic models of this standard software package. According to the already-defined cost functions, the cost of these default control strategies is:

Cost of the control strategy at CP#1 ('NODE_01'): 1.022E+05
Cost of the control strategy at CP#2 ('JUNC_01'): 7.512E+05
Cost of the control strategy at CP#3 ('NODE_02'): 4.451E+05
Cost of the control strategy at CP#4 ('NODE_03'): 1.170E+06
Cost of the control strategy at CP#5 ('BASIN_1'): 0.919E+02
TOTAL COST OF THE CONTROL STRATEGY: 2.489E+06 C.U.

As an essential reference, we must add that the situation of regulators completely open would have given a total cost of 2,559 x 10^6 C.U., largely due to the cost of untreated combined sewer overflows at 'BASIN_1'.
After 1 main updating cycle to the Hessian matrix (one search for the sliding direction plus one single-line search) the optimisation framework found the vectors illustrated in Figure 50.

![Quasi-Optimal Vectors after one main updating cycle.](image)

*Figure 50: Vectors found by the optimisation system after one main updating cycle*

This control strategy produced a total surrogate cost of $8,443 \times 10^5$ CU. Seventeen function evaluations (each involving one hydrodynamic simulation) were required to achieve this result. In order to reduce the number of function evaluations to fifteen, a four-point control strategy could also have been used.

The time series corresponding to the situations of regulators closed, regulators open (no control) and regulators operated according to the strategy vectors found after one main updating cycle are shown below. Since further updating cycles were observed to provide negligible improvements in this case, they have not been included. The time series have been plotted for a better comprehension of the results.
BASIN_1: Relatively large volumes of CSO's are observed in the neighbourhood of this point as well as around JUNC_01. The upper peaks of these discharges cannot be even primarily treated at the treatment plant.

NODE_01: Slight surface flooding is observed in the neighbourhood of this node.

Time series at the costpoints in the 'SERIAL' system, corresponding to the application of 'default' control strategies on the regulators (a. - regulators 'open')

DATAFILE : EUREKA_2.SWF
RESULT FILE : NO_CONT.PRF
PIPE MODEL DYN.WAVE
NODE_03: Considerable and persisting street flooding is observed in the vicinity of this point. A similar situation is observed in NODE_02, NODE_01, and JUNC_01. No CSO's are observed in the neighbourhood of BASIN_1.

Time series at the costpoints in the 'SERIAL' system, corresponding to the application of 'default' control strategies on the regulators (o.- regulators 'closed').

<table>
<thead>
<tr>
<th>DATAFILE</th>
<th>EUREKA_2.SWF</th>
<th>PIPE MODEL DYN.WAVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT FILE</td>
<td>CLOSED.PRF</td>
<td>CALCUATED: 27-JAN-1994, 12:50</td>
</tr>
</tbody>
</table>
**NODE_03**: The situation in NODE_03 and NODE_02 is relatively normalised when this 'quasi-optimal' strategy is applied.

**BASIN_1**: There is a considerable reduction of the CSO's. They can at least receive primary treatment. The comparison with other time-invariant default strategies (i.e., regulators set in an intermediate position during the whole forecasted horizon), shows similar results to those obtained by comparing with the situation of no control. Therefore they have not been included. The reference situations are included here for they are of interest to the numerical optimiser.

---

**Time series of the costpoints in the 'SERIAL' system, corresponding to the quasi-optimal vectors found after one main updating cycle to the Hessian matrix.**

- **DATAFILE**: EUREKA_2.SWF
- **PIPE MODE**: DYN.WAVE
- **RESULT FILE**: SOL1CYC.PRF
- **CALCULATED**: 27-JAN-1994, 13:01
Preliminary conclusions

* The surface flooding at 'NODE_01', 'NODE_02', 'NODE_03', 'JUNC_01', which was present in the situation in which the regulators were closed was completely prevented by the control strategies found by the optimizer as a solution for this example.

* The total volume of untreated overflows through the weir in 'BASIN_1' was considerably reduced by the quasi-optimal strategy (to approximately 1475 m$^3$) if compared to the situation with no control (in which this figure was approx. 4050 m$^3$). The difference between these two numbers represents - in this simple system - the volume stored inside the system (2575 m$^3$ increase in the storage capacity of the system).

* After the first updating cycle to the inverse of the Hessian matrix, the total cost associated with the solution strategy was only 33.0% (cost reduction 67%) of the cost of no control in the system. In relation to the hypothetical situation of keeping regulators closed during the whole rain event, this figure would be 34.2% (cost reduction 65.8%).

* Again, three-point cost function curves with straight segments were successfully used in order to describe the economic model of the system, although a more effective cost representation could have probably been achieved with exponential functions in this case.

* The fully non-linear optimizer was shown to be capable of controlling systems with a serial layout of regulators.

* An extensive redistribution of mass was achieved by the optimization process in the system, allowing a substantial increase (as indicated above) of its storage capacity.

7.4 EXAMPLE Nr. 3 - "THE URBAN DRAINAGE NETWORK IN GOTHENBURG"

The third example included in this report had necessarily to be a real-life example. The combined sewer system in Gothenburg (Sweden) was selected due to several reasons such as: the presence of a complete and organized set of data, a significant natural potential for optimization, a large-scale system, with complex hydraulic response, relatively large catchments, and other physical and economic features. For this study, a decision was taken in relation to the location of the above mentioned sensors and regulators, in order to proceed to perform the fully non-linear optimization on this system. A first proposal from the team of engineers in charge of the operation of the system was adopted for the sake of consistency.

This scheme (which appears to be far from 'optimal' from the present point of view since it allows an important part of the water in the system to flow without regulation under 'extreme' rain events) consists of four regulators (one underflow gate and three weirs) located north, east, south and at the south-centre of the system. In addition there are pumps located at the entrance of the treatment plant, and these are locally controlled with the default strategy.
The existence of large non-regulated inflows causes, as will be shown shortly, a sub-optimization of the system for this extreme rain event. Still, the results obtained show a substantial increase in the storage capacity of the system, when the solution strategy is applied to the proposed regulators.

The rain event selected to test the optimizer, although very intense and with a large total amount of rainfall, did not belong to the statistical records of rainfall in the Gothenburg area. It was taken from records of rainfall in Denmark (Odense), a point some 300 Km south-west of Gothenburg. According to preliminary calculations been performed, almost 40% of the inflows were beyond regulation in the situation in which the real-time control was performed (in which case the system contained 300,000 m³ of water from an earlier rainfall). This particular rain event had been chosen earlier as a one that provided exceptional characteristics for testing real-time control systems and it was adopted here in order to simplify comparison with earlier simulations of the Gothenburg system.

The main purpose of this exercise was to show the ability of the non-linear optimizer to deal with the operation of a large, complex system, equipped with a relatively high number of different types of regulators, as well as the effectiveness of the economic representation of a system with a very complex hydraulic response to the operation of flow regulators. This situation required a careful selection of the number and location of the costpoints in the system. The specification of the cost functions to provide the so-called rewarding criteria is also a critical issue.

7.4.1 Brief description of the system

The combined sewer system in Gothenburg has a surface of roughly 152 Km² (10.5 by 14.5 Km). This system serves the whole region of Gothenburg in Sweden’s west coast. About 793,000 population equivalents including industry are connected to this system. The treatment plant at ‘RYAPS’ can treat flows up to 518 000 m³/day (6 m³/s) with full secondary treatment.

This system has been built as a tunnel system, with approximately 120 Km of tunnels excavated in bedrock, with cross sectional areas varying from 6 to 18 m² and bottom slopes from 0.07% to 0.1%. Several pumping stations are installed along the system and play an active role in the flow regulation. The storage capacity in Gothenburg has been estimated in approximately 1,000,000 m³. The system is composed of 78 nodes, 72 pipes, 4 pumps and 21 overflow weirs, and in this example was tested with 4 flow regulators located as follows:

A gate in branch 'A3 \( \& \) A1'; a weir in branch 'NORRSTR \( \& \) R5'; a weir in branch 'OSTSTR \( \& \) KO' and a weir in branch 'SYDSTR \( \& \) KNAPEG'. There was also a set of pumps in RYAPS that were locally controlled. Four level sensors and four flow sensors were finally selected to represent the ‘cost sub-network’ as follows:

1. Level sensor at node 'A1'.
2. Level sensor at node 'NAM9'.
3. Level sensor at node 'NAM10N'.
4. Level sensor at node 'NAM11'.
5. Flow sensor in branch 'RYAPS ᶦ RYABF'.
6. Flow sensor in branch 'NAM13BP ᶦ 0'.
7. Flow sensor in branch 'REGLKOD ᶦ 0'.
8. Flow sensor in branch 'NAM16JG ᶦ 0'.

A plan view of Gothenburg urban drainage network is schematised in Figure 51.

![Plan view of the urban drainage network of the city of Gothenburg](image)

*Range of slope = 0.07% ~ 0.1%*  
*Equiv. Diameters = 1.0 ~ 2.5 m*

*Figure 51: Plan view of the urban drainage network of the city of Gothenburg*

The operational ranges (physical boundaries) for the regulators are:

- Gate 'A3 ᶦ A1' : Between -1.37 m and -0.37 m.
- Weir 'NORRSTR ᶦ R5' : Between 4.25 m and 5.0 m.
- Weir 'OSTSTR ᶦ KO' : Between 11.65 m and 15.00 m.
- Weir 'SYDSTR ᶦ KNAPEG' : Between 7.96 m and 10.00 m.

The cost functions associated with each of the *costpoints* (in the order stated above) are shown in Figure 52. In these graphs we have plotted Water levels (WL) or discharges (Q) ᶦ X axis against Cost (CU) ᶦ Y axis.
These cost functions were carefully designed with the objective of rewarding a high storage inside the system. In this case, non-linear functions were provided. These were mostly either exponential, or mixed exponential and linear functions (segmented for convenience). They were generated by using the program CFGEN.

Figure 52: Reward functions associated to the costpoints in the model of Gothenburg

The rain hydrograph used in this example is shown in Figure 53. As explained above this rain was actually registered on the 29/07/1972 in the area of Odense, Denmark. This rain possesses a relatively large total volume mostly concentrated between 200 and 300 min.

The urban drainage network in Gothenburg can 'drain' relatively fast due to its inherent characteristics.
Figure 53: Rain hydrograph of the rain event employed for the model of the UDN of Gothenburg

This rainfall was selected in order to create a scenario in which relatively large volumes of combined overflows were produced in several areas in the system. It is considered important to provide an inflow event which cannot be contained by the system, in order to effectively perform optimisation on the basis of the criteria discussed in section 7.1.

In this last case study the 'decision matrix' assembled was composed of (24 x 24) decision variables since a six-stage, time-shifted control strategy was evaluated for four main flow regulators.

7.4.2 The initial situation in the system

As stated earlier, the hydraulic response of the combined sewer system in Gothenburg to the operation of regulators is rather complex. This is a large system with multiple connections in which multiple overflow weirs have been built along the system. On the other hand, there are several pumping stations which play an active role in the regulation of the system. The pumping strategies applied at 'RYAPS' for example will have considerable influence on the total volume
of combined sewer overflows produced through the overflow weirs in the neighbourhood.

For this example, the rain that occurred on the 29/07/72 in Odense was selected from the ODE100 database. This rain was considered to be uniformly distributed and enter the system through 20 catchments.

As a result, large inflow hydrographs are produced, resulting in surface flooding at certain points located downstream in the system. The situation may then occur, in which persistent surface flooding is produced downstream in the system while spare storage capacity is observed upstream. One of the ways to 'correct' this imbalance, is to construct some extra regulators further upstream in the system. However, this could be expensive.

A situation often observed is that in which the regulators are located downstream in the system, which can be very effectively operated to reduce costs for small or medium scale systems, and to a certain extent even for large systems like this one. If the regulators are located downstream, a larger time span is available for the numerical optimization, since most of the inflows will take more time to reach the downstream points of the sewer network.

According to this analysis, we might speak of a 'reduction' rather than a 'prevention' of undesirable effects in this problem. The same holds for untreated overflows, which occur through many points in different locations in the system. We shall subsequently argue that the situation in which a complete prevention of undesired effects is achieved (regardless the approach employed for real-time control) under extreme storm conditions for complex real world systems like this one, is unrealistic, due to the existence of large volumes of flow which are beyond any effective regulation since these rain storms simply cannot be contained within the system with any affordable level of control.

However, as suggested above, significant improvements can be achieved. This is shown by the results achieved by the non-linear optimizer. The full scale test of the combined logical-numerical optimisation system has been realised through this last example, although clearly further testing will still be required.

7.4.3 Results

The forecasted horizon, (duration of the hydrodynamic simulations) was set in this problem to twelve hours (720 min). This is enough for a complete representation of the average inflow hydrograph caused by this rain event. A time step equal to 60 seconds was used for the HD simulations. The saving rate for the results was set to 30 in order to obtain a convenient resolution for the output of results. A number equal to four Fibonacci experiments was specified for the single-line search, which together with a value of 0.050 for the fractional resolution (parameter of the fibonacci search), allowed a final interval of uncertainty of approximately 0.227 m from the initial unitary interval of uncertainty to be obtained.

A six-point, time-shifted control strategy was used in order to describe the settings for the regulators during the forecasted interval. This time-variable strategy was in this case discretized
every 120 min, approximately (although the discretisation interval can be variable in this framework).

The initial strategy for the flow regulators was obtained from the results of the logical optimisation step. These vectors, which were rapidly obtained (in just a few minutes), provided a good starting point for the fully non-linear numerical optimisation system. The initial vectors allowed a significant enhancement in the system's performance.

As the most important reference for comparison, we may observe that the cost of 'default control strategies' (corresponding to no control of the weirs and gates and local control of the pumps) in this system would be $7.977E+09$ CU according to the economic model specified by the cost functions given earlier.

The vectors corresponding to the solution obtained by the intelligent agent (initial vectors for the non-linear, numerical optimiser) and to the solution obtained after one main updating cycle to the inverse of the Hessian matrix are illustrated in Figure 54.

![Figure 54 (a): Quasi-optimal vectors found by the intelligent agent and the fully non-linear numerical optimiser](image-url)
The total costs associated to these 'quasi-optimal' vectors are:

\[\text{i)}\]
For the strategies found by the intelligent agent:

\[
\begin{align*}
\text{Cost of the control strategy at CP} #1 & : 2.117E+06 \\
\text{Cost of the control strategy at CP} #2 & : 1.954E+08 \\
\text{Cost of the control strategy at CP} #3 & : 2.298E+09 \\
\text{Cost of the control strategy at CP} #4 & : 4.516E+06 \\
\text{Cost of the control strategy at CP} #5 & : 5.48E+08 \\
\text{Cost of the control strategy at CP} #6 & : 7.501E+08 \\
\text{Cost of the control strategy at CP} #7 & : 3.912E+06 \\
\text{Cost of the control strategy at CP} #8 & : 3.911E+06 \\
\text{TOTAL COST OF THE CONTROL STRATEGY} & : 4.213E+09
\end{align*}
\]

\[\text{ii)}\]
For the strategies found by the fully non-linear, numerical optimiser:

\[
\begin{align*}
\text{Cost of the control strategy at CP} #1 & : 4.082E+06 \\
\text{Cost of the control strategy at CP} #2 & : 2.737E+08 \\
\text{Cost of the control strategy at CP} #3 & : 1.707E+09 \\
\text{Cost of the control strategy at CP} #4 & : 4.515E+06 \\
\text{Cost of the control strategy at CP} #5 & : 5.932E+08 \\
\text{Cost of the control strategy at CP} #6 & : 4.344E+06 \\
\text{Cost of the control strategy at CP} #7 & : 7.220E+08 \\
\text{Cost of the control strategy at CP} #8 & : 3.911E+06 \\
\text{TOTAL COST OF THE CONTROL STRATEGY} & : 3.744E+09
\end{align*}
\]
The numerical solution vectors were obtained after 23 function evaluations, with a final interval of uncertainty in the results of approximately 0.220 m. Approximately 45 minutes were required to achieve the solution with one main updating cycle (with two cycles this figure is approximately 1 hour 20 minutes on a HP9000 UNIX workstation). The time series of water levels or discharges at the costpoints corresponding to the situations of regulators closed, regulators open, and regulators operated according to the 'quasi-optimal' vectors have been plotted for a better illustration of the results, as shown below.
Combined Logical Numerical Enhancement of Real-Time Control

This and the following cases are described in the preliminary conclusions (corresponding to section 6.3.3)

**Meter Water Level Nodes**

- **A1**
- **NAM9**
- **NAM10N**
- **NAM11**

**Discharge**

- **RYAP** → **Ryabf**
- **NAM13BP** → **0**
- **NAM16JG** → **0**
- **Reklod** → **0**

Data File: GOTESORG.SWF  Pipe Model: DYN.WAVE
Result File: CLOSED.PRF  Calculated: 19-Apr-1995, 09:54
Combined Logical Numerical Enhancement of Real-Time Control

METER WATER LEVEL NODES — A1

METER WATER LEVEL NODES — NAM9

METER WATER LEVEL NODES — NAM10N

METER WATER LEVEL NODES — NAM11

M3/s

DISCHARGE... RYAPS -> RYABF

M3/s

DISCHARGE... NAM13BP -> 0

M3/s

DISCHARGE... NAM16JG -> 0

M3/s

DISCHARGE... REGUKOD -> 0

DATAFILE : GOTEBOG.SWF
RESULT FILE : FIRST.PRF
PIPE MODEL DYN WAVE
CALCULATED : 19-APR-1995, 10:02
Comparison between TS corresponding to default strategies and the vectors derived by the intelligent agent.

HECTOR (RHD)

DATAFILE : GOTEBOUG.SWF
RESULT FILE : FIRST.PRF
PIPED MODEL : DYN.WAVE
CALCULATED : 19-APR-1995, 10:02

MOUSE
Comparison between TS corresponding to default strategies and the quasi-optimal vectors (after 1 main cycle)

HECTOR (RHD)

DATAFILE : GOTEBOGG.SWF
RESULT FILE : PRFICYCPRF
PIPE MODEL DYN.WAVE
CALCULATED : 10-MAY-1995, 09:25
Preliminary conclusions

* The surface flooding at nodes 'A3', 'NAM10N' and 'NAM9' was significantly reduced by the strategy vectors derived by the combined logical-numerical framework after one main updating cycle. The situation employed as a reference for this analysis is that of 'regulators closed'.

* The total volume of untreated overflows, observed to be critical in the neighbourhood of 'RYAPS' during the non-controlled situation, was also significantly reduced by this quasi-optimal strategy found by the combined logical-numerical framework. This fact, which is shown in the time series, is also mirrored in the results of the process of cost evaluation.

* Observation of the operation of the regulators performed by the fully non-linear optimizer indicates that the gate at 'A3 p A1', the pumps at 'RYAPS' and the gate at 'NORRSTR p R5' act as the main flow regulators in the system. In other words, according to this configuration, these regulators have a major influence on the flow situation at the costpoints defined for this problem.

* The influence of the other regulators does seem to be considerably less. The solution strategy shows that the remaining weirs were lifted slightly during the moment when most of the inflows arrived to the downstream part of the system (about 300 minutes after the beginning of the rain) in order to prevent excessive untreated overflows in this area.

* As mentioned earlier, the pumping strategy applied at 'RYAPS' have a considerable influence in the magnitude of the untreated overflows in this region. The default control strategy for these pumps is to control them locally - according to strategies specified beforehand - by using a standard control technique.

* An interesting test would be to increase the resolution of the control strategy around this time (300 min approximately) and repeat the optimization under the same conditions. A more effective control is expected since the strategy resolution must be enhanced when the inflows are varying more rapidly. However, this would increase the number of points to be analyzed in the strategy for each regulator (the size of the matrices assembled by the optimizer is equal to the product of the number of regulators and the number of points in the strategy vectors), and therefore produce an increase in the time required to complete the optimization process.

* After the logical optimisation cycle the total cost associated to the solution strategy was only 52.81% (cost reduction 47.19%) in relation to the cost of 'default' control in the system. This initial vector was then supplied to the fully non-linear numerical optimiser which found (after one main cycle of updating of the Hessian matrix) a quasi-optimal vector. The cost associated to this 'quasi-optimal' vector was only 46.9% (cost reduction 53.1%) in relation to the cost of 'default' control in the system. In absolute numbers, the results of the optimization show that according to this solution it is possible to spare approximately 4.233 x 10^6 Cost Units (CU).

This is considered a conservative estimate since only seven costpoints were considered to provide a good enough economic representation of a large-scale system like this one.
* The solution presented here is only one among several possible solutions in order to improve the utilization of the storage capacity of the system. The result achieved will depend on many factors, such as the configuration of the cost sub-network, the number and location of costpoints, the discretisation system used to describe the control strategies in the regulators, and the construction of the cost functions.

* A more effective redistribution of mass inside the system would probably be achieved by shifting the location of some regulators towards points further upstream in the system. This would require, of course, of a careful analysis, due to the system's complexity and also due to the fact that the situation might be created in which the numerical strategy would not be found in time for a given regulator located too far upstream in the system since the main part of the inflows could reach the regulator before the forecast could be made up. However, it will always be possible to obtain the first approximate quasi-optimal control strategy after the logical optimisation step.

* The reduction in the global integrated cost according to well defined flow criteria, depends upon the volume of non-regulated flows in the network, for a given flow event. In this case, due to the initial volume of water in the system (approximately 300,000 m³) and due to the magnitude of the rain event selected, the volume of non-regulated flows was approximately 40%, which represents a considerable percentage of the total flow. If this volume had been smaller, the cost reduction achieved* would have been expected to be bigger. This is indicated by the results achieved in the first two examples.

* The results provided by the intelligent agent were obtained very rapidly. This step takes typically just a few minutes. These vectors were then supplied to the fully non-linear numerical optimisation framework, which required approximately 45 minutes (with only one main updating cycle) in order to find the quasi-optimal vectors.

* The differences between benefits associated to 'default' and 'optimal' control strategies are directly established in relation to well defined flow criteria (flowrates, volumes of overflows, magnitude of the surface flooding when present) rather than in relation to globally integrated surrogate 'costs'. However, it is observed that as pointed out in paragraph 7 of this section the results of the reward criteria integration process mirrors the accomplishment of favourable flow conditions in the terms discussed in section 1.3 and 1.4. By 'default strategies' we understand those strategies applied on the flow regulators of the network (gates, weirs, valves) in which these regulators are either not operated (constant settings) in time or (in the case of pumps) operated according to locally defined control strategies.

* Provided a good economic model is available for the process; see section 2.8.
Chapter 8  Conclusions

8.1  Introduction

This research was carried out with the purpose of assessing the feasibility of applying a combination of potentially applicable techniques for the real-time control of urban drainage networks. Having investigated the capabilities of the prototype system on several urban drainage networks, it is now time to recapitulate contents with the purpose of assessing the potential of the present hybrid optimisation framework for the solution of this complex problem.

An evaluation of the performance of any real-time control system, necessitates an insight into the influences of the following dynamic processes:

- The system's input.
- The system's response to this input.
- The output to the environment.
- The response of the environment to this output.

• The system's input

The loading of an urban drainage system can be defined as a vector which is variable in time and space. This vector is also stochastic. Due to its stochastic characteristic, an optimal operation of the system can only be attained if the control strategies are derived in real-time on the basis of the actual (ongoing) situation, because every loading pattern is different and cannot be 'extrapolated'. In order to react properly to this dynamic loading, a flexible operation of the flow regulators is required. We have argued that an effective response to this input can be achieved by properly combining on-line measurements and forecasts in order to avoid partial decisions, that is, decisions which represent the best choice at the present stage of the event, but which may not be suitable for the future stages.

• The system's response to the input loading

The response of real world systems to dynamic loading is in general, complex. As explained in section 7.3, a situation commonly observed in urban drainage networks, is that in which persistent surface flooding and sewer overflows occur in downstream areas of the network while spare storage capacity is available upstream. Even an ideal (homogeneous) loading would lead to an uneven use of the system’s capacity.

The objective of the operation of the flow regulators is to perform a mass redistribution in the system which allow an optimal utilisation of the storage capacity of the network.
The other component of the system's response to precipitation loading concerns the water quality parameters. Unfortunately, due to the number and complexity of the processes involved it is still difficult to predict accurately the distribution of pollutants in the flow.

- The output to the environment and the response of the environment to this output

The effects of combined sewer overflows on the receiving waters are very variable in space and time. In general, they are strongly related to the characteristics of the receiving natural recipient. For example, overflows to big streams with strong circulation are in principle preferred to overflows to small stagnant water bodies.

A quantitative evaluation of the effects of combined sewer overflows on receiving waters seems to be both difficult and costly. In effect, high-rate processes such as fish kills by toxic waste or chemical oxidation-reduction processes require frequent measurements on a local scale. On the other hand, moderate-rate processes such as BOD-oxygen depletion require measurements over wider areas and time intervals.

From the standpoint of elaborating a viable operational strategy for the system, one of the main goals can be defined as that of 'smoothing' the inflow hydrograph so as to deliver the water at a rate which can be assimilated by the treatment plant. The objective of such a strategy is to prevent or at least minimise untreated or partially treated combined-sewer overflows. This is achieved by performing a maximal utilisation of the storage capacity of the network.

The fact that the available capacities are not homogeneously distributed over the system contributes with additional complexity to this problem. The above mentioned problems are the reasons why a dynamic operation of the system is required.

The methods which utilize knowledge - of any kind - for the system's operation are only approximate, although quick. Within the field of numerical optimisation the linear programming methods have proven to be much too restrictive to represent this problem successfully. On the other hand, the most accurate control methods are those of non-linear and dynamic programming, but they may be computationally too demanding in real-time.

The present combined logical-numerical optimisation framework makes use of several techniques in order to reduce the response time needed to effect the optimisation process and at the same time enhance the effectiveness of the control. As described in chapters four and five, these techniques involve the use of knowledge about the system which is used in order to effect a restriction of the search space and also to steer the numerical search process.

It has been argued that - from the hydroinformatic point of view - the use of a hybrid approach represents one of the most effective ways of approaching the problem of real-time control of urban drainage networks.

The reasons for this, from this hydroinformatic point of view, is that in this way we employ two very different kinds of sign vehicles, each of which has its own very specific advantages within equally specific but separated fields of application.
On the other hand we are using numbers as indicative signs, with all their algorithmic, and specially recursive advantages, while on the other side we are using logical statements as expressive signs, with their quite other, but equally specific advantages. The power of combining signs vehicles in this way (for which a paradigm in Semiotics would be the combination of light and sound in cinema and television productions) is what is really being exploited in this approach.

Within the present context, the problem can then be solved in a more efficient way, by approaching its solution from different directions, in order to construct a quasi-optimal strategy in a sequence of stages of increasing accuracy (which has proven to be a quite flexible scheme for real-time control). In this way it is possible to ensure a broader coverage of real-life situations as the spectra of sign vehicles and corresponding methods applied in the solution is also that much wider.

In other words, the characteristics of the problem to be solved (a not very well-defined structure, with strong non-linearities and stochastic input vectors which vary in both time and space) already suggest the application of a hybrid control tool (in the widest sense of the word).

8.2 Some directions for further development

This section strives to analyze the potential directions of further development towards the complete implementation of a general and effective real-time control system for urban drainage networks.

The main elements for such an analysis have been a constant theme through this work. As is usually the case in the process of scientific and technological research, every concluded study represents a further step towards the achievement of a final goal.

As a result, there are always aspects which can be further developed. This is mainly possible (in the present case) due to the rapid development observed in those areas which constitute the foundations for this problem (see Chapter 4).

The ideas which shall be discussed were born during the process of developing of the present prototype on the basis of the study of the latest works in these foundational subjects. These ideas are also based on the observation of those factors which have a significant influence on the performance of the present combined logical-numerical optimisation framework.

This analysis corresponds to the state-of-the-art in the real-time control of urban drainage networks in 1995. As explained in section 1.4, the conceptual basis for approaching this complex problem, have been - mostly - provided by hydroinformatics, the discipline of integration of techniques dealing with the information flows carried by the waters of the Earth: the very arteries and veins of the Biosphere.
The main directions in which further research work could be carried out - in a rather straightforward way - in order to enhance the effectiveness of the present combined logical-numerical framework are:

i) Automatic generation of cost functions

ii) Use of the present combined logical-numerical optimisation framework for training a neural network, an evolutionary algorithm (or similar technique) in such a way that this becomes capable of producing an almost instantaneous response when applied to large scale systems with a relatively large number of flow regulators.

A brief description about each of these suggestions follows:

i) Automatic generation of cost functions

As explained in sections 2.9 and 3.1, the results of the fully non-linear numerical optimisation process can only be as good as the economic model is a good representation (accurately captures the economic consequences) of the complex processes which occur in the real-time control of urban drainage systems during storm events. This economic model is mapped on the basis of the cost functions which in this case can be defined as non-linear functions of practically any required graphical form.

This aspect, which provides one of the most important sources of generality for this kind of non-linear optimisation systems, is also one of the reasons why these cost functions must be carefully defined according to the criteria discussed in sections 2.9 and 5.6.2.

The automatic generation of appropriate cost functions can be viewed as a 'residual optimisation sub-problem' in itself. In this problem, the objectives can be expressed in terms of determining the set of analytical parameters of the 'curve' which better represent (or fits) the objectives imposed by the system's analyst at each of the costpoints.

Apparently this residual optimisation sub-problem is less complicated than the problem of optimisation of the real-time control of an urban drainage system if only because of the relatively well-defined structure of the knowledge domain in the economic analysis and the absence of a stochastic component in the input vectors.

Moreover, this problem can be solved without introducing the real-time constraint, since the cost functions have to be specified beforehand in terms of the objectives to be achieved by the operation of the system.

It is believed that this process can well be carried out automatically by means of an off-line calibration process of the kind described in section 6.1. As an optimisation problem, this problem can be solved by any of the available optimisation techniques.

However, it is important to observe that in this problem the 'rewarding criteria' cannot be accurately defined on the basis of quantitative criteria but rather on the basis of an analysis of 'trends' of the system's 'behaviour' within known economic models. In other words, the
system needs to 'know', a set of output vectors in order to derive an optimal economic model for the process.

It should also be observed that such a economic model should be defined as an independent function of the stochastic component (in the interference vectors) in the domain of urban drainage networks.

Recapitulating, in this problem the system’s input can be regarded as a set of vectors containing 'trial cost functions' (or rather its set of analytical parameters as supplied by the program CFGEN, for example) while the system’s output could be represented as a set of vectors containing parameters describing the effectiveness of the overall optimisation process (such as, for example the cost reduction achieved) under known scenarios. Furthermore, such a system could used initial 'suggestions' used to distinguish 'good' from 'bad' scenarios in order to speed-up the search.

A 'black-box' model could be employed - for example - in order to carry out this task. It is believed that an intelligent agent could also be used in order to tackle this problem in an efficient and fast way.

\[ ii) \quad \text{Use of the present combined logical-numerical optimisation framework for training a neural network, evolutionary algorithm (or similar technique) to produce an almost instantaneous response when applied to large scale systems with a relatively large number of flow regulators.} \]

An urban drainage network can be efficiently operated in real-time if the data about the ongoing process is combined with forecasts about the system state and is used in order to optimally operate the flow regulators during the actual process.

The approach employed by the present combined optimisation framework utilizes an on-line, model-based control technique in which the system state is represented on the basis of the actual (ongoing) situation as indicated by on-line measurements and forecasts. Therefore the control decision is taken on the basis of the knowledge and information which is obtained from the two sources.

This 'scheme' allows a general representation of the process since the ongoing situation is modelled 'on-line' - as it occurs - by combining the two available sources of information as described. It is effective since the control decision is taken not only on the basis of the present situation but also on the expected one which means that the control strategy is not only the best for the present stage of the event but also for the future stages. The pipeflow simulations are performed by numerically solving all terms in the Saint-Venant equations (fully dynamic pipeflow simulation, see section 2.8.3).

However, with the present state-of-the-art in hardware and associated system-software platforms available for real-time control, this general solution obtained, as it employs a fully non-linear numerical optimisation methodology, could still be computationally demanding* (due to the time employed for the fully dynamic pipeflow simulations) in cases of large
networks (with several hundreds or even thousands of nodes) with a relatively large number of active flow regulators which are to be controlled over a long time span (event horizon).

In these cases an approximate but very fast solution could be obtained by using a trained Neural Network (ANN) or similar sub-symbolic technique, which has learned over a previous training set - in an off-line mode, comprising several events of similar characteristics to the expected situation - how to control the network under a sufficiently wide range of extreme storm conditions.

The training set for the sub-symbolic code can then be derived with the help of the combined logical-numerical framework working in a high-accuracy mode (several optimisation cycles) without the real-time constraint.

A considerable research effort was being conducted along these lines both at IHE and DHI, parallel to this work. Reference is made to the publications by Hall and Minns (1993), Minns and Hall (1995), Khondker (1995), Wilson (1995), and Babovic (1995). The use of a sub-symbolic (evolutionary algorithms) technique with application to hydrodynamic model calibration has been reported by Wu (1994) and Babovic and Wu (1995).

8.3 Final remarks

The problem of the optimisation of the real-time control of hydrodynamic networks can be characterised as dynamic, continuous, highly non-linear and noisy due to stochastic input vectors. The use of model-based control techniques offers a unique possibility of dealing with this problem in a satisfactory way.

The aim when applying real-time control on an urban drainage system is the derivation of optimal control strategies for the active flow controllers in the network. These control strategies describe time series of setpoints for the flow regulators which are obtained in order to prevent undesirable effects from occurring, or at least to minimize their effects during the on-line operation of the system. Alongside the reduction of combined overflows, side benefits, such as reduction of surface flooding, lower energy costs, flow equalisation, in-sewer sediment control, supervision and better understanding of the system's operation can also be achieved.

*In spite of the several features incorporated inside and outside the non-linear numerical methodology, which are aimed at real-time implementation. When we speak of computationally demanding we refer to the full optimisation cycle which includes the numerical solution. It is believed that it will always be possible to obtain (in real-time) the solution from the intelligent agent regardless the size and complexity of the network. As explained the logical optimisation step takes usually a few minutes.*
The term 'intelligent control' has become extensively used in today's engineering practice. Intelligent control involves both intelligence and control theory. It must be based upon a serious attempt to understand and replicate what we really mean by intelligence: the generalized, flexible and adaptive capability that we observe in the human brain. Furthermore, it should be firmly rooted in control theory. In relation to our field of interest, our designs must often be intuitive in their early stages, but, once these designs are specified, we should at least do our best to understand them and to evaluate them in terms of the deepest possible mathematical theory.

Intelligent control embraces classical control theory, neural networks, fuzzy logic, Artificial Intelligence (AI) and a variety of search techniques (including for example, genetic algorithms) some of which have been reviewed in section 2.5. We consider useful to quote the words of Werbos (1992) in order to define intelligent control:

"Intelligent control is the use of general-purpose control systems, which learn over time how to optimise in complex, noisy, non-linear environments whose dynamics must ultimately be learned in real-time. This kind of control cannot be achieved by simple, incremental improvements over existing approaches ..."

Hydroinformatics (Abbott, 1991) and its fifth-generation modelling concept in particular, offer a unique possibility to approach this kind of problems from both the conceptual and practical point of view. In order to solve this problem, use must be made of knowledge about the domain based on both general and site-specific properties. While the general knowledge can be 'encoded' if certain well-established knowledge-engineering techniques are used, the site-specific knowledge must be acquired through modelling. There are several techniques which can be utilised in order to solve this complex problem. The most promising today seem to be:

i) Numerical optimisation (especially those of dynamic and discrete-continuous non-linear programming)
ii) Heuristic rules
iii) Diagnosis
iv) Fuzzy logic
v) Artificial Neural Networks
vi) Learning control systems
vii) Adaptive critic methods
viii) Approximate dynamic programming

Often, the complexity of this problem makes it necessary to employ more than one of the above mentioned techniques in order to obtain a solution which satisfies all the constraints.

Intelligent Agents (IA) encoding domain knowledge of different levels seem to offer an effective tool of providing a first - approximate - solution to the problem of real-time control of urban drainage networks. These intelligent agents must make use of fuzzy heuristic rules
in order to represent a world whose structure is often not well-defined. Also the above mentioned Artificial Neural Networks (ANN) provide a useful tool in order to perform the so-called 'first approximation' to the solution.

This first approximation is important because it provides the information required by the numerical optimisation systems (specially those making use of non-linear search techniques) in order to restrict the search space and orientate the search vectors towards the 'feasible-constrained global optimum' discussed in Chapter 5. On the other hand, a numerical optimisation step is necessary because the solutions derived by making use of any kind of knowledge about the domain are usually quite far from the feasible constrained global optimum.

Another possibility aimed at increasing the range of applicability of the fully non-linear numerical methodology that is included in this hybrid optimisation framework - under study at the present time - is that of extending the capabilities of the pipeflow model in order to reduce the time necessary for a single pipeflow simulation. This would be achieved by reducing the number of explicit elements in the resulting pipe matrix. According to a preliminary analysis this would allow a reduction of at least 33% in the CPU time required for a single - fully dynamic - pipeflow simulation.

Model-based control offers a unique possibility for real-time control of urban drainage systems because fully dynamic numerical simulations provide an important source of knowledge in the form of information about a problem whose structure cannot be precisely defined beforehand. The principal task of the intelligent agents is that of reasoning on the basis of this 'information' so to derive the necessary modifications of the so-called decision variables.

This task can be fairly performed with the help of fuzzy logic because this technique was intended to deal with problems which do not have the sharp, well-defined boundaries often associated to mathematical descriptions.

These intelligent agents offer a unique possibility of preventing those 'combinatorial explosions' of possibilities which otherwise arise when performing an exhaustive search on the array of decision variables.

It is the author’s belief, that there is a significant potential in the present combined logical-numerical framework to deal with real-time control. Like any other work, it can and should be further improved. The initial objectives will only be completely realised as this research continues towards practical implementation. However, if, as expected, this work is properly employed in the future within the framework provided by hydroinformatics, it should provide the basis for a modification of our view of how to approach the problem of real-time control of hydrodynamic networks. It must be made explicit again that this has only been possible after '... the Copernican revolution produced by hydroinformatics in hydraulics ..' (Abbott, 1994).
Further developments of the computer hardware leading to significant increases in the computational capabilities in the available software platforms (which is already being realised) will only contribute to expand the scope of applicability of accurate numerical optimisation techniques, which is still rather restricted nowadays. But even if the above mentioned facts are put aside, the final goal of intelligent and effective control of hydrodynamic networks seems to be more within reach now than ever before.
References


About the author

Hector Martín García was born in Havana, Cuba, on August 15, 1962. He studied at the Technical University of Havana, Faculty of Civil Engineering from 1979 to 1985 where he graduated with distinction as a Civil Engineer. After participating in several post-graduate study programs he worked from 1985 to 1991 as an assistant professor and researcher at the Technical University of Havana. There he taught several subjects such as Fluid Mechanics, Physical and Mathematical Modelling, and Applied Hydraulics, among others. During this period he took part in several ongoing hydraulic projects in Cuba, dealing with dams for flood protection and water supply, irrigation systems, pump stations, drainage systems and coastal protection works. He came to Delft in October 1991 in order to attend the first international post-graduate course on Hydroinformatics at the International Institute for Hydraulic, Infrastructural and Environmental Engineering (IHE), in Delft, The Netherlands. After finishing his studies in Delft, he was selected to carry out an MSc study at the Danish Hydraulic Institute in Denmark in a join project with IHE. He received his MSc Degree from IHE for the thesis titled "Optimal Real-Time Control of Unsteady Combined Sewer Flow". On the basis of the results achieved since 1991, the author carried out a PhD study in a joint project of the International Institute for Hydraulic, Infrastructural and Environmental Engineering (IHE), the Technical University of Delft (TUD) and the Danish Hydraulic Institute (DHI). This PhD study, titled "Combined Logical-Numerical Enhancement of Real-Time Control of Urban Drainage Systems" has been supervised by Prof. Dr. Michael B. Abbott. It describes the design and implementation of a hybrid (logical-numerical) framework for the real-time control of urban drainage systems, which makes use of several advantageous features of both logical and numerical techniques in order to approach the optimal control. This approach has been performed for the first time, under the new perspective offered by Hydroinformatics. Several of the author's papers describing different aspects of the research have been included in the proceedings of the XXVIth IAHR Congress "HYDRA 2000" in London, UK, and the Symposium on Intelligent Data Analysis and Systems Research in Baden-Baden, Germany, respectively. An extended paper describing the main aspects of his work on real-time control has also been submitted and accepted for publication in the Journal of Advances in Water Resources.
The aim of the International Institute for Infrastructure, Hydraulic and Environmental Engineering, IHE, is to transfer scientific knowledge and technological know-how related to transport, water and the environment to professionals, especially from developing countries.

IHE organizes regular one-year postgraduate courses which lead to either an MSc degree or an IHE diploma. IHE also has a PhD-programme based on a research, that can be executed partly in the home country. Moreover IHE organizes short tailor-made and regular non-degree courses in The Netherlands as well as abroad and takes part in projects in various countries to develop local training and research facilities.
The pollution of natural recipients with emissions from combined urban-sewer systems is a very dangerous and somehow unresolved kind of negative environmental impact on the water bodies of our planet. Among the several approaches to this problem, the real-time control of those systems offers the most rational and economic solution. The application of real-time control aims at an optimal utilisation of the existing storage capacity and thereby a considerable reduction of undesirable effects by employing regulators for flow control.

This work addresses fundamental problems in this area and offers a combined logical-numerical methodology which computes quasi-optimal control strategies in real-time. The resulting control system is general and has been applied to optimisation of critical flow events in the combined urban drainage system of the city of Gothenburg in Sweden. The present study shows in a practical way the potential of combining different sign vehicles in order to approach the solution of a complex problem. This is an essential postulate of Hydroinformatics.