The COST *Simulation Benchmark*: Description and Simulator Manual

(a product of COST Action 624 & COST Action 682)

Edited by John B. Copp

Preface

This publication focuses on the COST *'simulation benchmark'*, which has been produced as a direct result of co-operation facilitated by two COST Actions. COST Action 682 'Integrated Wastewater Management' (1992-1998) focused on biological wastewater treatment processes and the optimisation of design and operation based of dynamic process models. The current COST Action, 624, is dedicated to the optimisation of performance and cost-effectiveness of wastewater management systems. To accomplish this goal, the Action is focussing on increasing the knowledge of microbial systems and by implementation of integrated plant-wide control based on a description of the entire wastewater system, thereby providing new concepts for dealing with wastewater in a future sustainable society.

COST Mission:

Founded in 1971, COST is an intergovernmental framework for European Co-operation in the field of Scientific and Technical Research, allowing the co-ordination of *nationally funded* research on a European level. COST Actions cover basic and pre-competitive research as well as activities of public utility.

The goal of COST is to ensure that Europe holds a strong position in the field of scientific and technical research for peaceful purposes, by increasing European co-operation and interaction in this field. COST has developed into one of the largest frameworks for research co-operation in Europe and is a valuable mechanism for co-ordinating national research activities in Europe. Today it has almost 200 Actions and involves nearly 30,000 scientists from 32 European member countries and more than 50 participating institutions from 14 different countries.

The *'simulation benchmark'* described in this publication is a fully defined simulation protocol and was developed as a tool for evaluating activated sludge wastewater treatment control strategies. This comprehensive tool has taken several years to develop and is truly the result of a group effort. Over the years, many people have contributed to the benchmark's development, but unfortunately not all of those people have been able to contribute to the production of this publication. Nevertheless, those individuals should be fully acknowledged for their work and input. The following is a list of all the people who have worked on the benchmark project.

Jens Alex	*Chris Hellinga*	*Gilles Patry*
Jean-François Béteau	*Nadja Hvala*	*Marie-Noëlle Pons*
Peppe Bortone	*Matty Janssen*	*Antonio Salterain*
Bengt Carlsson	*Ulf Jeppsson*	*Henri Spanjers*
John B. Copp	*Karel Keesman*	*Imre Takács*
Denis Dochain	*Stefano Marselli-Libelli*	*Henk Vanhooren*
Jeremy Dudley	*Khanh Nguyen*	*Peter Vanrolleghem*
Sylvie Gillot	*Gustaf Olsson*	*Mario Zeč*

In addition to those mentioned above special recognition should go to Dr. Ulf Jeppsson whose 'quality control' and troubleshooting efforts were instrumental in identifying and solving many of the problems presented in the simulation chapters of this book. Further, Dr. Jeppsson should also be recognised for verifying (in some cases) and compiling many of the results that appear in this publication. Dr. Marie-Noëlle Pons also should be recognised for her ongoing efforts with respect to the '*simulation benchmark*' and the maintenance of the COST web page, which has been the forum for disseminating benchmark information to this point. I would also like to thank Dr. Henri Spanjers for his comprehensive review of the manuscript.

The first sections of this book provide an introduction to the '*simulation benchmark*' including the rationale behind its development and a complete description of the benchmark as it is currently defined. The later sections deal with use of the benchmark, and specifically with the implementation of the benchmark into a number of simulation platforms. Experience tells us that commercially available simulation software packages have specific features that can impact on the benchmark implementation. This publication is intended as a means to disseminate the lessons we have learned about specific platforms and make the '*simulation benchmark*' implementation easier for new users. A substantial effort has gone into verifying the steady state and dynamic output data included in the '*simulation benchmark*' description. So far, these results have been verified using BioWin™, EFOR™, GPS-X™, Matlab/Simulink™, Simba®, STOAT™, WEST® and a user defined FORTRAN code (Alex et al., 1999; Pons et al., 1999).

Although this process of cross-platform testing has been a time consuming exercise, it has provided a means to develop a significant insight into the simulators and the simulation process. Further, as each of the simulators requires a different method of implementation, each of the simulators has required simulator-specific alterations and fine-tuning to get agreement in the output data. Knowledge of these simulator-specific alterations is crucial for benchmark use. So, to facilitate the transfer of that knowledge to interested parties, this publication was devised. This publication outlines issues and procedures that are specific for the use of various simulators with the '*simulation benchmark*', but it should be made clear that this publication is not meant to be used or interpreted as a comparative study of these different simulators.

Editor
John B. Copp

Contributing Authors

Jens Alex, Ph.D.
ifak - Institut fuer Automation und Kommunikation e.V. Magdeburg
Steinfeldstr. (IGZ),
D-39179 Barleben, Germany
e-mail: ali@ifak.fhg.de

Jean-François Béteau
Laboratoire d'Automatique de Grenoble,
UMR 5528, ENSIEG-INPG
BP 46
F-38402 St Martin d'Hères cedex, France
e-mail: Jean-Francois.Beteau@inpg.fr

John B.Copp, Ph.D.
Postdoctoral Fellow
AEST, Environmental Technology
Wageningen University & Research Centre
P.O. Box 8129
NL-6700 EV Wageningen, The Netherlands
e-mail: John.Copp@algemeen.mt.wau.nl

Jeremy Dudley, Ph.D.
WRc
Franklin Road, Blagrove
Swindon, WILTS SN5 8YF
United Kingdom
e-mail: dudley@wrcplc.co.uk

René Dupont
EFOR ApS, c/o Krüger A/S,
Gladsaxevej 363,
DK-2860 Søborg, Denmark
e-mail: RED@kruger.dk

Sylvie Gillot, Ph.D.
Postdoctoral Fellow
BIOMATH
Department Applied Mathematics, Biometrics and Process Control
Ghent University,
Coupure Links 653,
B-9000 GENT, Belgium
e-mail: Sylvie.gillot@hobbes.rug.ac.be

Ulf Jeppsson, Ph.D.
Postdoctoral Fellow
Department of Industrial Electrical Engineering and Automation (IEA)
Lund University
P.O. Box 118
SE-221 00 Lund, Sweden
e-mail: Ulf.Jeppsson@iea.lth.se

Jean-Marc LeLann
Laboratoire de Génie Chimique,
UMR-CNRS 5503
ENSIGCT-INPT
18 chemin de la Loge
F-31078 Toulouse cedex, France
e-mail: JeanMarc.LeLann@ensigct.fr

Marie-Noëlle Pons, Ph.D.
Laboratoire des Sciences du Génie Chimique
CNRS-ENSIC-INPL
1, rue Grandville BP 451
F-54001 Nancy cedex, France
e-mail: Marie-Noelle.Pons@ensic.inpl-nancy.fr

Peter A. Vanrolleghem, Ph.D.
BIOMATH
Department Applied Mathematics, Biometrics and Process Control
Ghent University,
Coupure Links 653,
B-9000 GENT, Belgium
e-mail: Peter.Vanrolleghem@rug.ac.be

Table of Contents

1

Benchmark Rationale

The activated sludge process aims to achieve, at minimum costs, a sufficiently low concentration of biodegradable matter in the effluent together with minimal sludge production. To do this, the process has to be controlled. Many control strategies have been proposed in the literature; however, the literature does not provide a clear basis for comparison of these strategies because of the many confounding influences that have an impact on the system. Many of these influences are easily recognised. For instance, physical characteristics of the process can have an impact on process performance, which makes the comparison of strategies applied to different reactor layouts difficult. As well, the influence of a control strategy on process performance is expected to vary with different disturbances, thus the disturbances used to test the control strategy become important. Also complicating the evaluation is the lack of standard evaluation criteria. That is, effluent requirements and treatment costs (i.e. labour costs) are often location specific. This makes it difficult to judge the particular influence of an applied control strategy from a reported performance increase. Furthermore, the objective of reported strategies is not always consistent which may result in the omission of data necessary to make fair and unbiased comparisons.

There are several common control strategies including the maintenance of biomass levels and/or dissolved oxygen concentrations in the aeration tanks by manipulating waste sludge flow, return sludge flow or aeration capacity. Such strategies are based on measurements of mixed liquor suspended solids and/or dissolved oxygen. Other control strategies make use of various process variables including biomass activity, influent composition, and toxicity, but the literature is unclear as to the utility of these algorithms in control systems. Controversies like this reinforce the need to devise an effective and unbiased evaluation method that can be used to judge the utility of different control strategies.

From a practical standpoint, it is not reasonable to experimentally test and verify the effectiveness of all reported control strategies and often the assessment of these control strategies is confounded by the multifaceted nature of the process under study. Alternatively, given a standardised procedure, it is possible to efficiently evaluate numerous strategies through realistic/dynamic computer simulations. Simulations provide a cost-effective means for the evaluation of control strategies, but the unlimited number of simulation permutations makes the need for a standardised protocol very important if different strategies (and different simulation results) are to be compared.

Each control strategy must be simulated under the same conditions to ensure unbiased comparisons. Validation of the computer simulations is difficult without supporting experimental or full-scale data, but the value of the work is enhanced through the use of accepted activated sludge models. Because appropriate simulation tools for the activated sludge process are available this approach has numerous advantages, but still there is a need for a standardised procedure. To this end, there has been a recent effort to develop a standardised simulation protocol - a *'simulation benchmark'*.

The idea to produce a standardised '*simulation benchmark'* was first devised and developed by the first *IAWQ Task Group on Respirometry-Based Control of the Activated Sludge Process* (Spanjers *et al.*, 1998a). This original benchmark was subsequently modified by the European Co-operation in the field of Scientific and Technical Research (COST) 682/624 Actions in co-operation with the second *IWA Respirometry Task Group* (Copp, 2000; Alex *et al.*, 1999; Pons *et al.*, 1999). In an attempt to standardise the simulation procedure and the evaluation of all types of control strategies, the two groups have jointly developed a consistent simulation protocol.

In this instance, the COST '*simulation benchmark'* is a comprehensive description of a standardised simulation and evaluation procedure including plant layout, simulation models and model parameters, a detailed description of disturbances to be applied during testing and evaluation criteria for testing the relative effectiveness of simulated strategies. The COST '*simulation benchmark*' is meant to provide an unbiased basis for comparing past, present and future control strategies without reference to a particular facility. This site independent tool is a fully defined wastewater treatment scenario. The simulation output generated with this modelled and uncontrolled scenario acts as a 'benchmark' from which to judge the impact of simulated control strategies. The power of this simulation tool becomes apparent when it is realised that because the *'simulation benchmark'* is a defined protocol all benchmark-implemented strategies can be compared, irrespective of control objective.

The *'simulation benchmark'*, by definition, must be independent of the simulation software being used. That is, the simulation software should have no impact on the modelling output such that different simulators modelling the same system should give the same result. However, because of the many simulator-specific options, this is not always the case, nor is it a trivial task to ensure similar results using different simulators. A substantial effort has gone into this aspect of the *'simulation benchmark'* development and a part of the development was a ring-test in which one and the same test was done with different simulators. By stipulating specific model equations, modelling procedures and simulator-specific options, similar results can be achieved. Cross-platform testing of the COST *'simulation benchmark'* has been successfully demonstrated using a number of commercially available simulation software tools and a user defined computer code. For users of the benchmark, demonstrating similar results in this way is the first step in the evaluation procedure, and ensures that the simulator being used is tuned according to the *'simulation benchmark'* specifications, which in turn should ensure the consistent comparison of control strategy results.

The purpose of this publication is to provide a description of the COST *'simulation benchmark'* as currently defined and provide specific information about implementing the benchmark into various simulation software packages.

2

Simulation Benchmark Overview

There is little doubt that control strategies can be evaluated by model simulation. However, the protocol used in the evaluation is critical and must be defined in such a way as to ensure unbiased comparisons. To make unbiased comparisons, each control strategy must be evaluated under the same conditions. Also, the effect of the control strategy must be compared to a fully defined and suitable reference output. Only then is it possible to truly evaluate a control strategy and compare it with another strategy. The '*simulation benchmark*' defines such a protocol and provides a suitable reference output.

2.1 PLANT LAYOUT

The '*simulation benchmark*' plant design is comprised of five reactors in series with a 10-layer secondary settling tank. Figure 2.1 shows a schematic representation of the layout.

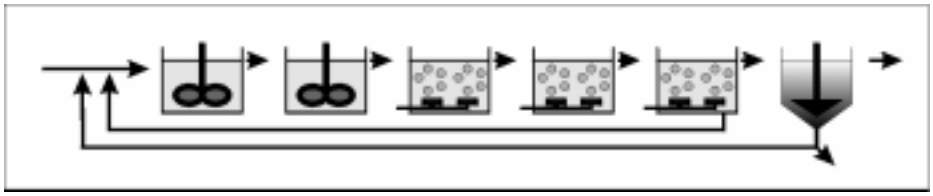

Figure 2.1: Schematic representation of the *'simulation benchmark'* configuration showing tanks 1 & 2 mixed and unaerated, and tanks 3, 4 & 5 aerated.

The layout is fully defined and has the following characteristic features:

- 5 biological tanks-in-series with a secondary settler
- total biological volume of 5999 m^3 (tanks 1 & 2 each 1000 m^3 and tanks 3, 4 & 5 each 1333 m^3)
- tanks 1 & 2 unaerated, but fully mixed
- aeration of tanks 3, 4 & 5 achieved using a *maximum* K_La of 10 hr^{-1}

- default K_La of 10 hr^{-1} in tanks 3 & 4 and 3.5 hr^{-1} in tank 5
- DO saturation in tanks 3, 4 & 5 of 8 gO_2 m^{-3}
- a non-reactive secondary settler with a volume of 6000 m^3 (area of 1500 m^2 and a depth of 4 m) subdivided into 10 layers
- a feed point to the settler at 2.2 m from the bottom (i.e. feed enters the settler in the middle of the sixth layer)
- two (2) internal recycles:
  - nitrate internal recycle from the 5th to the 1st tank at a default flow rate of 55338 m^3 d^{-1}
  - RAS recycle from the underflow of the secondary settler to the front end of the plant at a default flow rate of 18446 m^3 d^{-1} (as there is no biological reaction in the settler, the oxygen concentration in the recycle is the same as in the fifth tank reactor)
- WAS is pumped continuously from the secondary settler underflow at a default rate of 385 m^3 d^{-1}

The physical attributes of the biological reactors and the settler are listed Table 2.1 and a selection of system variables are listed in Table 2.2.

Table 2.1: Physical attributes of the biological reactors and settling tank for the COST *'simulation benchmark'* process configuration.

	Physical Configuration	Units
Volume - Tank 1	1000	m^3
Volume - Tank 2	1000	m^3
Volume - Tank 3	1333	m^3
Volume - Tank 4	1333	m^3
Volume - Tank 5	1333	m^3
Depth - Settler	4	m
Area - Settler	1500	m^2
Volume - Settler	6000	m^3

Table 2.2: A selection of system variables.

	Default System Flow Rates	Units
Influent flow rate	18446	m^3 day^{-1}
Recycle flow rate	18446	m^3 day^{-1}
Internal recycle flow rate	55338	m^3 day^{-1}
Wastage flow rate	385	m^3 day^{-1}
K_La - Tank 1	n/a	-
K_La - Tank 2	n/a	-
K_La - Tank 3	10	hr^{-1}
K_La - Tank 4	10	hr^{-1}
K_La - Tank 5	3.5	hr^{-1}

2.2 PROCESS MODELS

To increase the acceptability of the results, two internationally accepted process models were chosen. The IAWQ's Activated Sludge Model #1 (ASM1) was chosen as the biological process model (Henze *et al.*, 1987) and the double-exponential settling velocity function of Takács *et al.* (1991) was chosen as a fair representation of the settling process.

2.2.1 Biological Process Model

It should be noted that since ASM1 was first introduced several modifications have been suggested such that now a number of activated sludge models exist including ASM2, ASM2d and most recently ASM3. However, unlike ASM1, the newer models have yet to be fully embraced by the international community. There are several limitations with ASM1, but its universal appeal and practical verification overshadow these limitations. A matrix representation of ASM1 (Henze *et al.*, 1987) is shown in Figure 2.2.

Figure 2.2 shows that ASM1 has 13 components (state variables) and 8 processes. This model representation is included here as a reference only. A complete description of the model and its development are available elsewhere (Henze *et al.*, 1987). Table 2.3 lists the ASM1 state variables, the associated symbols and the state variable units.

Table 2.3: State variables for the IAWQ Activated Sludge Model #1 (ASM1)

State Variable Description	State Symbol	Units
Soluble inert organic matter	S_I	g COD m^{-3}
Readily biodegradable substrate	S_S	g COD m^{-3}
Particulate inert organic matter	X_I	g COD m^{-3}
Slowly biodegradable substrate	X_S	g COD m^{-3}
Active heterotrophic biomass	$X_{B,H}$	g COD m^{-3}
Active autotrophic biomass	$X_{B,A}$	g COD m^{-3}
Particulate products arising from biomass decay	X_P	g COD m^{-3}
Oxygen	S_O	g COD m^{-3}
Nitrate and nitrite nitrogen	S_{NO}	g N m^{-3}
$NH_4^+ + NH_3$ nitrogen	S_{NH}	g N m^{-3}
Soluble biodegradable organic nitrogen	S_{ND}	g N m^{-3}
Particulate biodegradable organic nitrogen	X_{ND}	g N m^{-3}
Alkalinity	S_{ALK}	mol L^{-1}

The matrix representation shows the stoichiometric relationships that relate the state variables to the process rate equations. By using this representation, it is possible to easily identify the various parameters involved in the model. To ensure the consistent application of the model in benchmarking studies, all of the kinetic and stoichiometric model parameters have been defined in the *'simulation benchmark'* description. The stoichiometric parameter values to be used are listed in Table 2.4 and the kinetic parameter values are listed Table 2.5. Included in these tables are the parameter descriptions, their recognised symbols and units as well as an associated value. The listed parameter estimates approximate those that are expected at 15°C.

Figure 2.2: Matrix representation of ASM1 showing the processes, components, process rate equations, and stoichiometry (Henze *et al*., 1987).

Component j	1 S_I	2 S_S	3 X_I	4 X_S	5 $X_{B,H}$	6 $X_{B,A}$	7 X_P	8 S_O	9 S_{NO}	10 S_{NH}	11 S_{ND}	12 X_{ND}	13 S_{ALK}	Process rate, ρ_j
i Process														
1 Aerobic growth heterotrophs		$-\frac{1}{Y_H}$			1			$-\frac{1-Y_H}{Y_H}$		$-i_{XB}$			$-\frac{i_{XB}}{14}$	$\mu_H\left(\frac{S_S}{K_S+S_S}\right)\left(\frac{S_O}{K_{OH}+S_O}\right)X_{B,H}$
2 Anoxic growth heterotrophs		$-\frac{1}{Y_H}$			1				$-\frac{1-Y_H}{2.86Y_H}$	$-i_{XB}$			$-\frac{1-Y_H}{14\times 2.86Y_H}-\frac{i_{XB}}{14}$	$\mu_H\left(\frac{S_S}{K_S+S_S}\right)\left(\frac{K_{OH}}{K_{OH}+S_O}\right)\left(\frac{S_{NO}}{K_{NO}+S_{NO}}\right)\eta_g X_{B,H}$
3 Aerobic growth autotrophs						1		$-\frac{4.57-Y_A}{Y_A}$	$\frac{1}{Y_A}$	$-i_{XB}-\frac{1}{Y_A}$			$-\frac{i_{XB}}{14}-\frac{1}{7Y_A}$	$\mu_A\left(\frac{S_{NH}}{K_{NH}+S_{NH}}\right)\left(\frac{S_O}{K_{OA}+S_O}\right)X_{B,A}$
4 Decay heterotrophs				$1-f_P$	-1		f_P					$i_{XB}+f_P i_{XP}$		$b_H X_{B,H}$
5 Decay autotrophs				$1-f_P$		-1	f_P					$i_{XB}+f_P i_{XP}$		$b_A X_{B,A}$
6 Ammonification										1	-1		$\frac{1}{14}$	$k_a S_{ND} X_{B,H}$
7 Hydrolysis organic compounds		1		-1										$k_H\frac{X_S/X_{B,H}}{K_X+X_S/X_{B,H}}\left[\left(\frac{S_O}{K_{OH}+S_O}\right)+\eta_H\left(\frac{K_{OH}}{K_{OH}+S_O}\right)\left(\frac{S_{NO}}{K_{NO}+S_{NO}}\right)\right]X_{B,H}$
8 Hydrolysis organic N											1	-1		$\rho_7(i_{XB}/i_{XS})$
Conversion rates	$r_1=\sum_j \nu_{1j}\rho_j$													

Table 2.4: Stoichiometric parameter values for ASM1 in the *'simulation benchmark'*.

Parameter Description	Parameter Symbol	Value	Units
Autotrophic yield	Y_A	0.24	g $X_{B,A}$ COD formed (g N utilised)$^{-1}$
Heterotrophic yield	Y_H	0.67	g $X_{B,H}$ COD formed (g COD utilised)$^{-1}$
Fraction of biomass to particulate products	f_P	0.08	dimensionless
Fraction nitrogen in biomass	i_{XB}	0.08	g N (g COD)$^{-1}$ in biomass ($X_{B,A}$ & $X_{B,H}$)
Fraction nitrogen in particulate products	i_{XP}	0.06	g N (g COD)$^{-1}$ in X_p

Table 2.5: Kinetic parameter values for ASM1 in the *'simulation benchmark'*.

Parameter Description	Parameter Symbol	Value	Units
Maximum heterotrophic growth rate	μ_{mH}	4.0	day^{-1}
Half-saturation (hetero. growth)	K_S	10.0	g COD m^{-3}
Half-saturation (hetero. oxygen)	K_{OH}	0.2	g O_2 m^{-3}
Half-saturation (nitrate)	K_{NO}	0.5	g NO_3-N m^{-3}
Heterotrophic decay rate	b_H	0.3	day^{-1}
Anoxic growth rate correction factor	η_g	0.8	dimensionless
Anoxic hydrolysis rate correction factor	η_h	0.8	dimensionless
Maximum specific hydrolysis rate	K_h	3.0	g X_s (g $X_{B,H}$ COD·day)$^{-1}$
Half-saturation (hydrolysis)	K_X	0.1	g X_s (g $X_{B,H}$ COD)$^{-1}$
Maximum autotrophic growth rate	μ_{mA}	0.5	day^{-1}
Half-saturation (auto. growth)	K_{NH}	1.0	g NH_3-N m^{-3}
Autotrophic decay rate	b_A	0.05	day^{-1}
Half-saturation (auto. oxygen)	K_{OA}	0.4	g O_2 m^{-3}
Ammonification rate	k_a	0.05	m^3 (g COD day)$^{-1}$

2.2.2 Settling Process Model

As with the biological process model, international acceptability was the overriding criteria for choosing a settling model. The double-exponential settling velocity function of Takács *et al.* (1991) is based on the solids flux concept, and is applicable to both hindered and flocculent settling conditions, unlike the standard Vesilind model (Vesilind, 1968), which is applicable only under hindered settling conditions. Equation 2.1 shows the Takács double-exponential settling velocity function. As with the biological model, a number of parameters are used in the function and these have been fully defined in the *'simulation benchmark'* description (Table 2.6). Table 2.6 lists the parameters, giving a description of each parameter, an associated symbol and the parameter units.

$$v_{sj} = v_o e^{-r_h X_j^*} - v_o e^{-r_p X_j^*} \qquad (2.1)$$
$$0 \le v_{sj} \le v'_o$$

where:
- v_{sj} is the settling velocity in layer j (m day^{-1})
- X_j^* is the suspended solids concentration in layer *j* (g m^{-3}), subject to the limiting condition that ($X_j^* = X_j - X_{min}$)
- X_j is the suspended solids concentration in layer *j* (g m^{-3})

- X_{min} is the minimum attainable suspended solids concentration (g m^{-3}) calculated from $X_{min} = f_{ns} \bullet X_{in}$ [where: X_{in} is the mixed liquor suspended solids concentration entering the settling tank, and *fns* is the non-settleable fraction]

Table 2.6: Settler model parameters and default values.

Parameter Description	Parameter Symbol	Value	Units
Maximum settling velocity	v'_o	250	m day^{-1}
Maximum Vesilind settling velocity	v_o	474	m day^{-1}
Hindered zone settling parameter	r_h	0.000576	m^3 (g SS)$^{-1}$
Flocculant zone settling parameter	r_p	0.00286	m^3 (g SS)$^{-1}$
Non-settleable fraction	f_{ns}	0.00228	dimensionless

2.3 INFLUENT COMPOSITION

It has already been stated that the disturbances used to test a particular control strategy play a critical role in the evaluation. That is, because of the multifaceted nature of activate sludge, a particular control strategy may react well to one disturbance and not well to another. Hence for an unbiased and complete evaluation, it is important that a series of disturbances be defined and that each control strategy be subjected to all the disturbances. Only then can a fair comparison be made. To this end, several influent file disturbances have been defined in the '*simulation benchmark*' description (Copp, 1999; Vanhooren and Nguyen, 1996). In total, there are three influent disturbances and each is meant to be representative of a different weather condition. The data files are available for download from various sources including the COST 624 web site (http://www.ensic.inpl-nancy.fr/COSTWWTP/).

Each of the files contains 14 days of influent data at 15-minute intervals. The data included in the files are listed in the following order: time; S_S; $X_{B,H}$; X_S; X_I; S_{NH}; S_I; S_{ND}; X_{ND}; Q with influent S_O; $X_{B,A}$; X_P; and S_{NO} assumed to be zero. The final component, S_{ALK}, is given a default value of 7 mol m^{-3} for the entire 14-day period. In general, these files depict expected diurnal variations in influent flow and COD. As well, expected trends in weekly data have been incorporated. That is, much lower peak flows are depicted in the 'weekend' data, which is consistent with *normal* load behaviour at a municipal treatment facility.

The files are representative of three disturbances: *dry weather*, a *storm event* and a *rain event*. The first file is a *dry weather* file and depicts what is considered to be *normal* diurnal variations in flow and COD load. In this file, the resultant peaking factor is 1.74 for maximum flow and 2.34 for maximum COD mass load (i.e. flow x concentration, mass/day) as compared to the flow-weighted average values. The second file is a variation on the first with the incorporation of two storm events. The first storm event in this file is of high intensity and short duration and is expected to *flush* the sewer of particulate material. The resuspension of these particles is reflected in the data through a significant increase in inert and biodegradable suspended solids. The second storm event assumes the sewers were cleared of particulate matter during the first storm event; hence, only a modest increase in COD load is noted during the second storm. This result occurs even

though the peak flow for both storms is the same and the peak flow of the second storm is maintained over a longer period of time. The third file is meant to represent a long rain event. The influent flow during this rain event does not reach the level attained during the storm events, but the increased flow is sustained for a much longer period of time. Unlike the storm events, there is no increase in COD load to the plant during the rain event. The flow-weighted average concentration of the influent components for the three files are shown in Table 2.7.

Table 2.7: Flow-weighted average influent composition in the influent files.

Component	*dry weather*	*storm event*	*rain event*	Units
S_S	69.50	64.93	60.13	g COD m^{-3}
$X_{B,H}$	28.17	27.25	24.37	g COD m^{-3}
X_S	202.32	193.32	175.05	g COD m^{-3}
X_I	51.20	51.92	44.30	g COD m^{-3}
S_{NH}	31.56	29.48	27.30	g N m^{-3}
S_I	30.00	28.03	25.96	g COD m^{-3}
S_{ND}	6.95	6.49	6.01	g N m^{-3}
X_{ND}	10.59	10.24	9.16	g N m^{-3}
Q	18446	19745	21320	m^3 day^{-1}

2.4 SIMULATION PROCEDURE

A two-step simulation procedure is defined in the *'simulation benchmark'* description and involves simulations to steady state followed by dynamic simulations using the three influent data files described in the previous section. Here again, the description is rigid to ensure the consistent application of the benchmark and to ensure that similar analyses are done on the output data.

2.4.1 Steady State Simulations

The initial step in the simulation procedure is to simulate the system under study to steady state using an influent of constant flow and composition. The flow-weighted *dry weather* data (Table 2.7) is used for this purpose and steady state is defined using either the software steady state solver (in GPS-X™, use an iteration termination criterion of 0.1) or by simulating 100 days using a constant influent. All dynamic simulations should follow a steady state simulation. This ensures a consistent starting point and should eliminate the influence of starting conditions on the generated dynamic output.

2.4.2 Dynamic Simulations

Next, dynamic simulations should be performed using the influent files described previously. The implementation of these dynamic simulations will vary with the simulator being used, however a general overview of the required procedure is outlined in this section.

Starting from the steady state solution, using the *dry weather* influent file as a dynamic input, the system under study should be simulated for 14 days. The resulting state variable values should

then be saved (if possible, in the simulator being used) for all unit processes. These state variable values represent the starting point for evaluating the dynamic response of the plant to each of the influent disturbance files. From the state achieved above, simulate a further 14 days using the *dry weather*, *storm event* and *rain event* influent files in separate simulation studies, but each time starting from the state achieved after the initial 14-day *dry weather* simulation. That is, for any one system at steady state, there are three 28-day dynamic simulations to perform: *dry-dry*, *dry-storm* and *dry-rain*.

The output data generated from the simulations described above is used to examine the dynamic performance of the process. The data of interest from these dynamic simulations is the data generated during the last 7 days of dynamic simulation. That is, if the 28-day simulations are considered, the data of interest is from day 22 to day 28 inclusive and includes three data sets: one for the *dry weather* simulation, one for the *storm event* simulation and one for the *rain event* simulation. Output data should be recorded at 15-minute intervals (i.e. a total of 4 x 24 x 7 data entries) for each variable of interest.

2.5 PERFORMANCE INDEX

Use of the weather files enables the examination of the dynamic behaviour of the system and/or control strategy under study and the simulation procedure outlined above is meant to ensure that any data analysed by '*simulation benchmark*' users is generated in a similar manner. Nevertheless, the result of these dynamic simulations leads to further questions; including how is the huge amount of output data to be evaluated. To aid the evaluation process, a performance index has been developed for comparing the dynamic responses and specifically for comparing the impact of different control strategies.

Because of the extensive amount of raw dynamic output data and the fact that that data may vary from simulator to simulator, the dynamic results are compared using a number of performance indices. The performance index, as a whole, is a set of geographically independent measures that combine the output data into a small number of composite terms. These composite terms include, among others, a general effluent quality measure, energy terms for pumping and aeration, and a measure of sludge production. The equations needed to calculate these terms are outlined below.

The system performance assessment included in the performance index is made at two levels. The first level concerns the process performance and the second level concerns the local control loops.

2.5.1 Process Assessment

The first level of assessment quantifies the effect of the control strategy on plant process performance and can be divided into three sub-levels:

- effluent quality index
- effluent violations
- operational costs

Definition of Composite Variable Calculations:

$$TSS_e = 0.75\,(X_{S,e} + X_{BH,e} + X_{BA,e} + X_{P,e} + X_{I,e})$$
$$COD_e = S_{S,e} + S_{I,e} + X_{S,e} + X_{BH,e} + X_{BA,e} + X_{P,e} + X_{I,e}$$
$$BOD_e = 0.25\,(S_{S,e} + X_{S,e} + (1 - f_p)\,(X_{BH,e} + X_{BA,e}))$$
$$TKN_e = S_{NH,e} + S_{ND,e} + X_{ND,e} + i_{XB}\,(X_{BH,e} + X_{BA,e}) + i_{XP}\,(X_{P,e} + X_{I,e})$$
$$NO_e = S_{NO,e}$$
$$N_{tot,e} = TKN_e + NO_e$$

2.5.1.1 Effluent Quality Index

Within the context of the '*simulation benchmark*', effluent quality is considered through an effluent quality index (EQ), which is meant to quantify into a single term the effluent pollution load to a receiving water body.

Effluent quality (EQ, in units of kg pollution units d^{-1}): calculated as follows by integrating through the final 7 days of weather simulations (T = 7 days):

$$EQ = \frac{1}{T \bullet 1000}\int_{t_o}^{t_{7\,days}} [PU_{TSS}(t) + PU_{COD}(t) + PU_{BOD}(t) + PU_{TKN}(t) + PU_{NO}(t)]\,Q_e(t)\,dt \quad (2.2)$$

where:

		β_i factors
$PU_{TSS}(t) = \beta_{TSS}\,TSS_e(t)$	β_{TSS} =	2
$PU_{COD}(t) = \beta_{COD}\,COD_e(t)$	β_{COD} =	1
$PU_{BOD}(t) = \beta_{BOD}\,BOD_e(t)$	β_{BOD} =	2
$PU_{TKN}(t) = \beta_{TKN}\,TKN_e(t)$	β_{TKN} =	20
$PU_{NO}(t) = \beta_{NO}\,NO_e(t)$	β_{NO} =	20

As a check on the EQ calculation, an influent quality index (IQ) can be calculated. To calculate the IQ, apply the above equations to the influent files, but change the BOD coefficient from 0.25 to 0.65. For reference purposes, the IQ is normally included in a dynamic performance report.

NOTE: The β_i factors in the table above were determined based, in part, on empirical effluent component weightings. The above weightings are based on a paper by Vanrolleghem *et al.* (1996) that cited a Flanders effluent quality formula for calculating fines. That formula is based on several terms including terms for organics, nutrients, metals, and heat. The metal and heat terms are not of interest to the benchmark, but the organic and nutrient terms are applicable. Using the steady state data for each of the layouts it is possible to calculate the organic and nutrient terms based on the Flanders equation. From these terms it is then possible to determine the specific fraction that each term makes up of the fine formula i.e. %nutrients = $N_{nutrients}$ / ($N_{nutrients}$ + $N_{organics}$). The β_i factors above were chosen to reflect these calculated fractions. For the COST '*simulation benchmark*' layout, the steady state EQ was found to be weighted as 22% nutrients and 78% organics.

2.5.1.2 Effluent Violations

Included in the performance evaluation is a measure of effluent violations. Constraints with respect to five effluent components are defined and the percentage of time that the constraints are not met is to be reported. As well, the methodology for reporting the number of violations is defined. The violations are calculated for five terms: ammonia, total nitrogen, BOD_5, total COD and suspended solids and the effluent constraints chosen for these five terms are as follows:

Table 2.8: Effluent constraints for the five violation variables.

		Adopted Effluent Constraints	Units
Ammonia	$S_{NH,e}$	4	gN m^{-3}
Total Nitrogen	$N_{tot,e}$	18	gN m^{-3}
BOD_5	BOD_e	10	gBOD m^{-3}
Total COD	COD_e	100	gCOD m^{-3}
Suspended Solids	TSS_e	30	gSS m^{-3}

The effluent violations are reported through two quantities: (i) number of violations; and, (ii) % time plant is in violation. These quantities are calculated from the output data generated at 15-minute intervals.

Number of violations:
This quantity represents the number of times that the plant is in violation of the effluent constraints (i.e. the number of times the plant effluent increases above the effluent constraint). This measure does *not* necessarily reflect the length of time that the plant is in violation.

% time plant in violation:
This quantity is a measure of the percentage of the time that the plant is in violation of the effluent constraints.

2.5.1.3 Operational Variables

Operational issues are considered through three items: sludge production, pumping energy and aeration energy (integrations performed on the final 7 days of weather simulations (i.e. from day 22 to day 28 of weather file simulations, T = 7 days)).

Sludge production: - (i) sludge for disposal; and, (ii) total sludge production

(i) sludge for disposal (in units of kg d^{-1})

$$P_{sludge} = [\ \Delta M(TSS_{system}) + M(TSS_w)\ ] / T \qquad (2.3)$$

where:
$\Delta M(TSS_{system})$ = change in system sludge mass from the end of day 21 to the end of day 28

$$\Delta M(TSS_{system}) = M(TSS_{system})_{end\ of\ day\ 28} - M(TSS_{system})_{end\ of\ day\ 21}$$

$$M(TSS_{system}) = M(TSS_{reactors}) + M(TSS_{settler})$$

$$M(TSS_w) = 0.75 \int_{t_o}^{t_{7\ days}} [X_{S,w}(t) + X_{BH,w}(t) + X_{BA,w}(t) + X_{P,w}(t) + X_{I,w}(t)] Q_w(t)\, dt$$

(ii) total sludge production (in units of kg d^{-1})

$$P_{total\_sludge} = P_{sludge} + M(TSS_e) / T \tag{2.4}$$

where:

$$M(TSS_e) = 0.75 \int_{t_o}^{t_{7\ days}} [X_{S,e}(t) + X_{BH,e}(t) + X_{BA,e}(t) + X_{P,e}(t) + X_{I,e}(t)] Q_e(t)\, dt$$

Pumping energy: (in units of kWh d^{-1})

$$PE = \frac{0.04}{T} \int_{t_o}^{t_{7\ days}} [Q_a(t) + Q_r(t) + Q_w(t)]\, dt \tag{2.5}$$

where:
$Q_a(t)$ = internal recycle flow rate at time t ($m^3\ d^{-1}$)
$Q_r(t)$ = return sludge recycle flow rate at time t ($m^3\ d^{-1}$)
$Q_w(t)$ = waste sludge flow flow rate at time t ($m^3\ d^{-1}$)

Aeration energy: (in units of kWh d^{-1})

$$AE = \frac{24}{T} \int_{t_o}^{t_{7\ days}} \sum_{i=1}^{i=5} [0.4032\, K_L a_i(t)^2 + 7.8408\, K_L a_i(t)]\, dt \tag{2.6}$$

where:
$K_L a_i(t)$ = the mass transfer coefficient in i^{th} aerated reactor at time t (in units of hr^{-1})

2.5.2 Controller Assessment

The second level of assessment quantifies the effect of the control strategy on controller performance and can be divided into two sub-levels:

- controlled variable performance
- manipulated variable performance

The following sections present the equations for calculating the assessment terms.

2.5.2.1 Controlled Variable Performance

IAE (Integral of the Absolute Error):

$$IAE_i = \int_{t_o}^{t_{7\ days}} \left| e_j \right| dt \tag{2.7}$$

where: e_j is the error in the controlled variable ($e_j = Z_{j,setpoint} - Z_{j,observed}$)
(note: subscript j is meant to distinguish different controlled variables in the same system)

ISE (Integral of the Squared Error):

$$ISE_j = \int_{t_o}^{t_{7\ days}} e_j{}^2\, dt \tag{2.8}$$

Maximum deviation from setpoint:

$$\max\left(Dev_j^{error}\right) = \max\left| e_j \right| \tag{2.9}$$

Variance in the controlled variable error:

$$Var\left(e_j\right) = \overline{e_j{}^2} - \left(\overline{e_j}\right)^2 \tag{2.10}$$

where:

$$\overline{e_j} = \frac{\int_{t_o}^{t_{7\ days}} e_j\, dt}{T} \qquad \overline{e_j{}^2} = \frac{\int_{t_o}^{t_{7\ days}} e_j{}^2\, dt}{T}$$

2.5.2.2 Manipulated Variable Performance

Maximum deviation in the manipulated variable:

$$\max\left(Dev_j^{MV}\right) = u_{j,\max} - u_{j,\min} \tag{2.11}$$

where: u_j is the value of the manipulated variable (*MV*) and the minimum and maximum are determined during the 7 days of interest defined above (note: the subscript j is meant to distinguish different manipulated variables in the same system)

Maximum deviation in the change in manipulated variable:

$$\max\left(Dev_j^{\Delta u_j}\right) = \max\left(\Delta u_j\right) \tag{2.12}$$

where:

$$\Delta u_j = \left| u_j(t+dt) - u_j(t) \right|$$

Variance in the change in manipulated variable:

$$Var\left(\Delta u_j\right) = \overline{\Delta u_j^{\,2}} - \left(\overline{\Delta u_j}\right)^2 \tag{2.13}$$

where:

$$\overline{\Delta u_j} = \frac{\int_{t_o}^{t_{7\,days}} \Delta u_j \, dt}{T} \qquad \overline{\Delta u_j^{\,2}} = \frac{\int_{t_o}^{t_{7\,days}} \Delta u_j^{\,2} \, dt}{T}$$

Although the performance index is meant to be geographically independent, the structure of the performance index allows for location specific criteria to be defined in subsequent analyses. That is, the performance measures described above *MUST* be calculated for each strategy simulation, but emphasis can be placed on specific performance terms depending on location specific criteria if a user so wishes. For example, for a particular user if effluent quality is of primary concern irrespective of overall costs then the analysis of the performance index terms can be weighted accordingly. Alternatively, in another location where reducing overall costs is the primary objective, the index can be tailored to that situation. This structure allows for substantial flexibility in applying the '*simulation benchmark*' to specific control strategies, while at the same time providing a means to make meaningful location specific comparisons and design decisions.

3

Simulator Tuning

This section and the sections that follow provide information on the tuning of specific simulation software tools according to the '*simulation benchmark*' specifications. That is, although the '*simulation benchmark*' is meant to be platform-independent, simulator-specific options make obtaining the same results using different simulators difficult even when simulating the same system using the same process models. However, by stipulating specific model equations, modelling procedures and simulator-specific options, similar results can be achieved.

For users, duplicating these steady state and dynamic results is an essential first step in the evaluation procedure. By synchronising the simulation tool, users ensure that the simulator being used is tuned in an appropriate way, which in turn should ensure the consistent comparison of process behaviour and the consistent comparison of implemented control strategies.

3.1 STEADY STATE TUNING

Once the benchmark configuration has been set-up in the simulator of choice (as defined in Chapter 2), the initial step in this tuning procedure is to simulate the uncontrolled plant to steady state using an influent of constant flow and composition. As described above, the flow-weighted *dry weather* data is used for this purpose and steady state is defined using either the software steady state solver or by simulating 100 days using a constant influent.

Following simulation to steady state, the generated output data must be compared to the standardised output that is included in the benchmark description (Table 3.1). The standardised steady state output results listed in this section have been duplicated using BioWin™, EFOR™, GPS-X™, Matlab/Simulink™, Simba™, STOAT™, WEST™ and a user defined FORTRAN code and for that reason are assumed to be correct [A full listing of all the steady state results generated with the different simulators can be found in Appendix 13.1.]. It is assumed that the simulator and associated models being used have been input correctly once similar steady state results have been attained. Users that do not generate these steady state results must re-examine their set-up looking for errors. Note that tuning experience has shown that these discrepancies may be the result of

user-input errors (i.e. incorrect parameters, incorrectly specified flow rates…) or simulator-specific options. Users may need to examine both possibilities to find a particular problem.

NOTE: It should be noted here that the data presented in Table 3.1 was generated using the default K_La of 3.5 hr^{-1} in the last tank. Discrepancies will result if the maximum K_La of 10 hr^{-1} is used in these uncontrolled steady state simulations.

Table 3.1: *'Simulation benchmark'* steady state simulation results - *dry weather* influent file.

Component	Tank 1	Tank 5	Settler Underflow	Effluent	Units
VSS	2959.7	2945.9	5760.5	11.25	g m^{-3}
TSS	3285.2	3269.8	6393.9	12.50	g m^{-3}
S_I	30	30	30	30	g COD m^{-3}
S_S	2.81	0.89	0.89	0.89	g COD m^{-3}
X_I	1149.2	1149.2	2247.1	4.39	g COD m^{-3}
X_S	82.14	49.31	96.42	0.19	g COD m^{-3}
$X_{B,H}$	2551.8	2559.4	5004.7	9.78	g COD m^{-3}
$X_{B,A}$	148.4	149.8	292.9	0.57	g COD m^{-3}
X_P	448.9	452.2	884.3	1.73	g COD m^{-3}
S_O	0	0.49	0.49	0.49	g COD m^{-3}
S_{NO}	5.37	10.42	10.42	10.42	g N m^{-3}
S_{NH}	7.92	1.73	1.73	1.73	g N m^{-3}
S_{ND}	1.22	0.69	0.69	0.69	g N m^{-3}
X_{ND}	5.28	3.53	6.90	0.013	g N m^{-3}
OUR	1.49	31.87	-	-	g m^{-3} hr^{-1}
			Steady State Retention Times		
Solids retention time (SRT)				9.18	days
Hydraulic retention time (HRT)				15.61	hours

NOTE: An absolute error tolerance of 0.01g m^{-3} is deemed acceptable for state variables less than 0.1g m^{-3} and a tolerance limit of 0.5% has been set for all state variables greater than 0.1g m^{-3}. If the achieved results do not fall within those tolerances, users are advised to re-examine their set-up looking for possible errors.

Once acceptable steady state values have been achieved, users are encouraged to perform the dynamic simulations to further test the tuning of their simulator.

3.2 DYNAMIC SIMULATIONS

A series of dynamic simulations should be performed as described previously in Section 2.4.2 using the uncontrolled plant and the three dynamic influent files. Then, using the generated data,

the performance indices should be calculated. Users are advised to compare their performance results with the corresponding performance results included in the benchmark description. The recognised performance index results are listed in Table 3.2 and a complete listing of the compiled dynamic results is included in Appendix 13.2. Once acceptable dynamic results are achieved, the user can be reassured that the simulator being used is tuned in accordance with the benchmark specifications.

Table 3.2: Recognised dynamic performance index results for the uncontrolled benchmark plant using the three influent data files.

Performance Index Variable	Recognised Results			Units
	dry weather	*storm event*	*rain event*	
Influent Quality (IQ)	42043	42043	42043	kg PU d^{-1}
Effluent Quality (EQ)	7067	7993	8840	kg PU d^{-1}
Sludge for Disposal	2436	2600	2353	kg SS d^{-1}
Total Sludge Production	2671	2915	2737	kg SS d^{-1}
Aeration Energy	6476	6476	6476	kWh d^{-1}
Pumping Energy	2967	2967	2967	kWh d^{-1}
Ammonia (eff. limit 4g m^{-3})				
Number of violations	7	7	7	
Time in violation	62.50	64.43	63.39	% of time
Total Nitrogen (eff. limit 18g m^{-3})				
Number of violations	5	4	3	
Time in violation	8.18	8.48	4.46	% of time
BOD_5 (eff. limit 10g m^{-3})				
Number of violations	0	0	0	
Time in violation	0.00	0.00	0.00	% of time
Total COD (eff. limit 100g m^{-3})				
Number of violations	0	0	0	
Time in violation	0.00	0.00	0.00	% of time
Suspended Solids (eff. limit 30g m^{-3})				
Number of violations	0	1	0	
Time in violation	0.00	0.15	0.00	% of time

A substantial amount of work has gone into evaluating the dynamic responses of various software packages and it has been determined that it is not realistic for each simulator to produce precisely the same instantaneous dynamic results (unlike the steady state condition which should be reproducible using all simulators). For instance, Figure 3.1 shows the dynamic output from three different simulators. The differences illustrated in the figure are the result of the different means used to propagate soluble components through the settler. In these three instances the particulate components are modelled in precisely the same way, but the soluble components are modelled differently. As it is not always feasible to alter the specific features of certain models in commercially available simulators, some consideration has to be given to the analysis of dynamic data generated using a variation to the defined system. In this example the variation is the number

of settler layers used for soluble components. At steady state, these differences make no difference, but the effect becomes apparent under dynamic conditions.

In this instance, users have to qualitatively as well as quantitatively evaluate their dynamic simulation results with the results included in the *'simulation benchmark'* description. Using the performance index terms as a measure of the simulated dynamic behaviour, it is possible to determine if the particular simulator being used is dynamically synchronised with the output of the many simulators that have verified the available dynamic performance data.

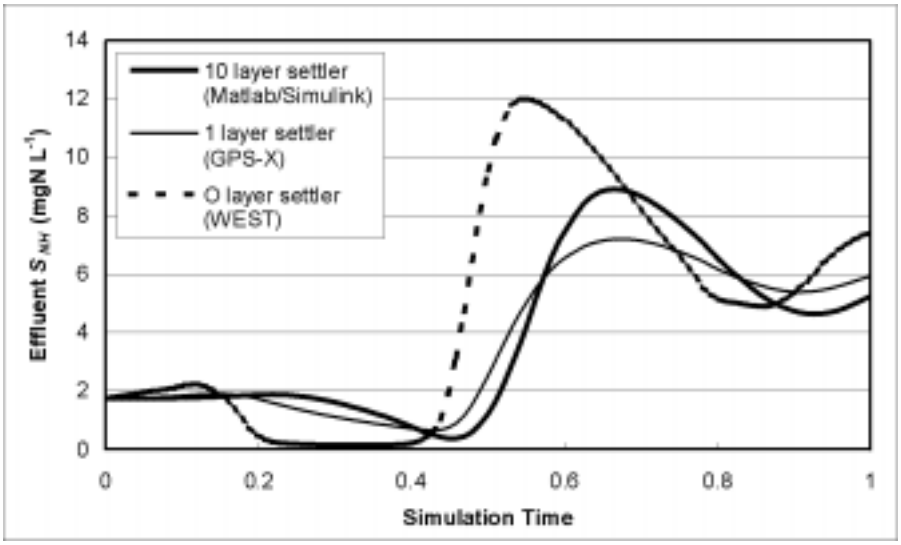

Figure 3.1: An example of three dynamic outputs produced by three different default settler models for soluble components in three different simulators [note: In addition to the 0-layer model, the 10-layer model has been implemented in WEST®, giving precisely the same results as illustrated for Matlab/Simulink. The 10-layer soluble model is defined in the benchmark.].

The tolerance limits for the dynamic simulations depend somewhat on the software being used and the exact models implemented. In particular, the method used to propagate the soluble compounds through the settler (Figure 3.1) seems to be a recurring problem. That is, dynamic results should be compared to a similar dynamic reference output, if possible. For simulations using a 1-layer or a 10-layer soluble settler model, a tolerance limit of 0.5% has been set for all state variables and performance indices. For results generated with neither of these models, a qualitative evaluation will have to be performed. Nevertheless, even with the undefined soluble models, differences for the most part should not be more than 0.5% for all variables and indices. If the achieved results do not fall within this tolerance, users are advised to re-examine their set-up looking for possible errors.

3.3 BASIC CONTROL STRATEGY

Following the successful outcome of the dynamic simulations, users can attempt to implement the sample control strategy outlined below. This control strategy was designed as a means to test the benchmark description and evaluate the impact of user/simulator-defined control algorithms on the simulation results. The basic control strategy has two control loops.

3.3.1 Control Loop #1

The first loop involves controlling the dissolved oxygen (DO) level in the final compartment to a setpoint of 2.0 g m^{-3} by manipulation of the oxygen transfer coefficient (Figure 3.2). The DO sensor used in this first loop is assumed to be ideal with no delay or noise. Recall that the K_La in the last compartment is constrained to a maximum of 10 hr^{-1} (Section 2.1).

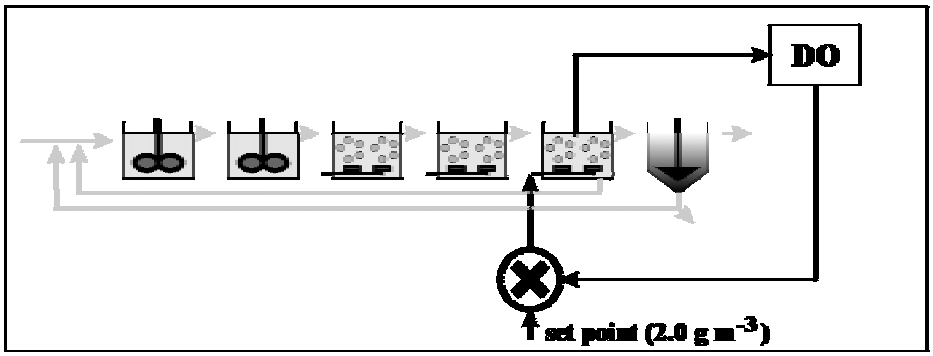

Figure 3.2: Basic control strategy; control loop #1.

3.3.2 Control Loop #2

The second control loop involves controlling the nitrate level in the second anoxic compartment (i.e. the second tank) to a setpoint of 1.0 g m^{-3} by manipulation of the internal recycle flow rate (Figure 3.3). In this loop, the nitrate sensor is assumed to have a time delay of 10 minutes, with white, normally distributed (standard deviation of 0.1 g m^{-3}), zero-mean noise. The internal recycle flow rate is constrained to a maximum of 92230 m^3 d^{-1} or 1.66 times the default rate.

To examine the effect of different simulators and different control loop implementations, the control strategy was implemented into a number of simulators. The nitrate controller performance results are shown in Table 3.3 [A complete listing of the control strategy results is in Appendix 13.3]. Only a portion of the performance results is shown, but the results illustrate that even with the fully defined benchmark plant and a well-defined control strategy, implementation of the same control strategy into different simulators may generate different results. In particular, tuning can have a large impact as can the criteria used during the tuning exercise. For instance, the ISE results in Table 3.3 indicate that the GPS-X controller is the most finely tuned. However, clearly

this controller could be more finely tuned if the maximum deviation from setpoint or standard deviation of the error is used as the tuning criteria (see WEST results). Unfortunately there is no clear solution to this problem and users should be aware of these types of problems when they use the '*simulation benchmark*' for strategy evaluations.

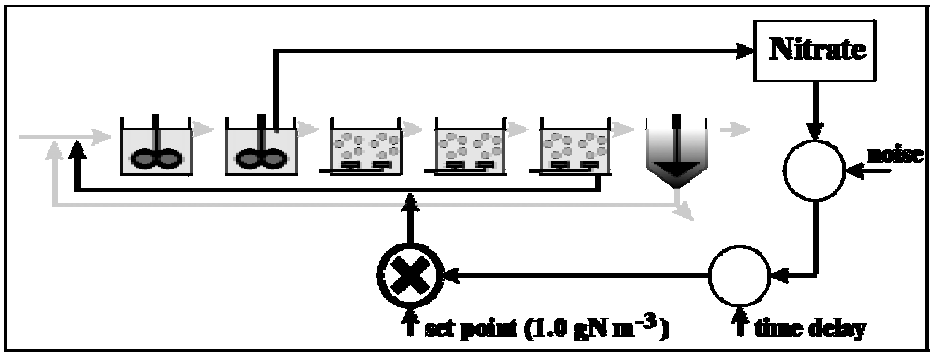

Figure 3.3: Basic control strategy control loop #2.

Table 3.3: Nitrate controller performance indices calculated from output data generated by three different simulators using the *dry weather* dynamic influent file with the 'basic control strategy' implemented into the *'simulation benchmark'*.

Nitrate Controller Performance (2nd tank)	GPS-X	Matlab/Simulink	WEST	Units
Controller type	velocity PI	cont PI with aw	PI	
Proportional gain (K)	7500	15000	10000	$m^3\ d^{-1}\ (g\ N\ m^{-3})^{-1}$
Integral time constant (T_i)	0.0125	0.05	0.01	d
Anti-windup time constant (T_t)	not used	0.03	not used	d
Controlled variable, S_{NO}				
Setpoint	1	1	1	$g\ N\ m^{-3}$
Integral of the absolute error (IAE)	0.185	1.482	0.829	$(g\ N\ m^{-3})*d$
Integral of the square error (ISE)	0.066	0.598	0.189	$(g\ N\ m^{-3})^2*d$
Max deviation from setpoint	0.883	0.887	0.652	$g\ N\ m^{-3}$
Standard deviation of the error	0.179	0.292	0.164	$g\ N\ m^{-3}$
Variance of the error	0.032	0.085	0.027	$(g\ N\ m^{-3})^2$
Manipulated variable, Q_a				
Max deviation in the MV (max-min)	49531	36691	46725	$m^3\ d^{-1}$
Max deviation in the MV (one 15-min interval)	10677	8078	9881	$m^3\ d^{-1}$
Standard deviation of delta MV	1623	1662	1554	$m^3\ d^{-1}$
Variance of delta MV	2632604	2762152	2414927	$(m^3\ d^{-1})^2$

The chapters that follow address specific features and options of specific simulators. Knowledge of this features and options is crucial, for benchmark users that are making use of one of these simulators in conjunction with the '*simulation benchmark*'. That is, tuning of specific simulators is not always straightforward and the following chapters have been written to disseminate the lessons we have learned about these simulators to all '*simulation benchmark*' users.

4

BioWin™

described by John B. Copp

NOTE: The issues and procedures outlined in this chapter are specific for the use of BioWin with the '*simulation benchmark*' and in no way should be interpreted as necessary procedures for using BioWin for any other purpose.

BioWin is a dedicated process simulator that makes use of linked process units to simulate biological wastewater treatment systems. It has been developed as a Microsoft Windows™ application and runs on an IBM PC-type computer. Implementation of the '*simulation benchmark*' into this simulator must take into consideration a number of BioWin-specific features that are not entirely consistent with the benchmark description. That is, to achieve benchmark consistent results, users must be aware of how to overcome the differences between BioWin and the rigidly defined benchmark description.

The main differences relate to the BioWin models because the benchmark specified models are not explicitly available. However, because the structure of the BioWin models are not significantly different from those defined in the benchmark, the benchmark models can be approximated by the BioWin models through manipulation of BioWin model parameters. The biological model is easily transformed, but the settler model impact is more difficult to overcome. Nevertheless, the settler model impact can be compensated for through several model, configuration and simulation alterations.

In addition to the model differences, there are several BioWin-specific features that '*simulation benchmark*' users need to be aware of when tuning BioWin to the benchmark specifications.

BioWin™ is a trademarked product of:
EnviroSim Associates Ltd., 7 Innovation Drive, Suite 205, Flamborough, Ontario, CANADA L9H 7H9
tel: +1 (905) 648-9814
fax: +1 (905) 338-5817
web: www.envirosim.com

These features include issues related to aeration, and setting up the proper structure for dynamic influent files. The following sections outline what BioWin users need to do to tune their software tool to the benchmark specifications and thus achieve the standardised benchmark results.

4.1 MODEL ISSUES - BIOWIN

The *'simulation benchmark'* specifies two process models: one for the biological processes and one for the settling process. For the biological processes, ASM1 (Henze *et al.*, 1987) is specified and for the settling process, the double-exponential settling function of Takács *et al.*, (1991). Unfortunately, neither of these models is explicitly available in BioWin. Rather, underlying the BioWin user interface are sedimentation models (both primary and secondary) and a comprehensive biological process model. The biological process model is an extension of the IAWQ's ASM1, and includes excess biological phosphorus removal (based on Dold, 1992). As the defined BioWin models do not differ significantly in structure from the benchmark specified models, it is possible to approximate the required models through manipulation of the BioWin model parameters. The following sections outline the necessary changes.

4.1.1 Biological Process Model

BioWin uses an extended version of ASM1 for its biological model. This extended model includes several additional features including biological phosphorus removal, but the BioWin user is able to choose between the full 'CNP' model and a reduced version of the model that includes only the carbon and nitrogen removal processes: the 'CN' model. The first step in the BioWin implementation procedure is to choose the 'CN' model rather than the default 'CNP' model. Although similar in structure to ASM1, the BioWin biological model uses different symbols for many of its state variables. Table 4.1 lists the ASM1 and BioWin 'CN' model symbols.

Table 4.1: Comparison of state variable symbols used in the IAWQ Activated Sludge Model #1 (ASM1) and in the BioWin 'CN' model (n/s not specified).

State Variable Description	ASM1 Symbol	BioWin Symbol	Units
Soluble inert organic matter	S_I	S_{US}	g COD m^{-3}
Readily biodegradable substrate	S_S	S_{BSC}	g COD m^{-3}
Particulate inert organic matter	X_I	X_I	g COD m^{-3}
Slowly biodegradable substrate	X_S	X_{SP}	g COD m^{-3}
Active heterotrophic biomass	$X_{B,H}$	Z_{BH}	g COD m^{-3}
Active autotrophic biomass	$X_{B,A}$	Z_{BA}	g COD m^{-3}
Particulate products arising from biomass decay	X_P	Z_E	g COD m^{-3}
Oxygen	S_O	S_O	g COD m^{-3}
Nitrate and nitrite nitrogen	S_{NO}	N_{O3}	g N m^{-3}
$NH_4^+ + NH_3$ nitrogen	S_{NH}	N_{H3}	g N m^{-3}
Soluble biodegradable organic nitrogen	S_{ND}	N_{OS}	g N m^{-3}
Particulate biodegradable organic nitrogen	X_{ND}	X_{ON}	g N m^{-3}
Alkalinity	S_{ALK}	S_{ALK}	mol L^{-1}
Phosphorus	-	P_S	g P m^{-3}
Inert suspended solids	-	n/s	g ISS m^{-3}

Table 4.2: *'Simulation benchmark'* stoichiometric parameter estimates for ASM1 and the BioWin 'CN' model (n/s not specified).

BioWin Description	ASM1 Symbol	Benchmark Value	BioWin Value	Units (using ASM#1 nomenclature where necessary)
Autotrophs				
Yield (Aerobic)	Y_A	0.24	0.24	g X_{BA} COD formed (g N oxidised)$^{-1}$
N in Biomass	i_{XB}	0.08	0.08	g N (g COD)$^{-1}$ in biomass (X_{BA} & X_{BH})
N in Inert	i_{XP}	0.06	0.06	g N (g COD)$^{-1}$ in X_p
P in Biomass	-	n/s	0.021	g P (g COD)$^{-1}$ in biomass (X_{BA} & X_{BH})
P in Inert	-	n/s	0.021	g P (g COD)$^{-1}$ in X_p
Endog. Residue	f_P	0.08	0.08	dimensionless
COD:VSS	f_{cv}	1.48	1.48	g COD g VSS^{-1}
Heterotrophs				
Yield (Aerobic)	Y_H	0.67	0.67	g X_{BH} COD formed (g COD utilised)$^{-1}$
N in Biomass	i_{XB}	0.08	0.08	g N (g COD)$^{-1}$ in biomass (X_{BA} & X_{BH})
N in Inert	i_{XP}	0.06	0.06	g N (g COD)$^{-1}$ in X_p
P in Biomass	-	n/s	0.021	g P (g COD)$^{-1}$ in biomass (X_{BA} & X_{BH})
P in Inert	-	n/s	0.021	g P (g COD)$^{-1}$ in X_p
Endog. Residue	f_P	0.08	0.08	dimensionless
COD:VSS	f_{cv}	1.48	1.48	g COD g VSS^{-1}
Yield (Anoxic)	-	n/s	0.67	g X_{BH} COD formed (g COD utilised)$^{-1}$
Adsorption Max	-	n/s	1.00	

Table 4.3: *'Simulation benchmark'* kinetic parameter estimates for ASM1 and the BioWin 'CN' model (n/s not specified).

BioWin Description	ASM1 Symbol	Benchmark Value	BioWin Value	Units (using ASM1 nomenclature where necessary)
Autotrophs				
Mu Max	μ_A	0.5	0.5	day^{-1}
Ks NH4	K_{NH}	1.0	1.0	g NH_3-N m^{-3}
Ba (endog.)	b_A	0.05	0.05	day^{-1}
Heterotrophs				
Mu Max	μ_H	4.0	4.0	day^{-1}
Ks COD	K_S	10.0	10.0	g COD m^{-3}
Bh	b_H	0.3	0.3	day^{-1}
Neta Anox. Hyd.	η_h	0.8	0.8	dimensionless
Neta Ana. Hyd.	η_h	0.8	0.8	dimensionless
Neta Anox Growth	η_g	0.8	0.8	dimensionless
Hydrolysis Rate	k_h	3.0	3.0	g X_S (g X_{BH} COD·day)$^{-1}$
Ks Hydrolysis	K_X	0.1	0.1	g X_S (g X_{BH} COD)$^{-1}$
Adsorption Ka	-	n/s	10.0	
Ferment. Rate	-	n/s	0.0	
Ferment Ks	-	n/s	5.0	
Ammonification	k_a	0.05	0.05	m^3 (g COD · day)$^{-1}$
Switching Functions				
Hetero. DO Limit	$K_{O,H}$	0.2	0.2	g O_2 m^{-3}
SND DO Limit	-	n/s	0.2	g O_2 m^{-3}
Auto. DO Limit	$K_{O,A}$	0.4	0.4	g O_2 m^{-3}
NH3 Limit	-	n/s	0.000	g NH_3-N m^{-3}
NO3 Limit	K_{NO}	0.5	0.5	g NO_3-N m^{-3}
Alk. Limit	-	n/s	0.01	mol L^{-1}

Further, the BioWin model uses several rate equations that vary slightly from the rate equations in ASM1 and makes use of several additional switching functions. Because of these variations, the following set of parameters should be used in BioWin to reduce the BioWin 'CN' model to the '*simulation benchmark*' defined ASM1. Table 4.2 lists the stoichiometric parameters and Table 4.3 lists the kinetic parameters to be used.

4.1.2 Settling Model

The double-exponential settling velocity function of Takács *et al.* (1991) was chosen as the '*simulation benchmark*' settling model due to its wide use and apparent acceptability as a fair representation of the settling process. The Takács model is based on the solids flux concept as is the standard Vesilind model (Vesilind, 1968), a modified version of which is used by BioWin. Equation 4.1 shows the Takács double-exponential settling velocity function specified in the benchmark, and Equation 4.2 shows the model used by BioWin.

$$v_{sj} = v_o e^{-r_h X_j^*} - v_o e^{-r_p X_j^*} \qquad (4.1)$$
$$0 \le v_{sj} \le v'_o$$

where:
- v_{sj} is the settling velocity in layer j (m d^{-1})
- X_j^* is the suspended solids concentration in layer j (g m^{-3}), subject to the limiting condition that ($X_j^* = X_j - X_{min}$)
- X_j is the suspended solids concentration in layer j (g m^{-3})
- X_{min} is the minimum attainable suspended solids concentration (g m^{-3}) calculated from $X_{min} = f_{ns} \bullet X_{in}$ [where: X_{in} is the mixed liquor suspended solids concentration entering the settling tank]

$$v_{sj} = v_o e^{-K' \bullet X_j} \bullet \left(\frac{X_j}{K_{ts} + X_j} \right) \qquad (4.2)$$

where:
- v_{sj} is the settling velocity in layer j (m d^{-1})
- v_o is the maximum settling velocity (g m^{-3})
- X_j is the suspended solids concentration in layer j (g m^{-3})
- K' is the Vesilind model parameter (m^3 g^{-1})
- K_{st} is the settling velocity TSS switch (g m^{-3})

The parameters used in the Takács function and BioWin model are listed in Table 4.4. The table lists the parameters, giving a description of the parameters, the associated symbol and the parameter units. Also given in the table are the model parameter values to be used in any benchmark work. It can be seen that although the models have a similar form, the magnitude of the parameters is significantly different.

Table 4.4: *'Simulation benchmark'* settler model parameters and their associated values.

Parameter Description	Parameter Symbol	Value	Units
Takács Function			
Maximum settling velocity	v'_o	250	m day^{-1}
Maximum Vesilind settling velocity	v_o	474	m day^{-1}
Hindered zone settling parameter	r_h	0.000576	m^3 $(g\ SS)^{-1}$
Flocculant zone settling parameter	r_p	0.00286	m^3 $(g\ SS)^{-1}$
Non-settleable fraction	f_{ns}	0.00228	dimensionless
BioWin Model			
Maximum settling velocity	v_o	970	m day^{-1}
Vesilind model parameter	K	0.75	m^3 kg^{-1}
Settling velocity TSS switch	K_{ts}	900	g m^{-3}

The BioWin settling model parameters were determined by plotting the Takács and BioWin functions for values of X_j ranging from 0 to 8000 mg L^{-1} (Figure 4.1). Then, the BioWin parameters were adjusted until the BioWin curve approximated the benchmark required curve. Figure 4.1 shows the results of this procedure. It is clear that the resulting curves are not identical, but they are sufficiently close to one another. That is, the observed differences do not result in significant differences in the modelled settling behaviour.

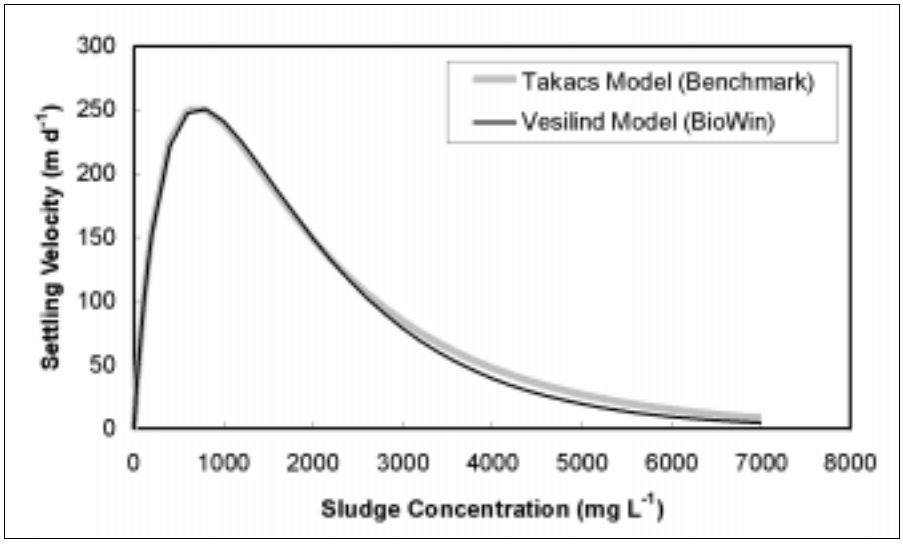

Figure 4.1: Examination of settling velocity profiles using the Takács and BioWin settling models.

4.2 CONFIGURATION ISSUES - BIOWIN

Set-up of the *'simulation benchmark'* configuration using the BioWin user interface requires 13 flowsheet elements including 5 *bioreactors*, a *model settler*, 2 *flow split nodes*, 2 *mixer nodes*, and 3 *input/output elements* as well as the various connecting *closed pipes*. The benchmark configuration generated using BioWin (version 4.4b) can be seen in Figure 4.2.

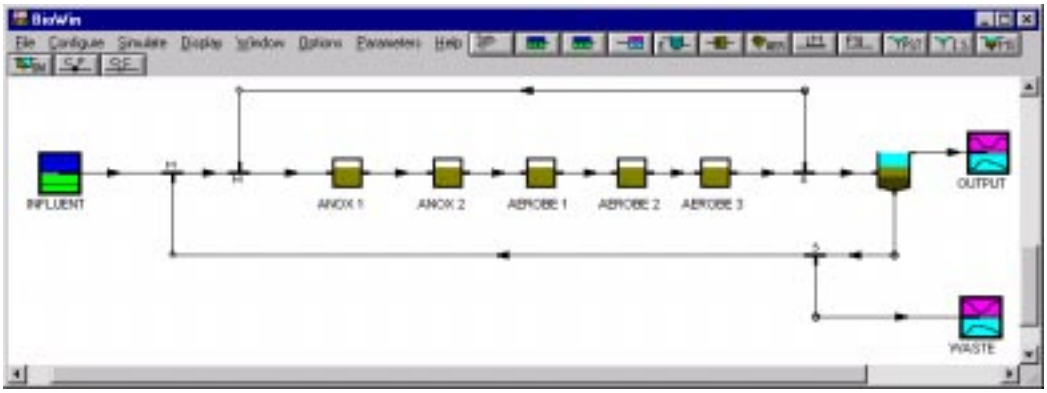

Figure 4.2: Interface layout of COST *'simulation benchmark'* plant in BioWin (Version 4.4b).

The appropriate volumes for each unit can be input using a 4m depth for the bioreactors, and as specified in the benchmark description a 4m depth for the settler. The settler should be assigned 10 layers with the settler influent fed to *layer 4* (Figure 4.3). This feed layer is inconsistent with the benchmark description that specifies layer 6 (from the bottom), but two points need to be emphasised here: (i) BioWin numbers its settler layers from the top rather than the bottom as the *'simulation benchmark'* does, and (ii) the dynamic settling results of the BioWin settler (Figure 4.4) were found to more closely approximate the confirmed dynamic benchmark results when layer 4 (as opposed to layer 5) was used. Differences in the settler models may provide some indication as to the cause.

The Takács model makes use of a term defined as the 'non-settleable fraction'. This term is used to calculate the minimum solids attainable after settling, but it also removes a portion of the solids from the settling equation (Equation 1.1). These 'non-settleable' solids are mapped directly to the effluent stream irrespective of the settling process that is occurring. This has the effect of increasing the effluent solids as compared to the case when such a term is not used and the BioWin model has no such term. Therefore, it is understandable that the BioWin model produces a lower effluent suspended solids concentration when the feed is introduced to the BioWin settler at layer 5 as specified in the benchmark description. This explanation also is consistent with the settler output data shown in Figure 4.4.

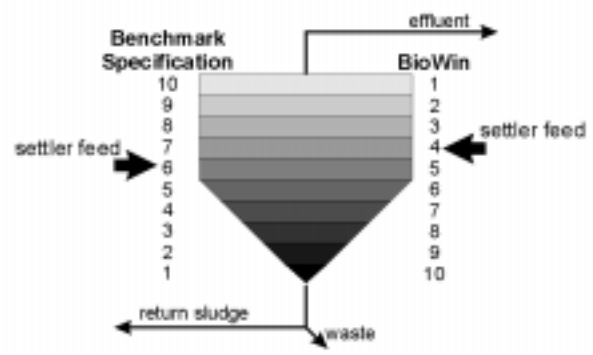

Figure 4.3: Schematic diagram of differences in the benchmark and BioWin settler set-up.

When the BioWin settler is fed using layer 5, the settler generates a dynamic profile that mirrors the expected benchmark profile, but the values are consistently 4mg L^{-1} lower (Figure 4.4). Unfortunately, there is no easy way to overcome this difference while still feeding to layer 5. Alternatively, the settler can be fed at layer 4. At steady state, this results in a settler profile that is consistent with the benchmark specified results, but dynamically it causes the settler to be more susceptible to variations in the settler input. This also is depicted in Figure 4.4 as greater oscillations in effluent solids can be seen when the settler is fed at layer 4. Nevertheless, for benchmarking purposes, the BioWin settler should be fed using layer 4, as on average, it produces results that more closely approximate the accepted benchmark results.

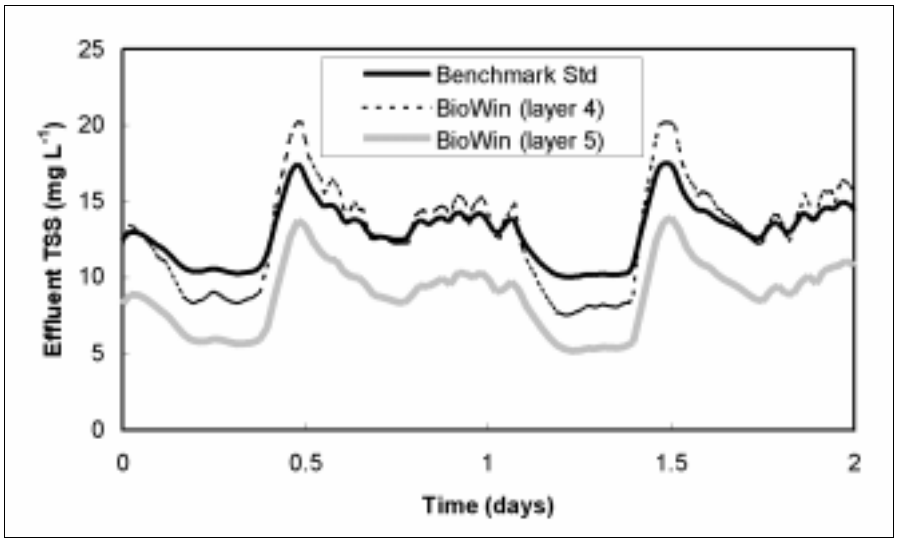

Figure 4.4: Illustrative example showing the differences in BioWin output (effluent solids) when the feed to the settler is changed from layer 4 to layer 5.

4.3 SIMULATION ISSUES - BIOWIN

At the simulation stage, several other factors must be addressed including configuration of the dynamic influent files and setting up proper aeration.

4.3.1 Influent Data Files

To set-up the dynamic influent files ('*.din' files in the BioWin environment), it is possible to use a BioWin utility program 'dinassit.exe' or it can be done manually using a spreadsheet program. Either way, it is important that users recognise that the benchmark influent files are not in the correct format for BioWin. That is, BioWin requires that the influent data be ordered in a particular way to be read correctly and this order is not the same as presented in the raw data files. Further, BioWin influent files have 24 components, so place holders (i.e. 0's) must be used for influent components not included in the data files, but are used by BioWin (i.e. poly-P heterotrophs) nevertheless. Figure 4.5 shows the structure of the influent files required by BioWin.

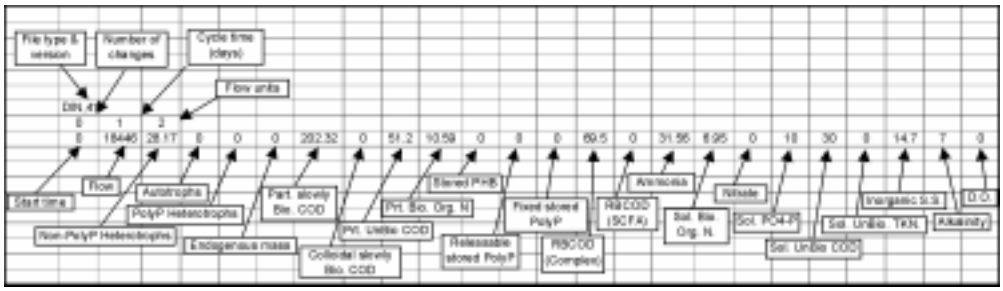

Figure 4.5: Illustrative example of the ordered BioWin influent file structure showing the influent composition for the steady state simulations and place holders for unused components.

From the previous discussion related to the settler model, it should be apparent that the total suspended solids (TSS) concentration is a critical component of the secondary settling process and needs to be calculated. By definition, TSS is the sum of the volatile and inert suspended solids (VSS + ISS), but it should be noted that ISS is not included in the benchmark defined influent files because ASM1 does not make use of an ISS state variable in its model structure. To overcome this apparent inconsistency, the '*simulation benchmark*' defines a bioreactor TSS:CODp ratio of 0.75 gTSS $gCOD^{-1}$ (where CODp is the bioreactor particulate COD concentration). This allows the TSS concentration to be calculated explicitly from the particulate COD concentration, which is calculable from the ASM1 state variables.

BioWin is not designed to allow this TSS calculation from CODp. Rather, BioWin specifies ISS as a state variable and calculates the TSS concentration from the addition of the ISS and VSS by tracking the dynamic flux of ISS into and out of the system. Therefore, influent ISS is a crucial influent component that must be included in a BioWin dynamic influent file. For the steady state case, the influent ISS was determined from calculations based on the accepted steady state '*simulation benchmark*' results (i.e. given the system SRT, volume and steady state influent flow

rate, what influent ISS would be required to give a bioreactor TSS:CODp ratio of 0.75 under steady state conditions). Figure 4.5 shows that for the steady state simulations, users should input an ISS of 14.7 g m^{-3}. Dynamically a constant influent ISS is less 'realistic'. So, for the dynamic influent files, a variable ISS was calculated based on a constant influent VSS:TSS ratio of 92.83%. This percentage was calculated based on the steady state influent composition outlined above using an influent ISS of 14.7 g m^{-3}. The influent ISS at any given influent time interval is calculated as follows.

$$VSS_{in} = \frac{(X_{B,H_{in}} + X_{S_{in}} + X_{I_{in}})}{1.48} \tag{4.3}$$

$$TSS_{in} = \frac{VSS_{in}}{0.9283} \tag{4.4}$$

$$ISS_{in} = TSS_{in} - VSS_{in} = \frac{VSS_{in} \bullet (1 - 0.9283)}{0.9283} \tag{4.5}$$

NOTE: It may be argued that a CODp:VSS ratio of 1.48 for the influent is too low, but it has been used here to avoid the introduction of another parameter. It also should be realised that this parameter has no effect on the calculated ISS as changing this ratio has a reciprocal and offsetting effect on the influent VSS:TSS ratio used in the calculation.

In addition to the ISS, one other state variable is of particular interest to the BioWin benchmark implementation: alkalinity. A constant influent alkalinity of 7 mmol L^{-1} is specified in the benchmark description and as the BioWin biological model makes use of this variable in an alkalinity switching function, it must be included in the influent data file.

4.3.2 Dissolved Oxygen Modelling

There are two BioWin issues that must be addressed and understood with respect to the benchmark- specified aeration. The first is dissolved oxygen (DO) saturation and the second is air supply. BioWin users are referred to the simulator documentation for a complete description of the principles involved, but some of the more important points and the necessary parameter values are outlined here.

The '*simulation benchmark*' defines the bioreactor DO saturation as 8 g m^{-3}, but this value cannot be explicitly entered into the BioWin 'DO saturation conc.' text box, because BioWin adjusts this entered value to account for temperature, pressure and tank depth and calculates the 'true' DO saturation in the bioreactor. This 'true' bioreactor DO saturation is calculated as follows:

$$C_{\infty}^{*} = C_{S}^{*} \bullet \left(\frac{P_B - P_{VT} + d_E \bullet 101.325/10.34}{P_S - P_{VT}} \right) \tag{4.6}$$

where: C^*_S tabulated value for dissolved oxygen surface saturation concentration at water temperature T, standard atmospheric pressure P_S, and 100 percent relative humidity, g m^{-3}

C^*_∞ steady-state dissolved oxygen saturation concentration attained at infinite time at water temperature T and field atmospheric pressure P_B, g m^{-3} [Note: this is "in the tank", is influenced by tank depth and is equal to 8 g m^{-3} for the '*simulation benchmark*']

d_E effective saturation depth (m)

P_B field atmospheric pressure (kPa)

P_S atmospheric pressure at standard conditions (101.325 kPa or 10.34 m water)

P_{VT} vapor pressure of water at temperature T (kPa)

and,

$$P_B = P_S = 101.325\ kPa \quad \rightarrow 0m \text{ elevation assumed} \qquad (4.7)$$

$$P_{VT} = 0.66304619 \bullet (1.06400888)^T = 2.293\ kPa \quad \rightarrow T = 20^oC \text{ assumed} \qquad (4.8)$$

$$d_E = f_{ED} \bullet tank\ depth \quad \rightarrow f_{ED} = 0.325 \text{ assumed} \qquad (4.9)$$

Substituting into Equation 4.6, it is possible to determine that C^*_S is 7.0882 mg L^{-1}. This is the value that should be entered into the BioWin 'DO saturation conc.' text box to be consistent with the benchmark defined DO saturation of 8 g m^{-3}. Figure 4.6 shows the applicable aeration dialogue box (BioWin version 4.4b) for entering the DO saturation value.

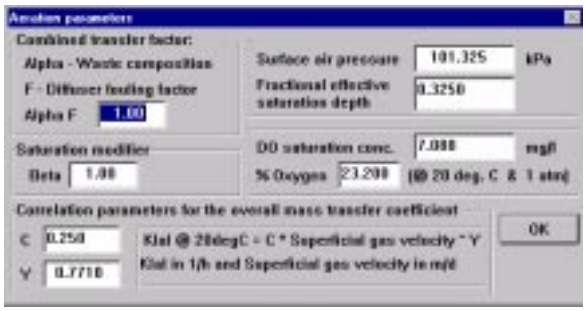

Figure 4.6: 'Aeration parameters' dialogue box (BioWin version 4.4b) showing the necessary BioWin aeration parameters for the *'simulation benchmark'*.

The method for calculating the air supply rate is the second issue that needs addressing to ensure that the BioWin aeration is consistent with the '*simulation benchmark*' defined K_La values of 10 and 3.5 hr^{-1}. In BioWin, K_La is calculated (in units of hr^{-1}) as follows:

$$K_La = C \bullet U_{SG}^{\ Y} \qquad (4.10)$$

where: C = 0.250 (day [m • hr]$^{-1}$)

Y = 0.771

U_{SG} = superficial gas velocity (m^3 [m^2 • day]$^{-1}$)
= Q_{air} / bioreactor area

Given that the aerobic *'simulation benchmark'* bioreactors have a volume of 1333 m^3 and a depth of 4 m, a bioreactor area of 333.25 m^2 can be calculated for each tank. Equation 4.10 can be solved for Q_{air} in units of m^3 d^{-1} and then converted to units of m^3 s^{-1}. The final aeration parameters to enter are α and *Beta* that should both be set to 1 to achieve the required benchmark K_La values (Figure 4.6). Figure 4.7 shows the applicable dialogue box (version 4.4b) for entering the air supply rate and Table 4.5 lists the required air flow rates to achieve the desired *'simulation benchmark'* defined K_La values.

Figure 4.7: Bioreactor dialogue box (BioWin version 4.4b) for the AEROBE 1 reactor - see Figure 4.2 – showing the required air supply rate to attain a K_La of 10 hr^{-1} in that bioreactor.

Table 4.5: BioWin air flow rates needed to achieve the desired *'simulation benchmark'* K_La values.

Tank #	Benchmark required K_La (hr^{-1})	BioWin Q_{air} (m^3 s^{-1})
3 & 4	10	0.4614825
5	3.5	0.1182507

4.3.3 Simulation Output Verification

The final step in the *'simulation benchmark'* implementation procedure is the verification of the steady state and dynamic simulation output. To achieve the correct output, users will need to make two further adjustments. The first relates to the waste and settler underflow flow rates and the second relates to a 'bug' in the steady state solver (note that this 'bug' appears only in older versions of BioWin and subsequently has been rectified in all versions of BioWin32).

In a deviation from the defined benchmark flows, users should adjust the settler underflow flow rate to 18832 m^3 d^{-1} and the waste flow rate to 386 m^3 d^{-1}. The change results in a 1 m^3 d^{-1} increase in waste flow rate, but maintains the recycle flow rate at the defined rate of 1 times the average Q_{in} (18446 m^3 d^{-1}). These changes are necessary to achieve the proper sludge age, and as a result, the proper output data. The difference in the BioWin settler model is the suspected cause and although attempts have been made here to achieve similar behaviour, clearly the model output is not precisely the same, as indicated by Figure 4.1 and Figure 4.4. The exact reason for this

minor change in waste flow rate is unknown, but the change is necessary to achieve the benchmark defined results.

The second simulation issue to address is a numerical 'bug' caused by a numerical 'shortcut' used to solve the BioWin settler (version 4.4 and before) under steady state conditions. The effect of this 'bug' is a small discontinuity in the soluble components when a dynamic simulation immediately follows from a steady state solution. The 'bug' does not appear to effect the steady state results as these are correct, but an observed discontinuity appears in the first dynamic time step following the steady state solution. The discontinuity is avoidable using the following procedure to set up the settler state variables correctly.

To overcome the dynamic discontinuity in the soluble components:
i) Determine the steady state solution using the steady state solver
ii) Set-up a constant influent '*.din' file (using the flow-weighted *dry weather* data, Table 2.7) and dynamically simulate the steady state solution, which should take approximately three sludge ages (25 days). [Most of the 'bug' effect is removed within a couple of simulation days, but to return to the exact steady state solver solution a simulation time closer to three sludge ages will be required. That is, during this 'dynamic' steady state simulation, the output will jump away from the steady state solution in the first time step, then slowly return to the values generated with the steady state solver.]

These steps will remove the effect of the 'bug' such that the dynamic weather file simulations can be performed from this 'dynamic' steady state without any discontinuity appearing in the output file. It should be made clear though, that this 'bug' appears only in BioWin versions prior to BioWin32 and has been fixed in all versions of BioWin32. However, benchmark users should follow this procedure to correctly set-up older versions of BioWin for the dynamic simulations.

4.4 BASIC CONTROL STRATEGY - BIOWIN

No attempt has been made to implement the 'basic control strategy' into BioWin.

4.5 CONCLUSION

Tuning BioWin to the *'simulation benchmark'* specifications is a relatively simple task once some of the specific BioWin features are understood. The most challenging part of the tuning process relates to understanding the BioWin settler model and its relationship to the benchmark specified settling model because these two models are not the same nor do they have the same numerical structure. It is not the purpose of this work to determine which model is more appropriate, but it is essential that they behave similarly if benchmark consistent results are to be attained using BioWin. The settler model impact can be overcome through several deviations from the benchmark description including the use of different model parameters, and some minor configuration changes. In addition to the model differences, *'simulation benchmark'* users need to be aware of several BioWin-specific features related to aeration, and setting up the proper structure

for dynamic influent files. That is, to achieve benchmark consistent results, users must be aware of how to overcome the differences between BioWin and the benchmark description to tune their simulator to the benchmark specifications. This chapter has outlined the BioWin-specific procedures that should ensure tuning to the benchmark specifications and thus result in the benchmark defined results.

4.6 ACKNOWLEDGEMENTS

The information contained in this chapter was compiled from various sources including personal experience with the software, the BioWin User Manual, and BioWin Technical Notes. Further, understanding some of the specific BioWin features would have been very difficult without input from EnviroSim Associates Ltd. principals, Peter L. Dold and Mark Fairlamb, hence their contribution should be fully acknowledged.

5

EFOR™

described by René Dupont

NOTE: The issues and procedures outlined in this chapter are specific for the use of EFOR with the *'simulation benchmark'* and in no way should be interpreted as necessary procedures for using EFOR for any other purpose.

EFOR™ is a software tool dedicated to the modelling of wastewater treatment systems, which makes it easy to construct a wide variety of wastewater treatment plants. Even treatment plants with a complex and dynamic operation are easily described and simulated with EFOR.

This chapter describes specific features of EFOR and how these EFOR features should be used to tune it to the prescribed '*simulation benchmark*' specifications. In order to do this, EFOR Version 3.1 or later should be used.

5.1 CONFIGURATION ISSUES - EFOR

Before beginning, the user must ensure that the *Advanced user interface* option is active. On the VIEW menu, click *Options* and activate the *Advanced user interface*. By doing this, a range of rarely used advanced features in EFOR will be available to the user.

Creating the '*simulation benchmark*' plant from scratch in EFOR is relatively straightforward (With Version 3.1 and later, one of the sample plants delivered with the program, is the '*simulation benchmark*' plant.). The initial step is to open a new workspace by selecting *New* from the FILE menu. ASM-1 should be selected when prompted for the model type. Select *Design*

EFOR™ is a trademarked product of:
EFOR ApS, c/o Krüger A/S, Gladsaxevej 363, 2860 Søborg, Denmark.
tel: +45 39 69 02 22
fax: +45 39 69 08 06
e-mail: efor@efor.dk
web: www.efor.dk

from the PLANT menu and 'draw' the plant with the necessary units. Figure 5.1 shows an illustrative example of what the finished 'drawing' should look like. After completing the 'drawing', each unit must be dimensioned according to the '*simulation benchmark*' and the needed meters and pumps must be installed. Figure 5.2 shows the an example of the dialogue box used to enter the physical characteristics of the biological reactors.

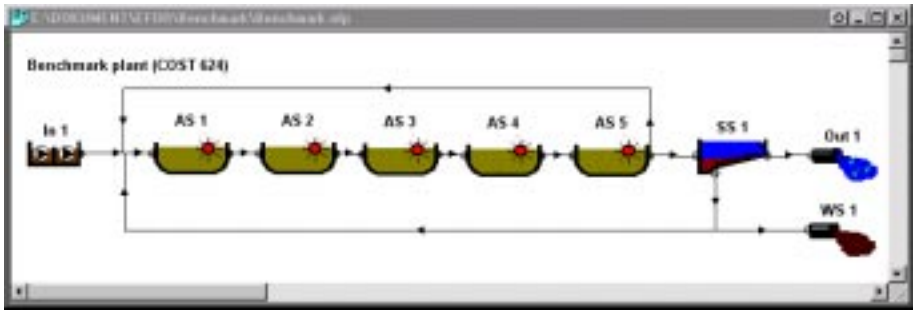

Figure 5.1: The '*simulation benchmark*' plant representation in EFOR 3.1.

Aeration in tanks 1 and 2 should be switched off using the 'No aeration' radio button. For tanks 3, 4 and 5 aeration should be switched on by selecting K_La aeration with a *maximum* capacity of 240 day^{-1}. The actual aeration is defined by the control loops (see later). Note that in EFOR, K_La is specified in units of day^{-1} as opposed to the hr^{-1} units defined in the '*simulation benchmark*'. Further, although indicated in Figure 5.2 the *Oxygen meter* is only needed in tank 5.

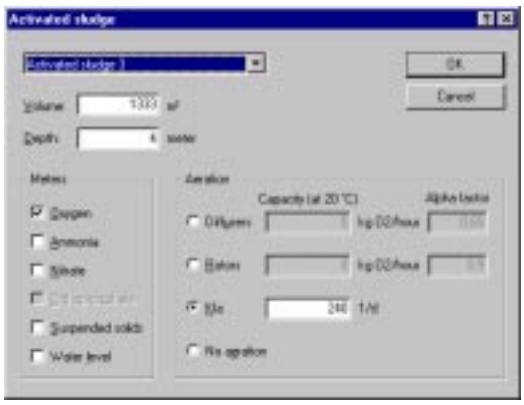

Figure 5.2: 'Activated sludge' dialogue box for tank 5 showing the required input values for volume, depth and K_La.

For the settler unit set the dimensions according to the '*simulation benchmark*' specifications as shown in Figure 5.3. Select *Flux settler* as settler type and ensure that the *Processes* check box is unchecked. From the MODEL menu, select *Advanced flux model parameters* and set the settling

velocity model to *Double exponential*. Set the layer height to 0.4 metres and set the inlet height to be fixed at layer 6. Figure 5.3 shows the applicable dialogue box.

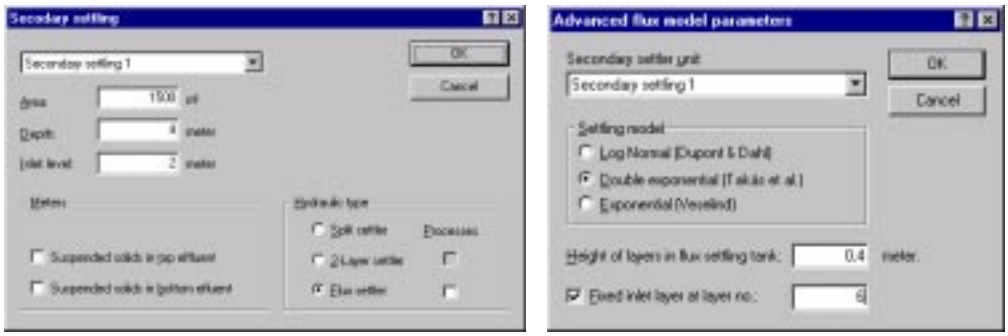

Figure 5.3: 'Secondary settling' and 'Advanced flux model parameters' dialogue boxes used to input the settler characteristics.

The final stage of the configuration set-up is constructing the necessary control loops. Table 5.1 lists the specifications for the required '*simulation benchmark*' control loops.

Table 5.1: Control loop set-up for the steady state and openloop cases.

Name	Meter	Controller	Controlled item	Setpoint	Units
Return sludge	Time	Timer	Pump on return sludge pipe	758.68	$m^3\ hr^{-1}$
Waste sludge	Time	Timer	Pump on waste sludge pipe	16.04	$m^3\ hr^{-1}$
Recirculation	Time	Timer	Pump on recirculation pipe	2305.75	$m^3\ hr^{-1}$
Aeration AS3	Time	Timer	Aeration (K_La) in AS3	240	day^{-1}
Aeration AS4	Time	Timer	Aeration (K_La) in AS4	240	day^{-1}
Aeration AS5	Time	Timer	Aeration (K_La) in AS5	84	day^{-1}

5.2 MODEL ISSUES – EFOR

The '*simulation benchmark*' specifies two models, one for the biological processes and one for the secondary settler. In EFOR, the user must choose the correct models from the various predefined models that come with the software. The default secondary settler is set-up in a slightly more advanced way than defined in the '*simulation benchmark*', but by changing a few options and setting some constants the model can easily be turned into a model which does not differ significantly from the one specified in the '*simulation benchmark*'.

5.2.1 Biological Process Model

The biological model, ASM-1, in EFOR is identical to the IAWQ's Activated Sludge Model No. 1 and should be chosen as the biological process model. Table 5.2 lists the stoichiometric parameters to be used and Table 5.3 lists the kinetic parameters.

Table 5.2: Stoichiometric ASM1 model parameters for the '*simulation benchmark*' in EFOR. The column with the EFOR benchmark constants at T=20°C should be used.

Name	Symbol	EFOR default	EFOR Benchmark	Benchmark Value	Units
Heterotrophic yield	Y_H	0.67	0.67	0.67	g COD g COD^{-1}
Autotrophic yield	Y_A	0.24	0.24	0.24	g COD g N^{-1}
Frac. of particulate products	f_p	0.08	0.08	0.08	dimensionless
Frac. of N in biomass	i_{xb}	0.086	0.08	0.08	g N g COD^{-1}
Frac. of N in products	i_{xp}	0.06	0.06	0.06	g N g COD^{-1}

The main issue to be dealt with in regards to the EFOR model parameters is temperature dependency. The '*simulation benchmark*' description indicates that the defined parameters are meant to reflect values observed at 15°C. However, if the user inputs a defined temperature of 15°C, an alternative parameter set must be used. This is because EFOR demands that parameters be input at a reference temperature of 20°C and automatically adjusts all temperature dependent parameters to the defined simulation temperature. That is, if the user defines a temperature of 15°C, the parameter set defined under EFOR Input (Table 5.3) should be used. The alternative approach is to use the '*simulation benchmark*' defined constants directly and specify a simulation temperature of 20°C. By using a simulation temperature of 20°C, no temperature correction will occur and users are assured that the correct parameters are used.

Table 5.3: Kinetic ASM1 model parameters for the '*simulation benchmark*' in EFOR. The column with the EFOR benchmark constants at T=20°C should be used.

Name	Symbol	EFOR default (T=20°C)	EFOR Input (T=20°C)	EFOR Calculated (T=15°C defined)	Units
Heterotrophic growth	μ_H	6.0	5.64796	4.0	day^{-1}
Heterotrophic decay	b_H		0.42359	0.3	day^{-1}
Anoxic correction factor	η_g	0.6	0.8	0.8	dimensionless
Half saturation for substrate	K_S	20.0	10.0	10.0	g COD m^{-3}
Half saturation for Oxygen	K_{OH}	0.2	0.2	0.2	g O_2 m^{-3}
Half saturation for nitrate	K_{NO}	0.25	0.5	0.5	g N m^{-3}
Autotrophic growth	μ_A	0.9	0.81615	0.5	day^{-1}
Autotrophic decay	b_A	0.15	0.0816	0.05	day^{-1}
Half saturation for ammonia	K_{NH}	0.2	1.0	1.0	g N m^{-3}
Half saturation for oxygen	K_{OA}	0.3	0.4	0.4	g O_2 m^{-3}
Ammonification rate	K_a	0.08	0.07059	0.05	m^3 (g COD $day)^{-1}$
Hydrolysis rate	K_h	3.0	4.23597	3.0	day^{-1}
Anoxic correction factor	η_h	0.4	0.8	0.8	dimensionless
Half saturation for hydrolysis	K_Ξ	0.1	0.1	0.1	g COD g COD^{-1}

In order to deal with total suspended solids, the '*simulation benchmark*' defines TSS as a constant factor of the particulate COD (i.e. TSS = 0.75 [X_S + X_P + X_I + X_{BH} + X_{BA}]). In EFOR, this is achieved by setting the conversion constant for activated sludge (COD-SS) to 0.75.

5.2.2 Settling Model

The default set-up of the flux-settling model in EFOR differs from the '*simulation benchmark*' in several ways. The settling model in EFOR has by default a layer height of 0.1 metres, a dynamic inlet level, short-circuiting, and a log-normal settling function with a temperature dependent settling velocity (Dupont & Dahl, 1995), but this model can easily be turned into the model specified by the '*simulation benchmark*'.

By setting the layer height to 0.4 metres and setting the inlet height to be fixed at level 6 (Figure 5.3), the default model is transformed into a 10-layer flux model using the Takács double exponential settling velocity model with fixed inlet height.

On the MODEL menu, click *Secondary settler* and select the *Double exponential* settling model tab. Set the constants according to those listed in Table 5.4. Setting the Ω_1 and Ω_2 constants to zero turns off the short-circuiting. Setting the t_{V0} constant to zero turns off temperature dependency for the settling rate. The applicable dialogue box is shown in Figure 5.4.

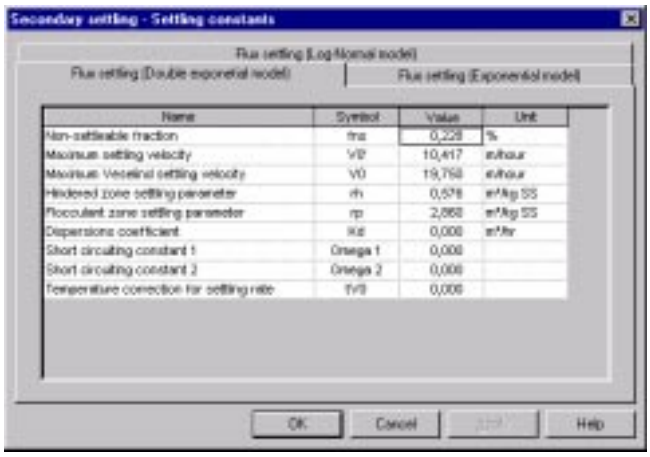

Figure 5.4: EFOR 'Secondary settling' dialogue box for entering the secondary settling parameters.

An EFOR-specific difference in the calculation of the flux between layers results in more sludge storage in EFOR as compared to other platforms. For the steady state case, this means that the lower layers of the settler contain more sludge in EFOR than found in the accepted '*simulation benchmark*' results. To illustrate this, Table 5.5 shows the EFOR results next to the accepted '*simulation benchmark*' results. For the dynamic simulations, this effect will be observed as a slight delay in the response of particulate components from the secondary settler due to the higher retention time that the EFOR implementation results in.

Table 5.4: '*Simulation benchmark'* constants for the secondary settler set in EFOR.

Description	Symbol	Value	Units
Non settle able fraction	*Fns*	0,228	%
Maximum settling velocity	V_0'	10,417	m hr^{-1}
Maximum Veselind settling velocity	V_0	19,75	m hr^{-1}
Hindered zone settling parameter	r_h	0,576	m^3 kg SS^{-1}
Flocculent zone settling parameter	r_p	2,86	m^3 kg SS^{-1}
Dispersion	K_d	0,01	m^2 hr^{-1}
Short circuiting factor 1	Ω_1	0.0	dimensionless
Short circuiting factor 2	Ω_2	0.0	dimensionless
Temperature correction	t_{V0}	0.0	dimensionless

Table 5.5: Steady state results for TSS in secondary settler (g COD m^{-3}).

Layer	Accepted '*simulation benchmark'* Results	EFOR Results
10	12.5	12.5
9	18.2	18.1
8	29.5	29.6
7	69.0	69.0
6	356	356
5	356	356
4	356	356
3	356	949
2	356	3277
1	6394	6397

5.2.3 Dissolved Oxygen Modelling

The '*simulation benchmark*' specifications define a maximum oxygen saturation concentration of 8 g O_2 m^{-3}. If the user has defined a simulation temperature of 15°C then this can be achieved in EFOR by setting the temperature constant, t_{OX1}, to 12.617812. As described previously (Section 5.2.1), it is also possible to define a simulation temperature of 20°C in which case, the temperature constant, t_{OX1}, should be set to 13.630192. The EFOR default value is 14.652.

5.3 SIMULATION ISSUES - EFOR

The most important simulation issue relates to the set-up of the influent files. The influent files defined in the '*simulation benchmark'* are segregated into ASM1 state variables components, but EFOR requires influent components be defined in terms of composite analytical terms (i.e. total COD, total TKN…). This means that the '*simulation benchmark*' influent data must be converted to analytical data and some conversion parameters must be estimated. This is, in principle, not a difficult task to perform with a spreadsheet program, provided the fractional composition of the influent does not change.

For the steady state case, conversion parameters can be calculated so that there is an exact match in the influent composition between the EFOR influent and the influent defined in the '*simulation benchmark*'. Table 5.6 lists the analytical data to be used in EFOR, which corresponds to the '*simulation benchmark*' influent data, and Table 5.7 lists the corresponding conversion constants required for the steady state case.

Table 5.6: Analytical influent data for the steady state case.

Parameter	Symbol	EFOR	Units
Inlet flow	Q	18446	$m^3\ day^{-1}$
Total COD	COD_T	381.19	g COD m^{-3}
Soluble COD	COD_S	99.5	g COD m^{-3}
Total Kjeldahl-N	KJN_T	51.52	g N m^{-3}
Soluble Kjeldahl-N	KJN_S	38.51	g N m^{-3}
Ammonium	S_{NH}	31.56	g N m^{-3}
Nitrate	S_{NO}	0.0	g N m^{-3}
Alkalinity	S_{ALK}	7.0	mol L^{-1}
Oxygen	S_O	0.0	g O_2 m^{-3}

Table 5.7: EFOR conversion constants for the steady state influent.

	Symbol	EFOR default	Benchmark	Units
Conc. of autotrophs	K_{Xba}	0.1	0.0	g COD m^{-3}
Conc. of particulate products	K_{Xp}	0.0	0.0	g COD m^{-3}
Frac of S_S in particulate COD	f_{SS}	0.1	0.0	g COD g COD^{-1}
Frac.of X_{BH} in particulate COD	f_{Xbh}	0.15	0,1	g COD g COD^{-1}
Frac. of S_I in soluble COD	f_{Si}	0.25	0.3015	g COD g COD^{-1}
Frac. of X_I in particulate COD	f_{Xi}	0.05	0.1635	g COD g COD^{-1}
Conversion from COD to SS	$f_{COD/SS}$		0.75	g SS g COD^{-1}

For the dynamic case it is not possible to achieve an exact match because the fractional composition of the '*simulation benchmark*' influent is not constant. For example, some of the influent components are fixed (i.e. S_i), but because the total COD is varying, the fraction of S_i to total COD also varies. In EFOR, this fraction is a constant, which results in a different EFOR influent when compared to the '*simulation benchmark*' defined influent data. Use of the steady state constants for the dynamic case results in influent files that have the same mean load and dynamic pattern, but individual components do not vary in exactly the same way. That is, some of the components show less variation and others show more variation than specified in the '*simulation benchmark*' influent files. Figure 5.5 shows the situation for readily biodegradable soluble substrate for the *dry weather* file. Similar deviations between EFOR and the '*simulation benchmark*' data are found for the other COD components. This problem only concerns COD components. Nevertheless, with EFOR Version 3.1, the best approximations of the '*simulation benchmark*' influent files are supplied as EFOR formatted ready-to-use files.

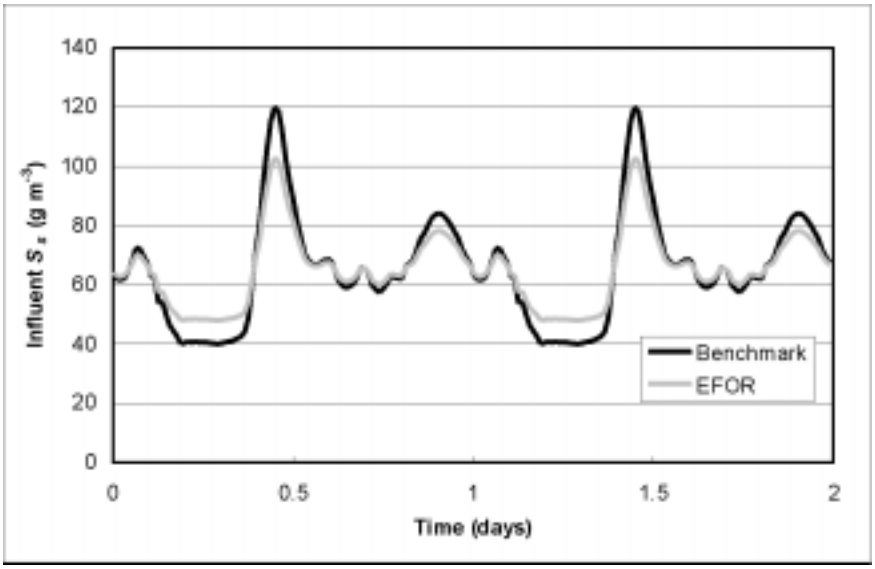

Figure 5.5: Effect of composite analytical terms approach in EFOR on the influent readily biodegradable substrate as compared to the defined *'simulation benchmark'* concentration.

5.4 BASIC CONTROL STRATEGY - EFOR

It is reasonably simple to implement the control strategy in EFOR. To implement the control strategy, the control loops already defined for the openloop case simply need to be changed according to Table 5.8. However, it should be noted that it is not possible to add noise to the nitrate measurement in EFOR.

Table 5.8: Control loop set-up for the closed loop case.

Name	Meter	Controller	Controlled item	Setpoint	Units
Return sludge	Time	Timer	Pump on return sludge pipe	758.68	m^3 hr^{-1}
Waste sludge	Time	Timer	Pump on waste sludge pipe	16.04	m^3 hr^{-1}
Recirculation	NO_3 (tank 1)	PID	Pump on recirculation pipe	1.0	gN m^{-3}
	Delay = 600 sec	K= T_I^{-1}= T_D=0	Max capacity = 3843 m^3 hr^{-1}		
Aeration AS3	Time	Timer	Aeration (K_La) in tank 3	240	day^{-1}
Aeration AS4	Time	Timer	Aeration (K_La) in tank 4	240	day^{-1}
Aeration AS5	O_2 (tank 5)	PID	Aeration (K_La) in tank 5	2.0	g O_2 m^{-3}
		K= T_I^{-1}= T_D=0	Max capacity = 240 day^{-1}		

5.5 CONCLUSION

The '*simulation benchmark*' can be implemented in EFOR, but users must be aware of some specific features and their effect on the simulation output. Specifically, the user needs a deeper understanding of the settler integration routine and the structure of EFOR influent files. However, once the user has some experience with EFOR and a general understanding of the '*simulation benchmark*', it is easy to set-up a treatment plant in EFOR that yields '*simulation benchmark*' consistent results. It is simply unfortunate that it is not possible to add noise to the nitrate signal that is involved in the basic control strategy. Users also need to realise that one issue remains unresolved. Because EFOR uses composite analytical terms for its influent and constant fractionation of the COD, dynamic differences with EFOR are unavoidable.

5.6 ACKNOWLEDGEMENTS

The information contained in this chapter was compiled from various sources including personal experience with the software, the EFOR 3.0 User's Guide and the EFOR Technical Reference.

6

FORTRAN

described by M.N. Pons, J.F. Beteau & J.M. LeLann

FORTRAN has been recognised for years as a powerful programming language for scientific applications. For this reason, it was chosen for the *'simulation benchmark'* implementation. On one hand, FORTRAN permits a rigorous translation of the *'simulation benchmark'* mathematical description into a set of code lines, which can be used to simulate the plant operation without any further plant assumptions. On the other hand, user-friendliness is restricted, especially with respect to data management.

The advantage of using FORTRAN for the implementation relates to the freedom it provides with respect to the introduction of generalisation. For example, the user may want to modify the total number of compartments or the number of unaerated compartments or the number of layers in the clarifier and this can be done very easily by over-parameterisation or dynamic allocation. The disadvantages relate to typing (and programming) mistakes that can occur especially for the biological section. Further, an integration algorithm is needed to solve the set of ordinary differential equations, and this necessitates writing the algorithm code or obtaining the source code from an external source and then testing the algorithm.

A very basic version of the *'simulation benchmark'* can be written in FORTRAN 90 to run in a DOS window. However a Digital Visual Fortran version, which includes a user-friendly interface and graphical result output, might be preferred in the long term. Nevertheless, irrespective of the user's goal, FORTRAN provides a reasonable platform in which to implement the *'simulation benchmark'*.

6.1 MODEL ISSUES - FORTRAN

For modelling the state variables, the verbal description of the benchmark has to be transformed into a set of ordinary differential equations (Equation 6.1) that can be solved with an integration routine.

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{v}(t)) \qquad (6.1)$$

where:

x = is the state vector,
u = the control vector (if applicable),
v = the perturbation vector (influent characteristics and flowrate)

6.1.1 Biological Process Model

FORTRAN programming of the bioreactor is very similar to that proposed for MATLAB/Simulink (Chapter 8) and will not be described in detail here. However, of particular interest is the dissolved oxygen (DO) balance in the bioreactor. Care should be taken so that the DO concentration never exceeds the defined saturation concentration (i.e. 8 g m^{-3}). This should, of course, never happen, but it may be necessary to include a check in the integration routine because clearly the integration routine does not care about the physical meaning of the results.

6.1.2 Settling Model

Several things need to be considered with respect to the settler. For instance, users should be aware of the fact that to write the mass balances, the solid fluxes between layers and the general upward and downward flows must be considered. This leads to the following set of equations for the particulate components, assuming that the solid flux due to gravity sedimentation is given by:

$$J_s = v_s(X)X \tag{6.2}$$

where:

X = the total sludge concentration

And, the double-exponential settling velocity function is written as:

$$v_s(X) = \max\left[0, \min\left\{v_0', v_0\left(e^{-r_h(X-X_{\min})} - e^{-r_P(X-X_{\min})}\right)\right\}\right] \tag{6.3}$$

where:

X_{min} = $f_{ns} X_f$ (X_f is the total solid concentration in the clarifier feed)

For the bottom layer (m = 1):

$$\frac{dX_1}{dt} = \frac{v_{dn}(X_2 - X_1) - \min(J_{s,2}, J_{s,1})}{z_1} \tag{6.4}$$

where: $$v_{dn} = \frac{Q_u}{A} = \frac{Q_r + Q_w}{A}$$

For the intermediate layers below the feed layer (m = 2 to m = 5):

$$\frac{dX_m}{dt} = \frac{v_{dn}(X_{m+1} - X_m) + \min(J_{s,m}, J_{s,m+1}) - \min(J_{s,m}, J_{s,m-1})}{z_m} \quad (6.5)$$

For the feed layer (m = 6):

$$\frac{dX_m}{dt} = \frac{\frac{Q_f X_f}{A} + J_{clar,m+1} - (v_{up} + v_{dn})X_m - \min(J_{s,m}, J_{s,m-1})}{z_m} \quad (6.6)$$

where: $v_{up} = \frac{Q_e}{A}$

For the intermediate clarification layers above the feed layer (m = 7 to m = 9):

$$\frac{dX_m}{dt} = \frac{v_{up}(X_{m-1} - X_m) + J_{clar,m+1} - J_{clar,m}}{z_m} \quad (6.7)$$

where: $J_{clar,j} = \begin{cases} \min(v_{s,j}X_j, v_{s,j-1}X_{j-1}) \\ or \\ v_{s,j}X_j \ if \ X_{j-1} \le X_t \end{cases}$

And finally, for the top layer (m = 10):

$$\frac{dX_{10}}{dt} = \frac{v_{up}(X_9 - X_{10}) - J_{clar,10}}{z_{10}} \quad (6.8)$$

where: $J_{clar,10} = \begin{cases} \min(v_{s,10}X_{10}, v_{s,9}X_9) \\ or \\ v_{s,10}X_{10} \ if \ X_9 \le X_t \end{cases}$

For the soluble components, noted here with the symbol *Z*, the equations are:

For the bottom layers (m = 1 to 5):

$$\frac{dZ_m}{dt} = \frac{v_{dn}(Z_{m+1} - Z_m)}{z_m} \quad (6.9)$$

For the feed layer (m = 6):

$$\frac{dZ_m}{dt} = \frac{\frac{Q_f Z_f}{A} - (v_{dn} + v_{up})Z_m}{z_m} \quad (6.10)$$

And, for the top layers (m = 6 to 10):

$$\frac{dZ_m}{dt} = \frac{v_{up}(Z_{m-1} - Z_m)}{z_m} \tag{6.11}$$

In the code example shown in Section 6.5.1, two subroutines have been used to describe the settler behaviour, one for the mass balance (subroutine settler) and another one for the solid fluxes (subroutine fluxsol).

6.2 SIMULATION ISSUES - FORTRAN

A general flowsheet of a FORTRAN-based implementation is shown in Figure 6.1 and contains four main sections: initialisation, calculation of the time-derivatives, integration routine and output management.

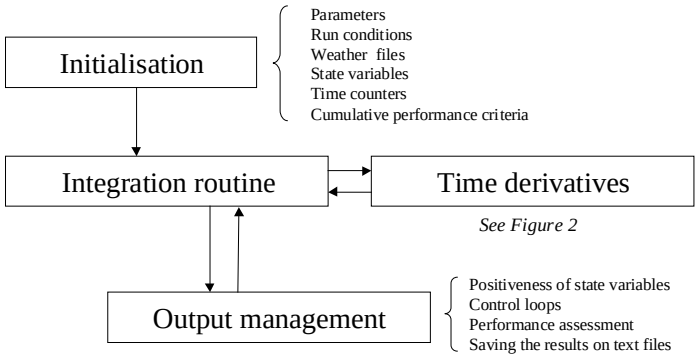

Figure 6.1: General flowsheet of FORTRAN implemented *'simulation benchmark'* program.

6.2.1 Initialisation

Many different initialisations need to be performed including cumulative variables, time counters and state variables. The initial state variables can either be read in from a file or default values can be used. The first step in the initialisation section would classically consist of reading all the model parameters, run conditions, and the like. However, due to the weather files, and the large amount of process related data, it may be preferable to organise the data in several files according to the type of information described. That is, plant data (total number of reactors, number of non aerated reactors, volumes of the aerated zone and of the non aerated zone, clarifier characteristics, etc), biomodel parameters, settling parameters, control loops (sensors characteristics, controller parameters), weather file names, event time steps, choice of integration routine can each be placed in separate modifiable text files. Structuring the files in this way is preferred to a full initialisation because it will add flexibility and should avoid re-compilation of the source code at every modification of the parameters (Section 6.5.2).

Considering the number of variables and the number of constants, it is recommended that a declaration file be used to minimise the errors between the subroutines. All the declarations (types of variables, variables in named COMMONs) can be stored in a file (variab.def), and included at the beginning of each subroutine. This provides an easy means to minimise information transfer errors between the subroutines. For example:

```
SUBROUTINE XXX
IMPLICIT NONE
INCLUDE 'variab.def'
```

There are different ways to manage the influent files, but one example is shown in Section 6.5.3. In this example, the files are managed in two sections, one for the dynamic stabilisation phase and another for the performance assessment. It should be noticed that all the weather files start at time 0. Because of this, some time rearrangement has to be done. It is suggested that this time rearrangement be done once, at the beginning of the program and should make use of a large matrix to store the influent characteristics. As shown in Section 6.5.3, a large vector, xyz (mmaxdat,10) is used to store the influent data. The constant inputs such as nitrate and oxygen (set to 0) are considered directly in the subroutine defining the influent characteristics at each integration time step.

6.2.2 Integration and Time Derivatives

Whatever the integration routine, the time derivatives of the state variables must be calculated at time t. This part is the most tedious to program and the most prone to (typing) errors. As shown in Figure 6.2, the time derivative section has been divided into several subroutines in this implementation.

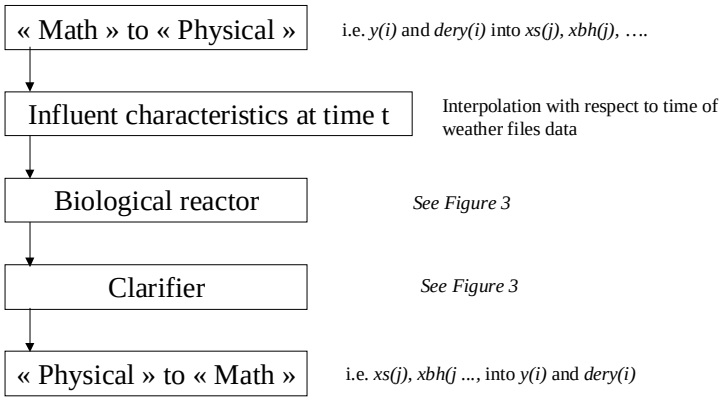

Figure 6.2: Flowsheet for the time-derivative calculation.

The most important subroutines (the bioreactor and the secondary settler) are coupled together by recycle flows and the flowsheets for the time derivatives are given in Figure 6.3.

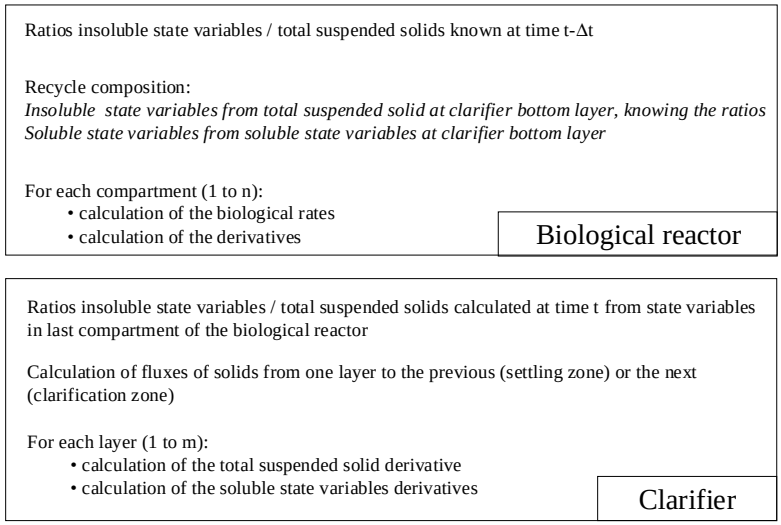

Figure 6.3: Flowsheet for the bioreactor and secondary settler time derivatives.

Due to the number of state variables (145) it is particularly dangerous to use generic names such as y(i) and dery(i), although they are generally required by the integration routines. Therefore it is necessary to call equivalence subroutines to transform the "numeric" variables into "physic" ones before the calculation of the derivatives and to back-transform the "physic" variables into the "numeric" ones after the calculation. An example of such a subroutine is shown in Section 6.5.4. Calling such a routine will increase the required memory and decrease the calculation speed, but it is much more convenient for the programmer. When such a framework is used, the names of the "physic" variables can be chosen to match those given in the benchmark description.

The choice of the integration routine requires some care. The biological system described by the '*simulation benchmark*' is reputed to give stiff differential equations that require special integration algorithms. A Gear algorithm was initially selected and the code was extracted from the Harwell library. However, as the problem contains a high number of state variables and as the differential equations are not easy to manipulate, a numerical calculation of the Jacobian matrix was opted for. Convergence was obtained with this procedure, but the steady state values were not correct. Another integration routine (DASSL) was tested but the calculation time was unrealistic. Finally, a simple Runge-Kutta 4^{th} order algorithm with fixed time-step was tested with success. The correct steady state values were obtained in a reasonable calculation time. The initial time step was 0.001 hr, which was later doubled without loss of performance (i.e. the same steady state values were obtained). It is recommended that this Runge-Kutta method be used by FORTRAN users.

6.2.3 Output

In the implementation used here, the output section is triggered by the integration routine. The output routine works using time thresholds that trigger various events such as control actions, sludge age calculations, performance evaluation and storage of data in result files. Following verification that all state variables are positive, the time (current_time) is checked against n_event thresholds (next_event_time(i), i= 1 to n_event) and when appropriate, an action is taken. For example:

```
for i = 1 to n_event
if(current_time ≥ next_event_time(i)) then
                do the action
                increment next_event_time(i) by event_time_step(i)
        end if
end do
```

Dealing with the various discrete actions in this way may not be the best, but it permits clear programming of the output subroutine, event by event, and new events are easy to add.

6.3 BASIC CONTROL STRATEGY - FORTRAN

Several different implementations can be used for the PI controllers. The one chosen here is the discrete type where Δt is the time interval between two actions of a controller, $y(k)$ is the measurement at time $k\Delta t$, and y^{set} is the setpoint . For such a controller, the control action to be applied, $u(k)$, is calculated as follows:

$$u(k) = Du + u(k-1) \tag{6.12}$$

where:
$$Du = K \left\{ [e(k) - e(k-1)] + \frac{\Delta t}{T_i} e(k) \right\}$$

K = proportionality constant
T_i = integral time
$e(k)$ = error at time, $k\Delta t$
$e(k-1)$ = error at time, $(k-1)\Delta t$

with the following constraints:

Constraint	Explanation
$\|Du\| \le Du_{max}$	limit on u variation between two successive actions
$u_{min} \le u(k) \le u_{max}$	permissible values of u

The implementation for the dissolved oxygen control loop is relatively easy, as it is a simple PI controller with an ideal sensor. However, care should be taken when implementing the nitrate control loop to store the correct nitrate concentration and take into account the signal delay

(Section 6.5.5). It should be noted that the FORTRAN closedloop results shown in Appendix 1.3 were obtained with strong constraints on Du_{max} to simulate the 'real' case where wear and tear on pumps and motors must be considered.

6.4 CONCLUSION

This chapter has attempted to describe some of the issues that will be faced by FORTRAN programmers that attempt to implement the simulation benchmark. Although many of the issues have been discussed only briefly and just a few code examples have been included, it is hoped that this chapter will help programmers with their implementation of the *'simulation benchmark'*.

6.5 FORTRAN - CODE EXAMPLES

6.5.1 FORTRAN – Example 1

```
c Model of the secondary settler
      subroutine settler(t)
      implicit none
      include 'variab.def'
      INTEGER i
      DOUBLE PRECISION xf,qf,vdn,vup
      DOUBLE PRECISION fluxs(MNDEC)
      DOUBLE PRECISION t
c Definition of the incoming concentrations
c Total concentration of solids
      xf = 0.75 * (xs(n) + xp(n) + xi(n) + xbh(n) + xba(n))
c Ratios
      rxs=xs(n)/xf
      rxp=xp(n)/xf
      rxi=xi(n)/xf
      rxbh=xbh(n)/xf
      rxba=xba(n)/xf
      rxnd=xnd(n)/xf
c Exit flow
      qe=q0-qw
c Incoming flow
      qf = qe + qr + qw
c Velocities
c downward
      vdn = (qr + qw)/A
c upward
      vup = qe/A
c Solid fluxes
      call fluxsol(xf,fluxs)
c Mass balances on total solids
      DO i = 1,long
c Below the feed layer
            if (i.LT.kfeed) then
                  dx(i) = (vdn*(x(i+1)-x(i))+fluxs(i+1)-fluxs(i))/z(i)
            end if
c Feed layer
            if(i.EQ.kfeed) then
                  dx(i) = (qf*xf/A - (vup+vdn)*x(i)-fluxs(i)+fluxs(i+1))/z(i)
            end if
c Clarification layers below top layer
            if(i.GT.kfeed) then
                  dx(i) = (vup*(x(i-1)-x(i)) + fluxs(i+1) - fluxs(i))/z(i)
            end if
c Top layer
            if(i.EQ.long) then
                  dx(i) = (vup*(x(i-1)-x(i)) - fluxs(i))/z(i)
            endif
      ENDDO
c Mass balances on the soluble components
```

```
            DO i=1,long
c Feed layer
                        if (i.EQ.kfeed) then
                                dsi2(i) = (qf*si(n)/A -(vdn+vup)*si2(i))/z(i)
                                dss2(i) = (qf*ss(n)/A -(vdn+vup)*ss2(i))/z(i)
                                dsno2(i) = (qf*sno(n)/A -(vdn+vup)*sno2(i))/z(i)
                                dsnh2(i) = (qf*snh(n)/A -(vdn+vup)*snh2(i))/z(i)
                                dsnd2(i) = (qf*snd(n)/A -(vdn+vup)*snd2(i))/z(i)
                                dsalk2(i) = (qf*salk(n)/A -(vdn+vup)*salk2(i))/z(i)
                                dso2(i) = (qf*so(n)/A -(vdn+vup)*so2(i))/z(i)
                        endif
c Below the feed layer
                        if (i.LT.kfeed) then
                                dsi2(i) = vdn*(si2(i+1)-si2(i))/z(i)
                                dss2(i) = vdn*(ss2(i+1)-ss2(i))/z(i)
                                dsno2(i) = vdn*(sno2(i+1)-sno2(i))/z(i)
                                dsnh2(i) = vdn*(snh2(i+1)-snh2(i))/z(i)
                                dsnd2(i) = vdn*(snd2(i+1)-snd2(i))/z(i)
                                dsalk2(i) = vdn*(salk2(i+1)-salk2(i))/z(i)
                                dso2(i) = vdn*(so2(i+1)-so2(i))/z(i)
                        endif
c Above the feed layer
                        if (i.GT.kfeed) then
                                dsi2(i) = vup*(si2(i-1)-si2(i))/z(i)
                                dss2(i) = vup*(ss2(i-1)-ss2(i))/z(i)
                                dsno2(i) = vup*(sno2(i-1)-sno2(i))/z(i)
                                dsnh2(i) = vup*(snh2(i-1)-snh2(i))/z(i)
                                dsnd2(i) = vup*(snd2(i-1)-snd2(i))/z(i)
                                dsalk2(i) = vup*(salk2(i-1)-salk2(i))/z(i)
                                dso2(i) = vup*(so2(i-1)-so2(i))/z(i)
                        endif
            ENDDO
            return
            end

c Solid flux due to gravity sedimentation in secondary settler
      SUBROUTINE fluxsol(xf,fluxs)
      IMPLICIT none
      INCLUDE 'variab.def'
      INTEGER i
      DOUBLE PRECISION xf,fluxs(MNDEC)
      DOUBLE PRECISION vitesse(MNDEC), xmin
            DO  i=1,long
c i = 1 --> bottom layer
c i = long --> top layer
c*************************************************************************
c           Double-exponential settling velocity function
c           (Takacs et al., 1991)
c*************************************************************************
                        xmin=fns * xf
                        vitesse(i) = v0 * (dexp(-rh*(x(i)-xmin))
     &                  -dexp(-rp*(x(i)-xmin)))
                                if(vitesse(i).GT.vprim0) vitesse(i)=vprim0
                        if(vitesse(i).LT.0) vitesse(i) = 0.
            ENDDO
c Fluxes
            fluxs(1) = 0.
            DO i=2,long
```

```
c Feed layer and below
                    if(i.LE.kfeed) then
                              fluxs(i)=vitesse(i)*x(i)
                              if((vitesse(i)*x(i)).GT.(vitesse(i-1)*x(i-1)))
   &                          fluxs(i)=(vitesse(i-1)*x(i-1))
                    END IF
c Clarification layers
                    if(i.GT.kfeed) then
                              if(x(i-1).LE.xthres) fluxs(i)=vitesse(i)*x(i)
                              if(x(i-1).GT.xthres) then
                                        fluxs(i)=vitesse(i)*x(i)
                                        if((vitesse(i)*x(i)).GT.(vitesse(i-1)*x(i-1)))
   &                                    fluxs(i)=(vitesse(i-1)*x(i-1))
                              END IF
                     END IF
          ENDDO
    END
```

6.5.2 FORTRAN – Example 2

```
c Reading the kinetics parameters
          subroutine kinet

c Declaration of variables
          IMPLICIT none
          INCLUDE 'variab.def'
          CHARACTER*30 titre
c Opening data file
          open(10,file="data/cinetica."//num)
c Reading the parameters
          read(10,'(A)') titre
c Aerobic growth of heterotrophs: muh, ks, koh
          read(10,'(A)') coments
          read(10,*) ks
          read(10,'(A)') coments
          read(10,*) muh
          read(10,'(A)') coments
          read(10,*) koh
          close(11)
end
```

With the first lines of the corresponding data file being:

```
Kinetic_parameters
# ks
10.
# muh
0.167
# koh
0.2
```

6.5.3 FORTRAN – Example 3

```
c General conditions of simulation
          subroutine init
```

```
c Declaration of variables
            IMPLICIT none
            INCLUDE 'variab.def'
            INTEGER i,j,k,l,bid
            DOUBLE PRECISION w(10)
            CHARACTER*30 finput,fbruit
c Stabilization values
            DOUBLE PRECISION sis,sss,xis,xss,xbhs,xbas,xps,sos,snos,
     &      snhs,snds,xnds,salks
            DOUBLE PRECISION qs
c Data file name
            open(30,file="data/debut."//num)
c Stabilization values and period
            read(30,'(A)') coments
            read(30,*) sis,sss,xis,xss,xbhs,xbas,xps,sos,snos,snhs,snds,xnds,
     &      salks,qs
            read(30,'(A)') coments
c Stabilization period under constant conditions
            read(30,*) tstab
c Stabilization period under dry weather conditions
            tstabc=14.
c Total stabilization period
            tstabc=tstabc+tstab
c Building xyz ….
            l=1
            xyz(l,1)=0.
            xyz(l,2)=sss
            xyz(l,3)=xbhs
            xyz(l,4)=xss
            xyz(l,5)=xis
            xyz(l,6)=snhs
            xyz(l,7)=sis
            xyz(l,8)=snds
            xyz(l,9)=xnds
            xyz(l,10)=qs
            l=2
            xyz(l,1)=tstab
            xyz(l,2)=sss
            xyz(l,3)=xbhs
            xyz(l,4)=xss
            xyz(l,5)=xis
            xyz(l,6)=snhs
            xyz(l,7)=sis
            xyz(l,8)=snds
            xyz(l,9)=xnds
            xyz(l,10)=qs
c Stabilization data files, to be run 1 times = 14 days
            read(30,'(A)') coments
             read(30,'(A)') finput
            open(29,file=finput)
            read(29,'(A)') coments
130         l=l+1
            if(l.GT.mmaxdat) go to 135
            read(29,*,end =135) (w(j),j=1,10)
c Storage in influent data table
            xyz(l,1) = w(1)+tstab
            do i=2,9
                        xyz(l,i)=w(i)
```

```
            enddo
            xyz(l,10)=w(10)
            go to 130
135         l=l-1
            close(29)
c Influent data file for test of control strategy
            read(30,'(A)') coments
            read(30,'(A)') finput
            open(29,file=finput)
            read(29,'(A)') coments
230         l=l+1
            if(l.GT.mmaxdat) go to 235
            read(29,*,end =235) (w(j),j=1,10)
c Storage in influent data table
            xyz(l,1) = w(1)+tstabc
            do i=2,9
                  xyz(l,i)=w(i)
            enddo
            xyz(l,10)=w(10)
            go to 230
235         l=l-1
            close(29)
c Final size of xyz
            maxdat=l
end
```

An interpolation routine is necessary to recalculate the characteristics at times when they are not given, but needed by the integration routine:

```
c Influent
            SUBROUTINE entree(t)
            IMPLICIT none
            INCLUDE 'variab.def'
            INTEGER i,j,k,index,kod
            DOUBLE PRECISION t
            DOUBLE PRECISION valeur
            k=0
            index=0
200         k=k+1
            if(k.GT.maxdat-1) then
            index=maxdat-1
            go to 250
            end if
            if(t.GE.xyz(k,1).AND.t.LT.xyz(k+1,1)) index=k
            if(index.EQ.0) go to 200
250         continue
            ss0=interpol(t,index,2)
            xbh0=interpol(t,index,3)
            xba0=0.
            xs0=interpol(t,index,4)
            xi0=interpol(t,index,5)
            si0=interpol(t,index,7)
            snd0=interpol(t,index,8)
            sno0=0.
            so0=0.
            xp0=0.
            snh0=interpol(t,index,6)
            xnd0=interpol(t,index,9)
```

```
            q0= interpol(t,index,10)
            salk0=7.
            xp0=0.
            RETURN
            END

c Interpolation for the influent data file
            FUNCTION interpol(t,index,kod)
            IMPLICIT NONE
            INCLUDE 'variab.def'
            DOUBLE PRECISION t
            INTEGER index,kod
            DOUBLE PRECISION deltat
            deltat=xyz(index+1,1)-xyz(index,1)
            interpol=(xyz(index+1,kod)-xyz(index,kod))/deltat
     &      *(t-xyz(index,1))+xyz(index,kod)
            RETURN
            END
```

6.5.4 FORTRAN – Example 4

```
c Transformation of the physic variables into math variables
            subroutine  numer(t)
            IMPLICIT none
            INCLUDE 'variab.def'
            INTEGER i,j
            DOUBLE PRECISION t
c Transformation into math variables
            j=0
c Aeration tank
            DO i=1,n
c Soluble inert organic matter (1)
                        j=j+1
                        dery(j)=dsi(i)
                        y(j)=si(i)
c Readily biodegradable substrate (2)
                        j=j+1
                        dery(j)=dss(i)
                        y(j)=ss(i)
            ENDDO
            RETURN
            END

c Transformation of the numeric variables into physic variables
            subroutine  physic(t)
            IMPLICIT none
            INCLUDE 'variab.def'
            INTEGER i,j
            DOUBLE PRECISION t
c Transformation into physical variables
            j=0
c Aeration tank
            DO i=1,n
c Soluble inert organic matter (1)
                        j=j+1
                        dsi(i)=dery(j)
                        si(i)=y(j)
```

```
c Readily biodegradable substrate (2)
                    j=j+1
                    dss(i)=dery(j)
                    ss(i)=y(j)
          ENDDO
          RETURN
          END
```

6.5.5 FORTRAN – Example 5

```
c Control of nitrate level in last anoxic compartment (loop nb 2)
          ic=2
c If time to activate the controller has be reached ….
          if(t.GT.tno3) then
          no3m=sno(nanox)*(1.+noise(t))
          if(no3m.LT.0.1) no3m=0.1
          no3new=no3m
c Previous value of control variable: u_relu
          u_relu(ic)=qa
c Nitrate measurement to consider due to delay (pasno3)
          ymes(ic)=no3old
c Setpoint to consider
          c(ic)=no3set
c Error to consider
          error=no3set-no3old
          call pid(ymes,c,u_relu,ucalc,ic)
c New value of the control variable
          qa=ucalc(ic)
```

7

GPS-X™

described by John B. Copp

NOTE: The issues and procedures outlined in this chapter are specific for the use of GPS-X with the *'simulation benchmark'* and in no way should be interpreted as necessary procedures for using GPS-X for any other purpose.

GPS-X is a modular, multipurpose modelling environment for the simulation of wastewater treatment systems. To tune GPS-X to the benchmark specifications users should be aware of several GPS-X-specific features and options. The implementation of the *'simulation benchmark'* into GPS-X is straightforward, but to achieve benchmark consistent results, users must be aware of how to overcome the differences between the simulator and the specifically defined *'simulation benchmark'* description.

In particular, this chapter outlines the changes that should be made to some of the GPS-X default settings to achieve benchmark consistent results. Further, parameter values for several important *'simulation benchmark'* variables are explained and derived specifically for GPS-X. A 'fix' for a model error is presented, a potential influent data read-in problem is discussed and an unresolvable deviation in the settler models is pointed out. Finally, a GPS-X-specific alternative to the *'simulation benchmark'* defined simulation procedure is proposed.

7.1 CONFIGURATION ISSUES – GPS-X

Set-up of the *'simulation benchmark'* configuration using the GPS-X user interface can be done in several ways. Figure 7.1 shows one alternative using five *CSTR* objects, a *3-way combiner* object,

GPS-X™ is a trademarked product of:
Hydromantis Inc., 1685 Main St. West, Suite 302, Hamilton, Ontario, CANADA L8S 1G5
tel: +1 (905) 522-0012
fax: +1 (905) 522-0031
web: www.hydromantis.com

a *circular secondary clarifier* object, two *discharge* objects and an *influent* object as well as the various *connections*. It should be made clear that this is only one option. For instance, users may choose to use a *plugflow tank* object rather than the *CSTR* objects. Nevertheless, the results of the simulations should be independent of the layout used provided the modelling options are entered correctly.

Table 7.1: GPS-X process models to be used with the *'simulation benchmark'*.

Process	GPS-X model
Biological	iawprc
Settling	noreac1d

Using the 'cnlib' macro library, users next will need to assign models to the process objects. Table 7.1 shows the GPS-X process models to be used (note that when assigning the process models, it is recommended that the GPS-X 'sourcing' option be used so as to minimise the potential for parameter input errors). Once the configuration is 'drawn' and the models have been assigned, users should input the physical and operational data for each tank as specified in the benchmark description.

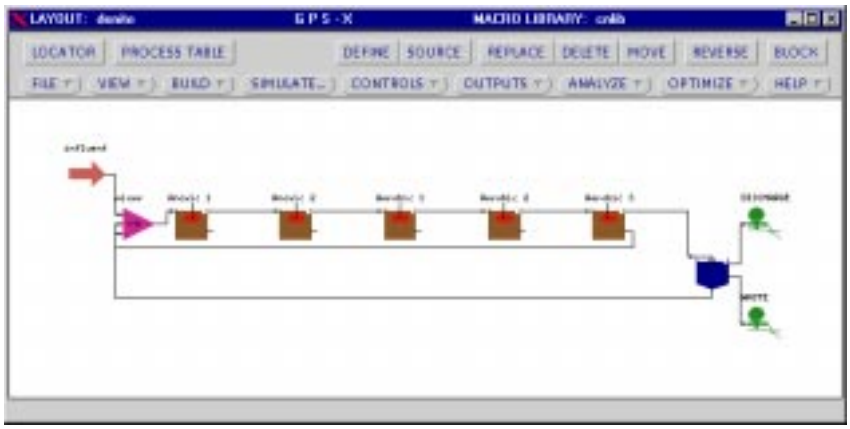

Figure 7.1: Interface layout of the COST *'simulation benchmark'* plant in GPS-X (Version 2.3.1).

7.2 MODEL ISSUES – GPS-X

The *'simulation benchmark'* specifies two process models: the IAWQ Activated Sludge Model #1 - ASM1 (Henze *et al.*, 1987) for the biological processes and the double-exponential settling function of Takács *et al.* (1991) for the settling process. The GPS-X models specified in Table 7.1 are consistent with those required by the *'simulation benchmark'*, but users should be aware of the subtle differences in these models as compared to those specified in the benchmark description.

7.2.1 Biological Process Model

Table 7.2 lists the state variables and symbols used in ASM1 and the GPS-X 'iawprc' model. The variables and symbols are essentially identical, but users should note that GPS-X uses the symbol X_U for the 'particulate products arising from biomass decay' whereas ASM1 uses X_P.

Table 7.2: Comparison of state variable symbols used in ASM#1 and in the GPS-X 'iawprc' model.

State Variable Description	ASM1 Symbol	GPS-X Symbol	Units
Soluble inert organic matter	S_I	S_I	g COD m^{-3}
Readily biodegradable substrate	S_S	S_S	g COD m^{-3}
Particulate inert organic matter	X_I	X_I	g COD m^{-3}
Slowly biodegradable substrate	X_S	X_S	g COD m^{-3}
Active heterotrophic biomass	$X_{B,H}$	X_{BH}	g COD m^{-3}
Active autotrophic biomass	$X_{B,A}$	X_{BA}	g COD m^{-3}
Particulate products arising from biomass decay	X_P	X_U	g COD m^{-3}
Oxygen	S_O	S_O	g COD m^{-3}
Nitrate and nitrite nitrogen	S_{NO}	S_{NO}	g N m^{-3}
$NH_4^+ + NH_3$ nitrogen	S_{NH}	S_{NH}	g N m^{-3}
Soluble biodegradable organic nitrogen	S_{ND}	S_{ND}	g N m^{-3}
Particulate biodegradable organic nitrogen	X_{ND}	X_{ND}	g N m^{-3}

Table 7.3 lists the stoichiometric parameters to be used in the 'iawprc' model. These are listed in the order that they appear in the GPS-X dialogue box and include two ratios not defined in the '*simulation benchmark*': 'VSS/TSS ratio' and 'BOD5 to BOD ultimate ratio'. The required VSS/TSS ratio is calculated from the benchmark specified TSS/CODp ratio of 0.75 and the benchmark specified CODp/VSS ratio of 1.48. By combining these specified ratios, it is possible to calculate the required VSS/TSS ratio (i.e. VSS/TSS = 1/(0.75 x 1.48)). The BOD ratio can be left at its default value of 0.66.

Table 7.3: *'Simulation benchmark'* stoichiometric parameters for ASM1 and the GPS-X 'iawprc' model (n/s not specified).

GPS-X Description	ASM1 Symbol	Benchmark Value	GPS-X Value	Units (using ASM#1 nomenclature where necessary)
Fractions				
particulate COD to VSS ratio	f_{cv}	1.48	1.48	g COD g VSS^{-1}
VSS/TSS ratio	-	n/s	0.9009	g VSS g TSS^{-1}
BOD_5 to $BOD_{ultimate}$ ratio	-	n/s	0.66	g BOD_5 g BOD_∞^{-1}
Heterotrophs				
Yield	Y_H	0.67	0.67	g X_{BH} COD formed (g COD utilised)$^{-1}$
N content of active mass	i_{XB}	0.08	0.08	g N (g COD)$^{-1}$ in biomass (X_{BA} & X_{BH})
N content of endogenous mass	i_{XP}	0.06	0.06	g N (g COD)$^{-1}$ in X_p
Endog. Residue	f_P	0.08	0.08	dimensionless
Autotrophs				
Yield	Y_A	0.24	0.24	g X_{BA} COD formed (g N utilised)$^{-1}$

Table 7.4 lists the kinetic parameters to be used with the 'iawprc' model in the order that they appear in the GPS-X dialogue box. These values are identical to those specified in the *'simulation benchmark'*.

Table 7.4: *'Simulation benchmark'* kinetic parameter values for the GPS-X 'iawprc' model.

GPS-X Description	ASM1 Symbol	GPS-X Value	Units (using ASM1 nomenclature where necessary)
Heterotrophs			
Maximum specific growth rate	μ_H	4.0	day^{-1}
Half saturation coefficient	K_S	10.0	g COD m^{-3}
Organism decay rate	b_H	0.3	day^{-1}
Anoxic hydrolysis factor	η_h	0.8	dimensionless
Anoxic growth factor	η_g	0.8	dimensionless
Maximum spec. hydrolysis rate	k_h	3.0	g X_S (g X_{BH} COD·day)$^{-1}$
Hydrolysis half saturation	K_X	0.1	g X_S (g X_{BH} COD)$^{-1}$
Ammonification rate	k_a	0.05	$m^3 \cdot$ (gCOD day)$^{-1}$
Autotrophs			
Maximum specific growth rate	μ_A	0.5	day^{-1}
Half saturation coefficient	K_{NH}	1.0	g NH_3-N m^{-3}
Organism decay rate	b_A	0.05	day^{-1}
Switching Functions			
Heterotrophic O2 half.sat.	$K_{O,H}$	0.2	g O_2 m^{-3}
Autotrophic O2 half.sat.	$K_{O,A}$	0.4	g O_2 m^{-3}
Nitrate half sat.	K_{NO}	0.5	g NO_3-N m^{-3}

The GPS-X 'iawprc' model is ASM1 and is consistent with the required '*simulation benchmark*' model, but the 'iawprc.asp' file may contain a model error that needs correcting (note: this file error is known to exist in GPS-X versions up to 2.4. Users of later versions should check for this error and correct it if necessary). The error appears in the oxygen rate equation (rso&o =…) and has the effect of eliminating the influence of K_{NH} on this calculated rate. Although this is a minor problem it needs correcting if the benchmark results are to be duplicated. Equation 7.1 shows the incorrect equation and Equation 7.2 shows the corrected equation. Users should either delete the incorrect equation and replace it with the corrected equation or 'comment out' the incorrect equation using a '!' character at the start of the line and add the corrected equation.

INCORRECT
```
rso&o = mrso&o*bso&o/(koh&o+bso&o)
```
(7.1)

CORRECTED
```
rso&o = -(1.0-yh&o)/yh&o*mr1&o*bso&o/(koh&o+bso&o)&
        -(4.57-ya&o)/ya&o*mr3&o*bso&o/(koa&o+bso&o)
```
(7.2)

To make this correction users will need to edit the process rates. The reader is referred to the GPS-X User's Guide for more information on this procedure, but Figure 7.2 shows an illustrative example of the editable process rate dialogue box.

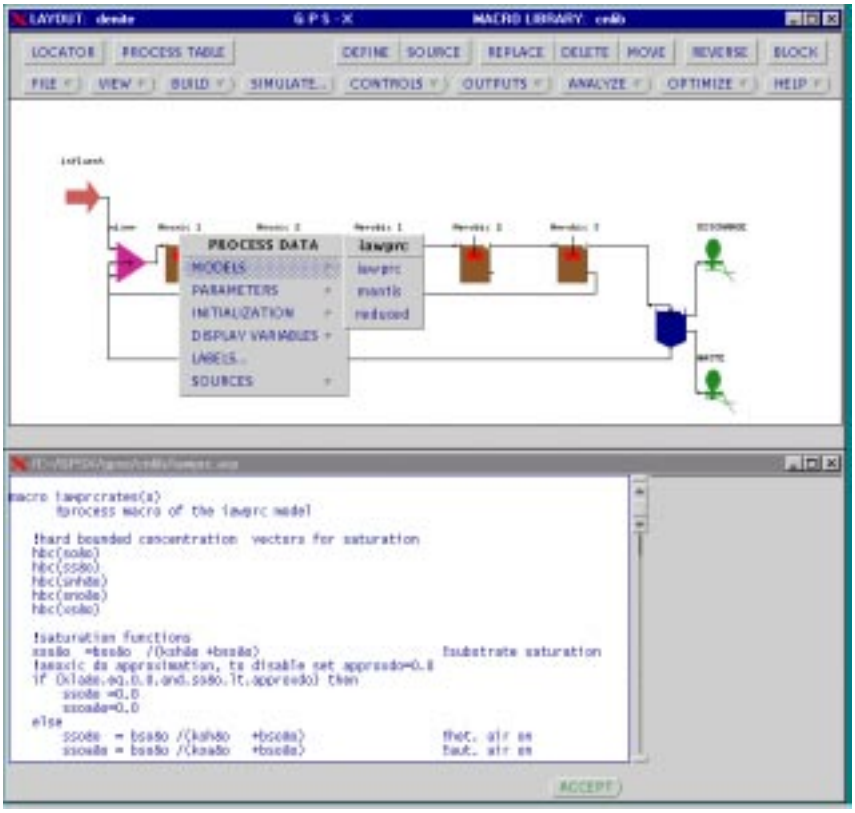

Figure 7.2: Illustrative example showing the process rate equation dialogue box that will need to be accessed to examine the process rate equations.

7.2.2 Settling Model

The double-exponential settling velocity function of Takács *et al.* (1991) was chosen as the '*simulation benchmark*' settling model due to its wide use and apparent acceptability as a fair representation of the settling process. The parameters in the Takács function and the associated values to be used are listed in Table 7.5. The table lists the parameters, giving a description of the parameters, the associated symbols and the parameter units. Also given in the table are the model parameter values to be used in any benchmark work.

Table 7.5: *'Simulation benchmark'* settler model parameters and their associated values.

Parameter Description	Parameter Symbol	Value	Units
Takács Function			
Maximum settling velocity	v'_o	250	m day^{-1}
Maximum Vesilind settling velocity	v_o	474	m day^{-1}
Hindered zone settling parameter	r_h	0.000576	m^3 $(g\ SS)^{-1}$
Flocculant zone settling parameter	r_p	0.00286	m^3 $(g\ SS)^{-1}$
Non-settleable fraction	f_{ns}	0.00228	dimensionless

The 'noreac1d' settling model specified for GPS-X users is essentially the model required by the benchmark, but users should be aware of one important difference. The *'simulation benchmark'* specified settler is a 10-layer non-reactive settler, which is consistent with the 'noreac1d' model. However, it is assumed in the benchmark settler, that both the solids and solubles will be subjected to the 10-layer configuration. This is not the case with the 'noreac1d' model. In this GPS-X model, the solids and the settling processes are modelled using the 10 layers, but the solubles are not. Instead, the solubles are 'subject to a complete mix zone' (GPS-X Technical reference), in essence a 1-layer reactor. This deviation has no effect on the steady state solution, but examination of this deviation under dynamic conditions reveals that this approach has a 'smoothing' effect on the effluent data. That is, under dynamic conditions, users should be aware of the fact that less variability will be observed in the GPS-X effluent data as compared to an identical system using a 10-layer soluble settler model. However, mass balance constraints ensure that, on average, this approach will have no effect on the discharge mass on any soluble component.

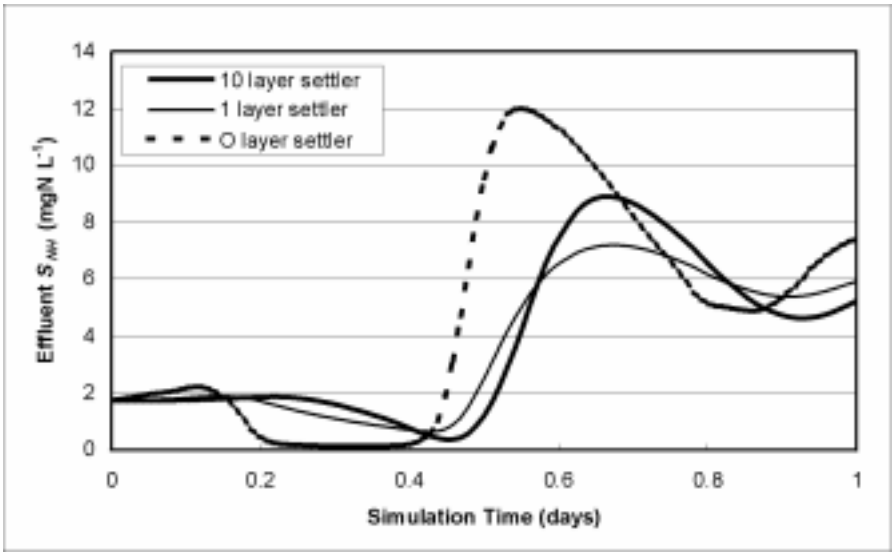

Figure 7.3: Illustrative example of the impact of the number of soluble settler layers on the effluent variability of a soluble component.

Figure 7.3 shows the differences in effluent variability for ammonia (S_{NH}) using a 10-layer soluble settler (i.e. ideal benchmark case), a 1-layer soluble settler (i.e. the 'noreac1d' model) and a 0-layer soluble settler (i.e. the solubles are mapped directly to the effluent without accounting for the settler volume). This deviation from the benchmark description is unfortunate, and users should be aware of this deviation, but as there is no suitable alternative in GPS-X, users need not try to correct it.

Figure 7.4 shows the 'physical' dialogue box and its associated benchmark parameter values for the settler. In particular the reader's attention is drawn to the settler feed point of 2.2 m. This distance pinpoints the settler feed to the middle of the 6th settler layer and is consistent with the benchmark description.

Figure 7.4: 'Physical' dialogue box for the settler showing the required settler feed point depth.

7.2.3 Dissolved Oxygen Modelling

Two aspects of dissolved oxygen (DO) modelling in GPS-X should be addressed. These include the calculation of DO saturation and DO modelling in the return sludge stream from the underflow of the secondary settler.

The '*simulation benchmark*' specifies a DO saturation concentration of 8 g m^{-3}, but achieving this with GPS-X requires a comprehensive understanding of how GPS-X calculates its saturation concentration. Table 7.6 provides a listing of the DO saturation variable values that are required for the '*simulation benchmark*', but the following equations are presented to explain their origin. The DO saturation concentration in GPS-X is calculated using the following equation:

$$SOST = \frac{uc \bullet \beta \bullet \rho \bullet P_{O_2}}{HenryO_2} \tag{7.4}$$

where:
- *SOST* is the saturation concentration of DO (gO_2 m^{-3})
- *uc* is 1777.8 (gO_2 m^{-3} H_2O)
- β is the salts and ions correction factor
- ρ is the density of water (kg m^{-3})

- P_{O_2} is the partial pressure of oxygen (atm)
- $HenryO_2$ is the Henry's Law constant for DO (atm)

The Henry's Law constant is calculated as follows:

$$HenryO_2 = (708.0 \cdot T) + 25700 \quad \rightarrow T = 20^oC \text{ assumed} \qquad (7.5)$$
$$= 39860$$

with the density of water calculated using the following equation:

$$\rho = 999.96 + (2.29 \times 10^{-2} \cdot T) - (5.44 \times 10^{-3} \cdot T^2) \quad \rightarrow T = 20^oC \text{ assumed} \qquad (7.6)$$
$$= 998.242$$

and the partial pressure of oxygen calculated using:

$$P_{O_2} = FractionO_2 \bullet \left(P_{atm} + \frac{depth \bullet \rho \bullet g}{101325 \bullet 2} \right) \quad \rightarrow P_{atm} = 1 \text{ assumed} \qquad (7.7)$$
$$= 0.21 \quad \rightarrow depth = 0 \text{ m}$$

For these DO saturation calculations, the use of 'global settings' is recommended. The values calculated above should be entered in the 'physical' dialogue box of the GENERAL DATA menu. Figure 7.5 shows the applicable dialogue box and the required benchmark settings.

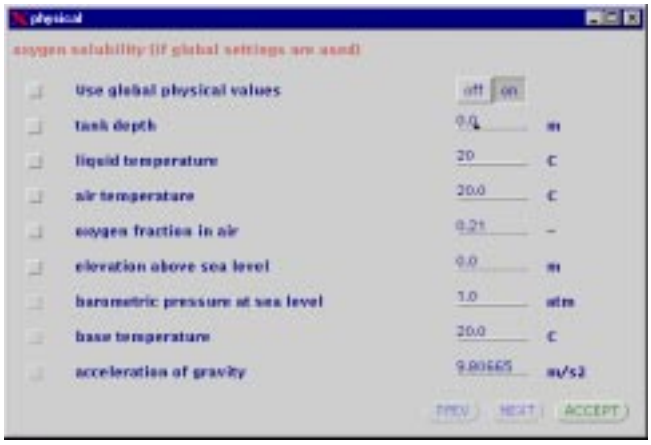

Figure 7.5: The GENERAL DATA menu 'physical' dialogue box showing the required DO parameter values to achieve a DO saturation concentration of 8 g m^{-3}.

Using the values calculated above and Equation 7.4, it is possible to calculate the required beta correction factor (β) to achieve a DO saturation concentration of 8 g m^{-3}. This value (Table 7.6) can then be entered in the 'operational' dialogue box for each reactor.

Table 7.6: GPS-X required DO related parameter values needed to achieve the *'simulation benchmark'* DO saturation concentration of 8 g m^{-3}.

DO Related Parameter	Benchmark Required Value	Units
Tank depth	0	m
Liquid temperature	20	^{o}C
Air temperature	20	^{o}C
Oxygen fraction in air	0.21	-
Beta factor (for DO saturation)	0.855636	-

One further DO modelling parameter needs to be specified correctly to complete the DO modelling issues related to the benchmark (Figure 7.6). This last parameter, 'critical sludge blanket level' relates to the modelling of DO in the return sludge flow from the underflow of the clarifier and 'is used to define the height of the sludge blanket in order for the dissolved oxygen in the underflow and pumped streams to be zero' (GPS-X Technical Reference). That is, with this parameter, users define a sludge blanket height that if exceeded will result in DO in the underflow and pumped flow streams being set to zero. However, this can be avoided by specifying a 'critical sludge blanket level' that is greater than the height of the clarifier. This way the simulated sludge blanket height will never exceed the defined level and the DO in these two streams will never be set to zero. Figure 7.6 shows a 'critical sludge blanket level' of 5 m, but any value greater than 4 m will ensure that the simulated system conforms to the benchmark specifications.

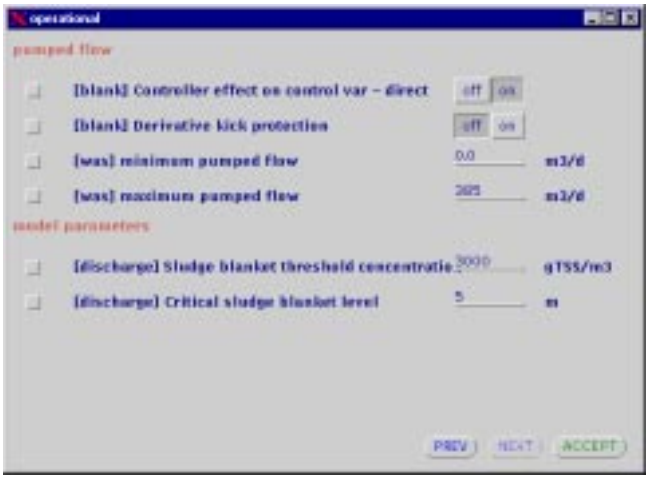

Figure 7.6: One screen of the settler 'operational' dialogue box showing the last DO modelling parameter of interest to the *'simulation benchmark'*: 'critical sludge blanket level'.

7.3 SIMULATION ISSUES – GPS-X

At the simulation stage, users should be aware of several numerical factors, and a number of adjustments are suggested. In particular, these adjustments relate to the criteria used for determining steady state, the communication time interval used during dynamic simulations and an influent data read-in problem (versions of GPS-X prior to 3.0) that may need correcting.

> **NOTE:** Although an investigation revealed that the choice of 'numerical solver' had no effect on the output data, it is suggested that users employ the Gear's Stiff numerical solver for '*simulation benchmark*' work.

The criteria used by the steady state solver impacts on the reproducibility of the calculated steady state such that with less tolerance allowed, more consistent results will be produced. Figure 7.7 shows the suggested parameter values for the 'steady state' dialogue box of the GENERAL DATA menu. In particular, attention is drawn to the 'error limit on individual variables' and the 'iteration termination criteria', which have both been lowered from the default values. It is suggested that these changes be implemented to increase the constraints on the steady state solver and thereby result in a more consistent and repeatable (to several decimals) steady state solution.

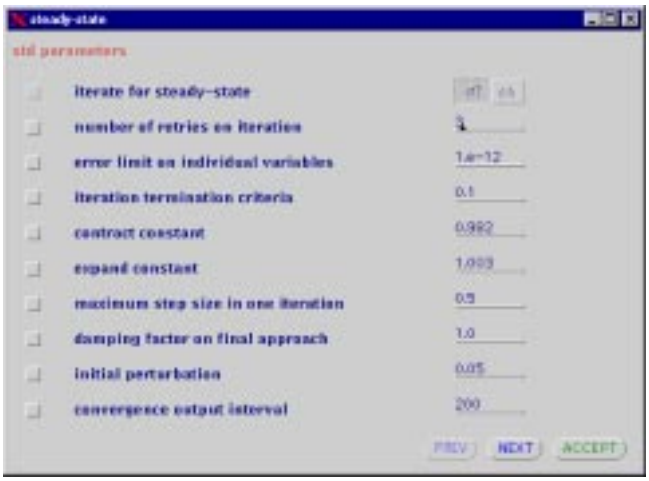

Figure 7.7: One screen of the GENERAL DATA menu 'steady state' dialogue box showing several suggested changes (from defaults) for the steady state solver.

The '*simulation benchmark*' influent data files are 14 days long, but in total, 28 days of dynamic simulation (following steady state) are required for each *weather* simulation. That is, the specified dynamic simulation procedure indicates that following the achievement of steady state, the system should be simulated dynamically for 14 days using the *dry weather* data file. After saving the system in the state achieved after this 14 days, each of the 14-day weather files (dry, rain & storm) is to be used, each time starting from that saved state. Two possibilities exist for executing these simulations. The first option involves using the 'Savestatus' and 'Readstatus'

commands found in the 'SETUP' menu of the *simulation control* dialogue box (Figure 7.8). After running the first 14-day *dry weather* simulation, the 'Savestatus' command can be used to save the achieved states in a file. Then, prior to each of the weather file simulations, the 'Readstatus' command can be used to reinstate those values from the file (it should be obvious that the 'STEADY STATE' check box must be unchecked for the 'Readstatus' command to have the desired effect). If the user is uncomfortable with this procedure an alternative procedure can be used. The alternative involves creating three new influent files, each of which contains 28 days of dynamic data (dry-dry, dry-rain & dry-storm). By using these files and the steady state solver option, each weather file simulation can be achieved in one step, using one click of the 'START' button in the *simulation control* dialogue box, which is shown in Figure 7.8.

Figure 7.8: 'Simulation control' dialogue box showing the use of the steady state solver option, the use of the necessary 0.25001 hours 'communication interval' and the 28-day simulation time.

According to the '*simulation benchmark*', the dynamic output data should be analysed at 15-minute intervals, which suggests a 'communication interval' of 0.25 hours (or 15 minutes) and is normally input in the 'simulation control' dialogue box (Figure 7.8). However, when this interval is explicitly entered, problems with the output data are observed. An analysis of the output data suggests that rounding errors result in the loss of several seconds over the course of a simulation. This causes the output data to be incorrectly recorded in the output file. As this has an impact on the performance index calculations, it is recommended that a 'communication interval' of 0.25001 hours be used instead. This eliminates the output data problem and adds less than 45 seconds to a 28-day simulation.

The last issue to be addressed is a minor problem that may appear in the influent data read-in. The benchmark influent data is specified to a maximum of 5 significant figures, but GPS-X versions prior to 2.4.1 allow a maximum of 4 significant figures for influent data. This results in the rounding of all influent flows and limits the precision on all influent components. It also causes differences to appear between the expected mass fluxes and the simulated mass fluxes. Users can test for this problem by dynamically simulating for an hour and checking the reported influent, waste and effluent flows against the flows that are expected. With the problem, all the influent flows will be rounded to the nearest 10 $m^3 d^{-1}$. If this problem is discovered it must be fixed, but

Hydromantis is aware of this potential problem with older versions of the software and has developed a solution, which is available directly from them.

7.4 BASIC CONTROL STRATEGY – GPS-X

Once users become familiar with the structure of GPS-X and how user defined code is incorporated into the executable code, it becomes a relatively simple task to implement the basic control strategy into GPS-X.

Each reactor object in GPS-X has a pre-defined dissolved oxygen control algorithm, which can be accessed through the reactor's *operational* dialogue box. The applicable dialogue box for tank 5 is shown in Figure 7.9. The velocity PI controller is consistent with the one specified in the *'simulation benchmark'* and should be used here for DO control.

Figure 7.9: Operational dialogue box for tank 5 with implemented DO control.

To implement the nitrate control, users will need to write a few lines of FORTRAN code in the layout.usr file and combine this code with the pumped flow controller associated with the 5th tank. Figure 7.10 gives an example of code that might be used to add noise and delay. In this example, *tdelay* is defined in the layout.con file and has an equivalent value of 10 minutes. The routine tracks the simulation time in terms of time blocks (based on the value of *tdelay*) and effectively delays the nitrate reading through two variables (*delayedno3anox2* and *conno3anox2*). The first variable, *delayedno3anox2*, stores the previous time block nitrate concentration plus noise and the second variable, *conno3anox2*, is used as the controlled variable. Only after the next time block is detected does *conno3anox2* get updated with the nitrate value observed 10 minutes previously. In this way, the variable *conno3anox2* represents the presumed sensor reading that has been delayed with noise added.

In the example shown in Figure 7.10, the *GAUSS* function is used to generate the random noise applied to the nitrate value. It should be noted that according to the ACSL manual, this function can be implemented in two ways, but when it was implemented in the other way, it did not work.

The reason for this is unknown. In addition, ACSL includes the *OU* function, which can be used to generate random numbers. However, as with the alternative *GAUSS* implementation, the *OU* function did not work as expected. Hence, it is recommended that users add noise using the *GAUSS* function as presented in Figure 7.10.

```
!*****************************************************************
***
macro userderivativesection
!DERIVATIVE SECTION

if (t.eq.0) then
      tbcounter = 0
endif

timeblock = INT((t/tdelay*1000+1)/1000)

if (tbcounter.eq.0) then
      conno3anox2 = snoanox2
      delayedno3anox2 = snoanox2
      savedtimeblock = timeblock
      tbcounter = tbcounter + 1
else
      if (timeblock.gt.savedtimeblock) then
            GAUSS(noise=0,0.1)
            savedtimeblock = timeblock
            conno3anox2 = delayedno3anox2
            delayedno3anox2 = snoanox2+noise
      endif
endif
```

Figure 7.10: Example of FORTRAN code used in the layout.usr file to add noise to the observed nitrate concentration and delay that concentration by 10 minutes.

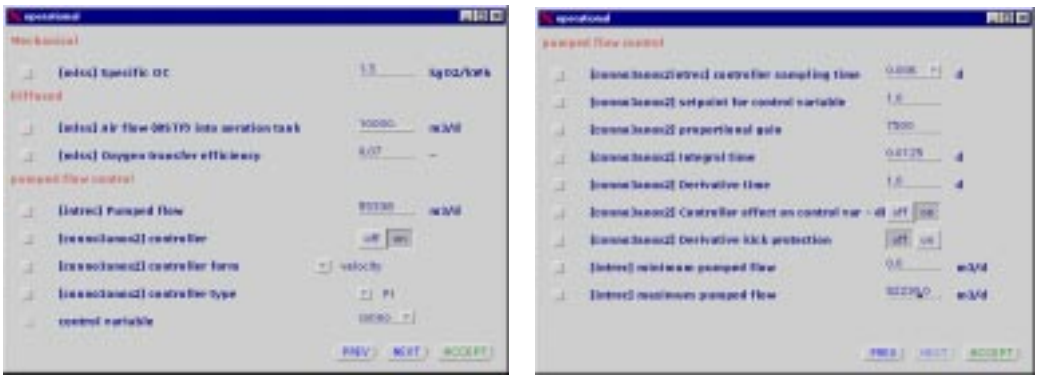

Figure 7.11: Dialogue boxes showing the implementation of the nitrate controller.

The second step in the control strategy implementation is to set-up the pumped flow control algorithm associated with the 5th tank. Figure 7.11 shows the applicable dialogue boxes. In this figure, the input terms are not readable, but the control variable is *conno3anox2* and the controller sampling time is 0.0069444 days (i.e. 10 minutes).

7.5 CONCLUSION

Tuning GPS-X to the *'simulation benchmark'* specifications requires that users be aware of how to overcome the differences between the simulator and the specifically defined *'simulation benchmark'*. This involves being aware of several GPS-X-specific features, options and calculation methods. Specifically, this chapter has explained several specific GPS-X variables and outlined several changes that should be made to some of the GPS-X default settings. Also explained was a 'fix' for a model error, a potential influent data read-in problem and an unresolvable deviation in the settler models. Finally, a GPS-X-specific alternative to the *'simulation benchmark'* defined simulation procedure was proposed. Nevertheless, with this knowledge at hand, it is a relatively straightforward task to implement the *'simulation benchmark'* into GPS-X, and thereby achieve benchmark consistent results.

7.6 ACKNOWLEDGEMENTS

The information contained in this chapter was compiled from various sources including personal experience with the software, and the GPS-X User Manual and Technical Reference. Also, input from Ulf Jeppsson and Hydromantis employees, Imre Takács, Bruce Gall and Eric Giroux should be fully acknowledged.

8

MATLAB™ & Simulink™

described by Ulf Jeppsson

NOTE: The issues and procedures outlined in this chapter are specific for the use of MATLAB/Simulink with the *'simulation benchmark'* and in no way should be interpreted as necessary procedures for using MATLAB/Simulink for any other purpose.

MATLAB (MATrix LABoratory) is a general, high-performance language for technical computing. It integrates computation, visualisation and programming in a common environment. MATLAB is an interactive system and includes a large library of predefined mathematical functions. It also provides the user with the possibility to extend this library with new functions. The code for such functions are based on mathematical notation and can often be formulated in a fraction of the time it would take to write similar programs in a scalar language, such as C or Fortran. Today, MATLAB is available for most hardware platforms (except for Macintosh computers where version 5.2 released in 1998 was the last one) and is considered to be one of the most fundamental software tools at many technical universities as well as industries all over the world.

MATLAB features a family of application-specific toolboxes that extend the MATLAB environment in order to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, optimisation, neural networks, system identification, statistics, symbolic mathematics and many more (approximately 30 toolboxes in total). Furthermore, it is possible to use MATLAB for on-line applications, to build application-specific graphical user interfaces and to create standalone (and platform independent) applications by automatic translation of MATLAB programs into C code. The wide field of applications that the toolboxes support is definitely one of the major advantages of MATLAB.

MATLAB™ and Simulink™ are a trademarked products of:
The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760-1500, USA
tel: +1 (508) 647-7000
fax: +1 (508) 647-7001
web: www.mathworks.com

Simulink is an add-on software product to MATLAB for modelling, simulating and analysing any type of dynamic system. MATLAB and Simulink are completely integrated, which means that all the functionality of the MATLAB toolboxes is available in the Simulink environment as well. Simulink provides a graphical user interface for building models as block diagrams and manipulating these blocks dynamically. It handles linear, non-linear, continuous-time, discrete-time, multi-variable and multi-rate systems. A large number of predefined building blocks are included and it is easy for the user to extend the functionality by customising these blocks or creating new ones. The models are simulated using a choice of integration algorithms, either from the Simulink menus or MATLAB's command window. The simulation results can then be put into the MATLAB workspace for post-processing and visualisation. Finally, the capabilities of Simulink may be further extended by the use of S-functions (system functions). S-functions can be written in the MATLAB language, C or Fortran, using a predefined syntax and allow users to add their own algorithms to Simulink models. Consequently, existing C or Fortran code may easily be incorporated and a dynamic system can be described as a mathematical set of equations instead of using block diagrams. The use of S-functions also has some implications with regard to performance, as will be discussed later in this chapter.

The implementation of the *'simulation benchmark'* in MATLAB/Simulink is relatively easy, but to achieve benchmark consistent results and good overall performance, users should be aware of certain aspects of the specifically defined *'simulation benchmark'* description. These have an impact on how the simulations should be carried out using MATLAB/Simulink.

This chapter suggests *one* possible way of implementing the *'simulation benchmark'* using MATLAB/Simulink to achieve consistent benchmark results. In particular, problems related to performance, algebraic loops, noise, delay, numerical solvers, hybrid systems, etc. are discussed and short code examples and illustrations are provided to enhance the understanding. However it should be made clear that other alternatives may work equally well.

8.1 CONFIGURATION ISSUES – MATLAB/SIMULINK

Set-up of the '*simulation benchmark*' configuration using the Simulink user interface can be done in several ways. Figure 8.1 shows one alternative using five bioreactor objects, two 2-way flow-combiner models, a 2-way flow-splitter model and a secondary clarifier model, all implemented as C-code S-function objects. Furthermore, a hydraulic delay model (to avoid algebraic loops) and models describing the nitrate sensor and the controllers for internal recirculation flow rate and oxygen concentration are used together with various connections and input blocks (plant input data and constants from the MATLAB workspace). It should be made clear that this is only one option. Nevertheless, the results of the simulations should be independent of the layout used provided the modelling options are entered correctly.

It is recommended that all model parameters, initial state variable values, plant input data, etc. be loaded into the MATLAB workspace before simulations are initiated and read from the Simulink environment using the name of the parameter rather than providing the explicit parameter value in Simulink. Special initialisation m-files should be used for this purpose. All values are then easily

accessible in text files instead of having to find the correct Simulink block to modify each time a certain parameter value needs modification.

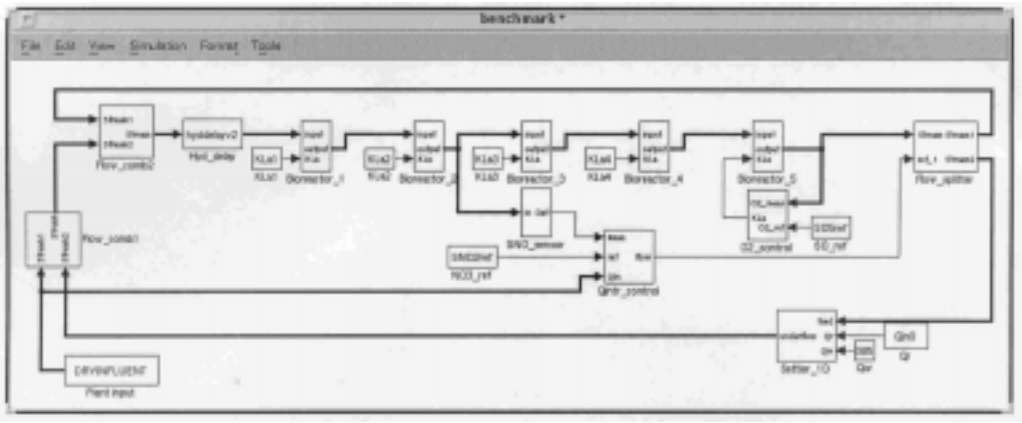

Figure 8.1: Interface layout of the COST *'simulation benchmark'* plant in Simulink (version 3.0).

8.2 MODEL ISSUES – MATLAB/SIMULINK

The required models describing the processes in the biological reactors and the settler can be implemented in three different ways. Firstly, the graphical user interface of Simulink may be used to build the models as block diagrams. Secondly, the MATLAB language can be used to build the models using a mathematical notation and the models can then be incorporated into the Simulink environment by the S-function block. Thirdly, a traditional scalar programming language (C or Fortran) can be used and when the code has been compiled (using an external compiler – a compiler is not included in MATLAB) the models may be included into Simulink by the use of the S-function block.

Using the graphical user interface is convenient for people who do not have significant experience writing mathematical models. However, when the models are highly complex, as is the case for the *'simulation benchmark'*, where the total number of state variables is approximately 150, the graphical description is not recommended. The block diagrams become so complex that programming errors are almost unavoidable and the errors are extremely difficult to find.

Using the MATLAB language is the easiest and most straightforward way to implement complex mathematical models for an experienced modeller. Unfortunately there is one major drawback: performance. MATLAB is an interpreting language, which means that every single line of an m-file is read and executed one after the other. This makes MATLAB slow compared to compiling languages (like C or Fortran) where all the code that is needed is compiled into binary machine code and then loaded into the CPU for execution. This problem does not appear when a model is built using only the Simulink graphical blocks (if no MATLAB Fcn blocks are used). However, every time a MATLAB language model is called from Simulink (by an S-function block), the

MATLAB interpreter is called at each time step and as the *'simulation benchmark'* is a dynamic, non-linear system this means that the interpreter is called very often. Consequently, performance may drop by a factor of 10 if the use of m-files is required by Simulink when a system is simulated.

Using a scalar programming language to implement the mathematical models is probably the most difficult way, especially if the user has limited or no previous experience of C or Fortran programming. However, if performance is an issue then it may well be worth the extra effort. Because the C or Fortran code models are compiled before they are called by Simulink (by an S-function block) these models do not have any negative effects on the overall performance of the simulation. As the *'simulation benchmark'* is a complex system using dynamic input data, which is to be simulated for a significant amount of time it is recommended that the user implements the biological reactor model (Henze *et al*., 1987) and the settler model in C or Fortran. Note that once validated, these models may be used in other types of wastewater treatment process simulations and need not be limited to the *'simulation benchmark'*. A C-code example of the biological reactor model using the S-function predefined syntax is provided in Section 8.6.1.

A common problem when modelling the biological reactors of the *'simulation benchmark'* is that some concentrations may show negative values during a dynamic simulation (which is obviously not physically possible). This may be caused by certain conditions and disturbances of the system in combination with the selected tolerance for the numerical solver, which allows the solver to take a time step that may be slightly too large. The problem is especially common for the oxygen concentration and the nitrate concentration in the anoxic reactors and ammonia concentration in the aerobic reactors but may appear for other variables as well. Once a concentration has become negative during a dynamic simulation then none of the results can be trusted as a negative sign in one concentration will immediately affect the other concentrations as well because the process rate equations are dependent. There is no built-in prevention of this in the ASM1 model, although the Monod-like process rate equations do help avoid this problem in most cases. It is recommended that the problem be solved using the following principle at every integration time step:

1. Check if the value of any state variable is negative.
2. If this is the case, then use the value 0 instead only when calculating the process rate equations, i.e. turn off the specific rate equations where the negative state variable is used.
3. Use the original values of all state variables when defining the differential equations, i.e. only adjust the state variable values when calculating the process rate equations.
4. Do not use additional limitations for the actual state variables in the model output description.

This will work if all processes that are consuming a substance are also limited with regard to this substance (e.g. by a Monod-like expression). For the aerobic growth of heterotrophs this is not the case with regards to the ammonia concentration and in some special cases (not for the *'simulation benchmark'*) it may be necessary to add an extra Monod function for ammonia in this process rate equation to avoid negative ammonia concentrations. If the model outputs also are limited to a minimum value of zero then this leads to incorrect mass balances. An example of how the above principle is implemented can be seen in the C-code example shown in Section 8.6.2 (the equations for `proc1` to `proc8` using the limited `xtemp` state vector).

Another common problem when modelling the *'simulation benchmark'* system is related to direct feed-through, which in turn creates so called algebraic loops. This generally occurs when an input to a Simulink block is driven by its own output, either directly or by a feedback path through other blocks with direct feed-through. When a model contains an algebraic loop, Simulink calls a loop solving routine at each time step. The loop solver performs iterations to determine the solution to the problem (if possible). As a result, models with algebraic loops run much slower (or not at all) than models without them. For the *'simulation benchmark'* the problem is related to the hydraulic flow through the reactors. No consideration is taken to how the flow propagates through the system, instead all flow-rate variations are instantaneous with no time delay. As the system exhibits internal feedback of the flow rate by the internal and external recycle flows, this may give rise to algebraic loops (also depending on how flow rate controllers are set up). Consequently, the input flow rate to a system depends directly on its output flow rate at the same time instant and the numerical solver runs into problems.

There are several ways to break up an algebraic loop. For example, various Simulink blocks may be used (e.g. the memory block, which delays an output signal with one time step), the flow equations may be solved analytically or a first-order reaction may be used for the flow rate in each reactor module. However, the author recommends benchmark users implement a special hydraulic delay C-code S-function module and insert it in a suitable position in the *'simulation benchmark'*, for example just before the first biological reactor (i.e. block Hyd_delay in Figure 8.1). Naturally, every artificial delay will have a slight impact on the dynamic behaviour of a system (in steady state there will be no effect) but these effects can be made so small that they are, for all intents and purposes, essentially undetectable. A first-order response (i.e. a low-pass filter) is used to avoid a direct feed-through of the flow rate. The mathematical expression for such a first-order response is:

$$\frac{dQ}{dt} = \frac{(Q_{in} - Q)}{T} \quad and \quad Q_{out} = Q \tag{8.1}$$

where: Q represents the flow rate and T is the time constant.

The chosen value for the time constant is a compromise between required CPU time for the simulation and the effect on the dynamic behaviour of the system. A smaller value of T leads to more CPU intensive simulations but the smaller value decreases the dynamic effects. For the *'simulation benchmark'* the recommended value for T is 0.0001 days (i.e. 8.64 seconds), which is a very short time period in relation to the sample time of the input data to the *'simulation benchmark'* (equal to 900 seconds) and the process dynamics. In order to guarantee that the mass balances of the system are not affected by the 'delay' in Q, the same type of first-order reaction should also be implemented for all the state variables. However, the equation should not be based on the concentrations of the various components but rather on the mass of each component and then the concentrations should be recalculated using the 'delayed' flow rate. An example of the mathematical expression for such a first-order reaction is:

$$\frac{dS_S}{dt} = \frac{\left(S_{S,in} \bullet Q_{in} - S_S\right)}{T} \quad and \quad S_{S,out} = \frac{S_S}{Q} \tag{8.2}$$

where: S_S represents readily biodegradable substrate, and T and Q are the same as in Equation 1.1

Note that $S_{S,in}$ and $S_{S,out}$ are concentrations whereas the unit of S_S is mass per unit time. An example of how the above principle is fully implemented can be seen in the C-code example in Section 8.6.2.

8.3 SIMULATION ISSUES – MATLAB/SIMULINK

At the simulation stage the use of the Simulink environment is quite straightforward. However, there are some aspects with regard to the choice of numerical solvers and performance that need to be addressed. The *'simulation benchmark'* influent data files are 14 days long, but in total, 28 days of dynamic simulation (following steady state) are required for each *weather* simulation. That is, the specified dynamic simulation procedure indicates that following the achievement of steady state (using active controllers but constant input data), the system should be simulated dynamically for 14 days using the *dry weather* data file. After saving the system in the state achieved after this 14 days, each of the 14-day weather files (dry, rain & storm) is to be used, each time starting from that saved state.

8.3.1 Solving Routines

The first problem is to determine the steady state using the constant input data file (i.e. the stabilisation values). The steady state solvers available in MATLAB/Simulink often cannot find a steady state for the benchmark system (depending on the type of control that is implemented there may not even exist a true steady state). Instead, it is recommended that the benchmark system be simulated for 100-200 days and that the final values of the state variables be accepted as the steady-state result. Note that it is important that such a 200-day steady-state simulation be effective.

The type of system defined in the *'simulation benchmark'* is normally considered to be a stiff dynamic system, i.e. the time constants for the different processes involved vary significantly. Such systems are quite difficult to solve numerically unless special numerical solvers are used, which have been developed especially to deal with these difficulties. Simulink provides the user with approximately 15 different solvers. Some of these are specifically designed for solving continuous, stiff systems (i.e. `ode23s` based on a modified Rosenbrock formula and `ode15s` based on Gear's method). It would appear that such algorithms would be the most suitable for simulating the benchmark system. However, there is a problem. The implementation of the nitrate sensor as described in Section 8.4.2 (Figure 8.3) contains a delay and a sample-and-hold block. This means that although the rest of the *'simulation benchmark'* is described as a traditional

continuous system the nitrate sensor turns it into a *hybrid system*, i.e. a combination of continuous and discrete systems.

The stiff solvers work very poorly for hybrid systems. A solution would be to use a more traditional solver (`ode23` or `ode45` based on an explicit Runge-Kutta formula) that does not experience such problems. On the other hand, the Runge-Kutta solvers are not the best ones for stiff systems. This is particularly important when such a system approaches steady state and the solver cannot take advantage of the large integration time steps used by a stiff solver. Because the Runge-Kutta solvers are limited in this way, a 200-day simulation would require a huge amount of CPU time (see Table 8.1). A much more effective way is to remove the noise source, the delay and the sample-and-hold block from the description of the nitrate sensor (and in all other parts of the system where similar implementations may be used in he future). That is, for the steady state case, the sensor should be considered ideal (i.e. have one benchmark-steady-state model and one benchmark-dynamic model). This will have no effect on the steady-state results (the noise and delay only affect the dynamic behaviour of the system). Moreover, the steady-state results are only used to provide reasonable initial values for the '*simulation benchmark*' before starting the dynamic simulations. The results shown in Table 8.1 demonstrate the need to use the proper numerical solver for the benchmark steady state simulations. Obviously `ode15s` or `ode23tb` should be chosen. For more information about the various numerical solvers the reader is advised to consult the Simulink user manual.

Table 8.1: Required CPU time to simulate the non-hybrid benchmark model for 200 days using constant input data and different numerical solvers (results from a 440 MHz Sun computer). Random function used to initialise state variables and same numerical tolerances used for all solvers.

Numerical solver	Required CPU time	Normalised CPU time
ode23 (Runge-Kutta)	35 minutes	> 100
ode45 (Runge-Kutta)	45 minutes	> 100
ode113 (Adams)	50 minutes	> 100
ode15s (Gear)	10 seconds	1
ode23s (Rosenbrock)	5 minutes	30
ode23t (trapezoidal)	days	> 100
ode23tb (TR-BDF2)	13 seconds	1.3

Note that some numerical problems may occur when using the best solvers for finding the steady-state solution. If the initial values are too close to the correct values or if simulations are run for a very long time ($\geq$ 500 days) the solver may fail to make any progress. It is not known at this time if this is related to the numerical resolution of the computer or a software-related problem.

When using the dynamic weather data files the measurement noise and time delay for the nitrate sensor model should be used and consequently the `ode45` numerical solver is the best choice. Because of the dynamics of the input data, the CPU time required for solving the system with the `ode45` or `ode15s` is not much different even if a *hybrid system* is avoided. As the input data are changing at 15-minute intervals, a stiff solver does not benefit from its capability to use larger time steps. Table 8.2 shows some examples of the required CPU time for different solvers when simulating the *hybrid system* for 14 days using the *dry weather* input data file. The results shown

in Table 8.2 demonstrate the need to use the proper numerical solver for the dynamic *'simulation benchmark'*. Obviously `ode23` or `ode45` should be used in this case.

Table 8.2: Required CPU time to simulate the hybrid benchmark model for 14 days using dry weather input data and different numerical solvers (results from a 440 MHz Sun computer). All state variables are initialised identically and numerical tolerances for the solvers are the same.

Numerical solver	Required CPU time	Normalised CPU time
ode23 (Runge-Kutta)	130 seconds	1
ode45 (Runge-Kutta)	175 seconds	1.35
ode113 (Adams)	400 seconds	3.1
ode15s (Gear)	Hours	> 100
ode23s (Rosenbrock)	90 minutes	42
ode23t (trapezoidal)	Hours	> 100
ode23tb (TR-BDF2)	28 minutes	13

When using numerical solvers to simulate dynamic systems it is a good idea to use low values for the numerical tolerances at an early stage (to ensure that the results are correct) and then increase them to promote faster computer performance. For the *'simulation benchmark'* it has been verified that setting the relative tolerance to 10^{-4} and the absolute tolerance to 10^{-7} (compared to the default tolerances of 10^{-3} and auto, respectively) produces correct and reliable results for the Simulink implementation (both for the steady state and dynamic situations). Increasing the suggested tolerances by a factor of ten only leads to a *reduced* CPU time of 2% (for the dynamic two-week simulations), whereas a decrease in the tolerances by a factor of ten leads to an *increased* CPU time of 21% (without any noticeable improvement in the results). Consequently, the suggested tolerances appear to be a reasonable compromise.

8.3.2 Debugging

For debugging purposes, Simulink uses a consistency-checking tool that validates certain assumptions made by the numerical solvers. This tool is especially useful for making sure that S-functions adhere to the same rules as the Simulink built-in blocks. However, once all models have been validated, users should make sure that consistency checking is turned off. This is done in the *Simulation Parameter* window (Figure 8.2) by selecting the tab 'Diagnostics' and deactivating the tool. Running the *'simulation benchmark'* with active consistency checking will increase the required CPU time for the two-week dynamic simulations by approximately 25%.

8.3.3 Data Processing

The results from a Simulink simulation can be saved either to the MATLAB workspace or to files. The choice does not have any significant impact, although saving to files increases the required CPU time somewhat. However, as the *'simulation benchmark'* evaluation tools are based only on the last week of each dynamic input data file and the required sampling time of the results is 15 minutes, there is no need to store results at each integration time step. If all state variables from all reactors (including the settler) are saved during a two-week dynamic simulation this will produce

approximately 100 Mb of data (depending on the selected tolerances and integration algorithm). If such an amount of data is saved to the MATLAB workspace then there is a distinct possibility that various MATLAB memory errors will occur. To avoid this, the results should be saved to files instead but the required CPU time will increase by 25-50%. Alternatively, it is better to use the 'produce specified output only' option in the Simulink simulation parameter window and only store results from the last week of the dynamic simulations using an explicit sample time of 15 minutes. If this is done then the amount of stored data from one dynamic benchmark simulation will be about 1.5 Mb. An example of how this is set up is shown in Figure 8.2.

Figure 8.2: Explicit results output times have been specified in the simulation parameter window. Results are saved from day 7 to day 14 with a time interval of 1/96 day = 15 minutes.

Once a simulation is finished and all the relevant results (state variables, controller outputs etc.) have been stored either in the MATLAB workspace or in data files it is a fairly simple task to write the necessary m-files for the performance assessment according to the benchmark description. Other types of m-files for plotting different results and setting up the benchmark system for a new simulation are also practical to enhance the use of the *'simulation benchmark'* in MATLAB/Simulink. If the user prefers to do the performance assessment in another software program (i.e. a spreadsheet program like Excel) the stored MATLAB files are easily exported. It is also possible to have on-line links between MATLAB and Excel so that data are immediately stored in a spreadsheet for further analysis.

8.4 BASIC CONTROL STRATEGY – MATLAB/SIMULINK

8.4.1 Hydraulic Delay Implications

Due to the effect of the hydraulic delay block some consideration must also be given to all types of flow rate variations that are used in the *'simulation benchmark'*. In particular, the basic *'simulation benchmark'* control strategy describes an on-line controller to modify the internal recycle flow rate (Q_a) from tank 5 back to tank 1 to maintain a specified nitrate concentration in the second biological reactor. Consequently, the value of Q_a may change from one time step to another. If all flow rates were modified instantaneously this would not be a problem (although this would imply an algebraic loop). However, the delay in the flow rate (and all components in the state vector) prior to the first biological reactor creates some minor problems. For example, if Q_a is increased by 100% in a step-wise manner then the output flow rate from tank 5 immediately increases by this value. However, the input flow rate to tank 5 only increases as a first-order reaction because of the hydraulic delay block, i.e. after 8.64 seconds the input flow rate will have increased by 63% (the characteristics of the first-order response). As the volumes of all the tanks are considered constant, the flow rate from tank 5 to the settler must be reduced in order to maintain the relationship that the input and output flow rates to a reactor must be identical at all times. Such an involuntary change in the settler input flow rate will have an immediate effect on the effluent flow rate (as the volume of the settler is also constant) and produce unwanted spikes in the flow characteristics of the settler. This, in turn, will affect the transportation of material through the settler. The same type of reasoning is valid for all variations in flow in the *'simulation benchmark'* system as all flows are correlated and affected by the behaviour of the hydraulic delay block.

To minimise the above effects and also to make the simulations somewhat more realistic, it is suggested that users make use of the same type of first-order response as is shown in Equation 8.1 whenever a flow rate is modified during a simulation (in reality a modification of a flow rate is not instantaneous but changes as a continuous function). In the example above, this implies that the output from the internal recycle flow rate controller should be delayed by a first-order reaction before it affects the rest of the process. The recommended time constant for this reaction is 86.4 seconds ($T_1 = 10 \cdot T$). Using this time constant essentially eliminates all sudden spikes in flow rate as the time constant for any user-imposed flow rate modification is ten times larger than the time constant for the hydraulic delay block, which is only used to improve the behaviour of the numerical solver. Naturally, no first-order reactions should be used for the *'simulation benchmark'* plant input flow rate or the plant effluent flow rate but only for the controlled internal flow rate variations (including any future control of the wastage flow rate).

8.4.2 Nitrate Sensor

According to the description of the basic control strategy, the signal from the available nitrate sensor is not an ideal continuous signal. Rather, the measurements are delayed by 10 minutes, the sampling period is ten minutes and the measurement noise is white, normally distributed, zero-mean with a constant standard deviation of 0.1 mg N/l and a low-level detection limit of 0.1 mg

N/l. Figure 8.3 illustrates how the sensor might be easily implemented in Simulink using the available standard building blocks.

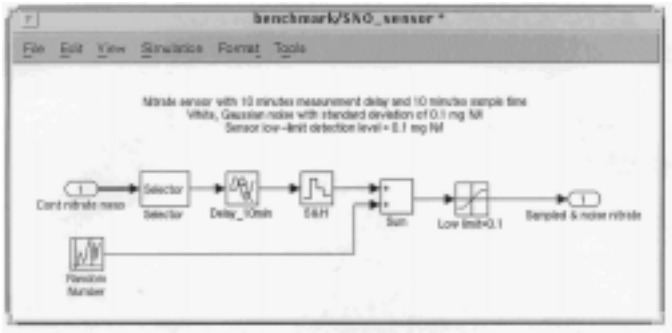

Figure 8.3: Block diagram of the nitrate sensor model in Simulink.

Preferably, the output from all sub-models should be verified independently. Diagnosing and correcting errors once all sub-models have been put together is a tiresome and difficult task. Figure 8.4 shows the effects of the delay and added noise on the nitrate measurement signal. This figure confirms the correct behaviour of the sub-model.

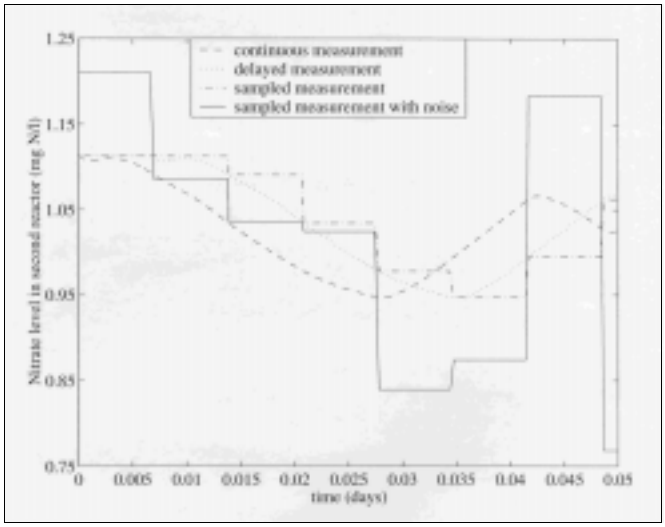

Figure 8.4: Verification of the behaviour of the nitrate sensor model.

8.5 CONCLUSION

The MATLAB/Simulink environment provides a versatile platform for a wide variety of simulation applications, including the '*simulation benchmark*'. Its flexibility and generality in combination with its world-wide use are strong arguments for the software. In addition, MATLAB/Simulink offers a number of tools to help users develop new control strategies and validate them using the '*simulation benchmark*'. However, there are some drawbacks especially with regard to performance and users should be aware of these drawbacks. Hence, the reasoning for recommending the use of C code to implement the complex models. Nevertheless, once the difficulties are understood and have been overcome, it is a relatively simple task to implement the '*simulation benchmark*' and take full advantage of all the other features of MATLAB/Simulink. Hopefully, the suggestions and recommendations provided in this chapter will assist future users achieve fast, accurate and consistent benchmark results.

8.6 MATLAB/SIMULINK - CODE EXAMPLES

8.6.1 MATLAB/Simulink - Example 1

C-code example of the biological reactor model to be included as an S-function in the Simulink environment.

```
/* ASM1 is a C-code S-function of the IAWQ AS Model No 1. */

#define S_FUNCTION_NAME asm1

#include "simstruc.h"
#include <math.h>

#define XINIT   ssGetArg(S,0)  /* initial state variable values */
#define PAR     ssGetArg(S,1)  /* model parameter values */
#define V       ssGetArg(S,2)  /* reactor volume m3 */
#define SOSAT   ssGetArg(S,3)  /* saturation concentration for DO */

/* mdlInitializeSizes - initialize the sizes array */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumContStates(S,13);    /* number of continuous states */
    /* order of states: S_I, S_S, X_I, X_S, X_BH, X_BA, X_P, S_O,*/
    /* S_NO, S_NH, S_ND, X_ND, S_ALK */
    ssSetNumDiscStates(S,0);     /* number of discrete states */
    ssSetNumInputs(S,16);        /* number of inputs */
    /* order of inputs: S_I, S_S, X_I, X_S, X_BH, X_BA, X_P, S_O,*/
    /* S_NO, S_NH, S_ND, X_ND, S_ALK, TSS, Q, kLa */
    ssSetNumOutputs(S,15);       /* number of outputs */
    /* order of outputs: S_I, S_S, X_I, X_S, X_BH, X_BA, X_P, S_O,*/
    /* S_NO, S_NH, S_ND, X_ND, S_ALK, TSS, Q */
    ssSetDirectFeedThrough(S,1); /* direct feedthrough flag */
    ssSetNumSampleTimes(S,1);    /* number of sample times  */
    ssSetNumSFcnParams(S,4);     /* number of input arguments */
    /* XINIT, PAR, V, SOSAT */
    ssSetNumRWork(S,0);          /* number of real work vector elements */
    ssSetNumIWork(S,0);          /* number of integer work vector elements */
    ssSetNumPWork(S,0);          /* number of pointer work vector elements */
}

/* mdlInitializeSampleTimes - initialize the sample times array */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

/* mdlInitializeConditions - initialize the states */
static void mdlInitializeConditions(double *x0, SimStruct *S)
{
int i;
for (i = 0; i < 13; i++) {
   x0[i] = mxGetPr(XINIT)[i];
}
}

/* mdlOutputs - compute the outputs */
```

```
static void mdlOutputs(double *y, double *x, double *u, SimStruct *S, int tid)
{
  double X_I2TSS, X_S2TSS, X_BH2TSS, X_BA2TSS, X_P2TSS;
  int i;
  X_I2TSS = mxGetPr(PAR)[19];
  X_S2TSS = mxGetPr(PAR)[20];
  X_BH2TSS = mxGetPr(PAR)[21];
  X_BA2TSS = mxGetPr(PAR)[22];
  X_P2TSS = mxGetPr(PAR)[23];
  for (i = 0; i < 13; i++) {
      y[i] = x[i];
  }
  y[13]=X_I2TSS*x[2]+X_S2TSS*x[3]+X_BH2TSS*x[4]+X_BA2TSS*x[5]+X_P2TSS*x[6];
  y[14]=u[14];
}

/* mdlUpdate - perform action at major integration time step */
static void mdlUpdate(double *x, double *u, SimStruct *S, int tid)
{
}

/* mdlDerivatives - compute the derivatives */
static void mdlDerivatives(double *dx, double *x, double *u, SimStruct *S, int
tid)
{
double mu_H, K_S, K_OH, K_NO, b_H, mu_A, K_NH, K_OA, b_A, ny_g, k_a, k_h, K_X,
ny_h;
double Y_H, Y_A, f_P, i_XB, i_XP;
double proc1, proc2, proc3, proc4, proc5, proc6, proc7, proc8;
double reac1, reac2, reac3, reac4, reac5, reac6, reac7, reac8, reac9, reac10,
reac11, reac12, reac13;
double vol, SO_sat, T;
double xtemp[13];
int i;

mu_H = mxGetPr(PAR)[0];
K_S = mxGetPr(PAR)[1];
K_OH = mxGetPr(PAR)[2];
K_NO = mxGetPr(PAR)[3];
b_H = mxGetPr(PAR)[4];
mu_A = mxGetPr(PAR)[5];
K_NH = mxGetPr(PAR)[6];
K_OA = mxGetPr(PAR)[7];
b_A = mxGetPr(PAR)[8];
ny_g = mxGetPr(PAR)[9];
k_a = mxGetPr(PAR)[10];
k_h = mxGetPr(PAR)[11];
K_X = mxGetPr(PAR)[12];
ny_h = mxGetPr(PAR)[13];
Y_H = mxGetPr(PAR)[14];
Y_A = mxGetPr(PAR)[15];
f_P = mxGetPr(PAR)[16];
i_XB = mxGetPr(PAR)[17];
i_XP = mxGetPr(PAR)[18];
vol = mxGetPr(V)[0];
SO_sat = mxGetPr(SOSAT)[0];

for (i = 0; i < 13; i++) {
   if (x[i] < 0)
     xtemp[i] = 0;
   else
     xtemp[i] = x[i];
}

proc1 = mu_H*(xtemp[1]/(K_S+xtemp[1]))*(xtemp[7]/(K_OH+xtemp[7]))*xtemp[4];
proc2 = mu_H*(xtemp[1]/(K_S+xtemp[1]))*(K_OH/(K_OH+xtemp[7]))*
(xtemp[8]/(K_NO+xtemp[8]))*ny_g*xtemp[4];
```

```
proc3 = mu_A*(xtemp[9]/(K_NH+xtemp[9]))*(xtemp[7]/(K_OA+xtemp[7]))*xtemp[5];
proc4 = b_H*xtemp[4];
proc5 = b_A*xtemp[5];
proc6 = k_a*xtemp[10]*xtemp[4];
proc7 =
k_h*((xtemp[3]/xtemp[4])/(K_X+(xtemp[3]/xtemp[4])))*((xtemp[7]/(K_OH+xtemp[7]))+ny
_h*
(K_OH/(K_OH+xtemp[7]))*(xtemp[8]/(K_NO+xtemp[8])))*xtemp[4];
proc8 = proc7*xtemp[11]/xtemp[3];

reac1 = 0;
reac2 = (-proc1-proc2)/Y_H+proc7;
reac3 = 0;
reac4 = (1-f_P)*(proc4+proc5)-proc7;
reac5 = proc1+proc2-proc4;
reac6 = proc3-proc5;
reac7 = f_P*(proc4+proc5);
reac8 = -((1-Y_H)/Y_H)*proc1-((4.57-Y_A)/Y_A)*proc3;
reac9 = -((1-Y_H)/(2.86*Y_H))*proc2+proc3/Y_A;
reac10 = -i_XB*(proc1+proc2)-(i_XB+(1/Y_A))*proc3+proc6;
reac11 = -proc6+proc8;
reac12 = (i_XB-f_P*i_XP)*(proc4+proc5)-proc8;
reac13 = -i_XB/14*proc1+((1-Y_H)/(14*2.86*Y_H)-(i_XB/14))*proc2-
((i_XB/14)+1/(7*Y_A))*proc3+proc6/14;

dx[0] = 1/vol*(u[14]*(u[0]-x[0]))+reac1;
dx[1] = 1/vol*(u[14]*(u[1]-x[1]))+reac2;
dx[2] = 1/vol*(u[14]*(u[2]-x[2]))+reac3;
dx[3] = 1/vol*(u[14]*(u[3]-x[3]))+reac4;
dx[4] = 1/vol*(u[14]*(u[4]-x[4]))+reac5;
dx[5] = 1/vol*(u[14]*(u[5]-x[5]))+reac6;
dx[6] = 1/vol*(u[14]*(u[6]-x[6]))+reac7;
dx[7] = 1/vol*(u[14]*(u[7]-x[7]))+reac8+u[15]*(SO_sat-x[7]);
dx[8] = 1/vol*(u[14]*(u[8]-x[8]))+reac9;
dx[9] = 1/vol*(u[14]*(u[9]-x[9]))+reac10;
dx[10] = 1/vol*(u[14]*(u[10]-x[10]))+reac11;
dx[11] = 1/vol*(u[14]*(u[11]-x[11]))+reac12;
dx[12] = 1/vol*(u[14]*(u[12]-x[12]))+reac13;
}

/* mdlTerminate - called when the simulation is terminated */
static void mdlTerminate(SimStruct *S)
{
}

#ifdef MATLAB_MEX_FILE     /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

8.6.2 MATLAB/Simulink - Example 2

C-code example of a hydraulic delay module to be included as an S-function in the Simulink environment to avoid problems with algebraic loops.

```
/* hyddelayv2 is a C-code S-function for a first-order reaction of flow and load
*/

#define S_FUNCTION_NAME hyddelayv2

#include "simstruc.h"
```

```
#define XINIT   ssGetArg(S,0)  /* initial state variable values */
#define PAR     ssGetArg(S,1)  /* model parameter value */
#define T       ssGetArg(S,2)  /* time constant for first-order reaction */

/* mdlInitializeSizes - initialize the sizes array */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumContStates(S,14);    /* number of continuous states */
    /* order of states: S_I, S_S, X_I, X_S, X_BH, X_BA, X_P, S_O,*/
    /* S_NO, S_NH, S_ND, X_ND, S_ALK, Q */
    ssSetNumDiscStates(S,0);     /* number of discrete states */
    ssSetNumInputs(S,15);        /* number of inputs */
    /* order of inputs: S_I, S_S, X_I, X_S, X_BH, X_BA, X_P, S_O,*/
    /* S_NO, S_NH, S_ND, X_ND, S_ALK, TSS, Q */
    ssSetNumOutputs(S,15);       /* number of outputs */
    /* order of outputs: S_I, S_S, X_I, X_S, X_BH, X_BA, X_P, S_O,*/
    /* S_NO, S_NH, S_ND, X_ND, S_ALK, TSS, Q */
    ssSetDirectFeedThrough(S,0); /* direct feedthrough flag */
    ssSetNumSampleTimes(S,1);    /* number of sample times */
    ssSetNumSFcnParams(S,3);     /* number of input arguments */
    /* XINIT, PAR, T */
    ssSetNumRWork(S,0);          /* number of real work vector elements */
    ssSetNumIWork(S,0);          /* number of integer work vector elements */
    ssSetNumPWork(S,0);          /* number of pointer work vector elements */
}

/* mdlInitializeSampleTimes - initialize the sample times array */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

/* mdlInitializeConditions - initialize the states */
static void mdlInitializeConditions(double *x0, SimStruct *S)
{
int i;
for (i = 0; i < 14; i++) {
   x0[i] = mxGetPr(XINIT)[i];
}
}

/* mdlOutputs - compute the outputs */
static void mdlOutputs(double *y, double *x, double *u, SimStruct *S, int tid)
{
  double X_I2TSS, X_S2TSS, X_BH2TSS, X_BA2TSS, X_P2TSS;
  int i;
  X_I2TSS = mxGetPr(PAR)[19];
  X_S2TSS = mxGetPr(PAR)[20];
  X_BH2TSS = mxGetPr(PAR)[21];
  X_BA2TSS = mxGetPr(PAR)[22];
  X_P2TSS = mxGetPr(PAR)[23];
  for (i = 0; i < 13; i++) {
      y[i] = x[i]/x[13];
  }
  y[13]=(X_I2TSS*x[2]+X_S2TSS*x[3]+X_BH2TSS*x[4]+X_BA2TSS*x[5]+
  X_P2TSS*x[6])/x[13];
  y[14]=x[13];
}

/* mdlUpdate - perform action at major integration time step */
static void mdlUpdate(double *x, double *u, SimStruct *S, int tid)
{
}

/* mdlDerivatives - compute the derivatives */
```

```
static void mdlDerivatives(double *dx, double *x, double *u, SimStruct *S, int
tid)
{
int i;
double timeconst;

timeconst = mxGetPr(T)[0];
if (timeconst > 0.0000001) {
  dx[0] = (u[0]*u[14]-x[0])/timeconst;
  dx[1] = (u[1]*u[14]-x[1])/timeconst;
  dx[2] = (u[2]*u[14]-x[2])/timeconst;
  dx[3] = (u[3]*u[14]-x[3])/timeconst;
  dx[4] = (u[4]*u[14]-x[4])/timeconst;
  dx[5] = (u[5]*u[14]-x[5])/timeconst;
  dx[6] = (u[6]*u[14]-x[6])/timeconst;
  dx[7] = (u[7]*u[14]-x[7])/timeconst;
  dx[8] = (u[8]*u[14]-x[8])/timeconst;
  dx[9] = (u[9]*u[14]-x[9])/timeconst;
  dx[10] = (u[10]*u[14]-x[10])/timeconst;
  dx[11] = (u[11]*u[14]-x[11])/timeconst;
  dx[12] = (u[12]*u[14]-x[12])/timeconst;
  dx[13] = (u[14]-x[13])/timeconst; }
else {
  dx[0] = 0;
  dx[1] = 0;
  dx[2] = 0;
  dx[3] = 0;
  dx[4] = 0;
  dx[5] = 0;
  dx[6] = 0;
  dx[7] = 0;
  dx[8] = 0;
  dx[9] = 0;
  dx[10] = 0;
  dx[11] = 0;
  dx[12] = 0;
  dx[13] = 0;
  x[0] = u[0]*u[14];
  x[1] = u[1]*u[14];
  x[2] = u[2]*u[14];
  x[3] = u[3]*u[14];
  x[4] = u[4]*u[14];
  x[5] = u[5]*u[14];
  x[6] = u[6]*u[14];
  x[7] = u[7]*u[14];
  x[8] = u[8]*u[14];
  x[9] = u[9]*u[14];
  x[10] = u[10]*u[14];
  x[11] = u[11]*u[14];
  x[12] = u[12]*u[14];
  x[13] = u[14];
}
}


/* mdlTerminate - called when the simulation is terminated */
static void mdlTerminate(SimStruct *S)
{
}

#ifdef MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"     /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"      /* Code generation registration function */
#endif
```

9

SIMBA®

described by Jens Alex

NOTE: The issues and procedures outlined in this chapter are specific for the use of SIMBA with the *'simulation benchmark'* and in no way should be interpreted as necessary procedures for using SIMBA for any other purpose.

SIMBA® is a simulation environment based on Matlab™ & Simulink™ for the simulation of wastewater treatment systems and includes a specific user interface for wastewater treatment system simulation. SIMBA extends Matlab/Simulink using block libraries for biological and chemical treatment processes (i.e. activated sludge and biofilm processes, chemical precipitation and sedimentation using different models, sewer systems…). Figure 9.1 shows the SIMBA control window and one of the block libraries of SIMBA.

From left to right, this library contains blocks for describing the influent, primary clarification, several different biological reactors, secondary clarification, flow distribution, measurement blocks and some frequently used blocks from the standard SIMULINK block sets. It should be noted that these blocks are not restricted to a particular activated sludge model but are usable with any biological/chemical model defining an arbitrary number of fractions and processes.

SIMBA includes several default models including ASM1, ASM2d, ASM3 and the Bio-P Model written at Delft University of Technology. However, because of the SIMBA structure, user-defined models can also be used. SIMBA facilitates the definition of new user-defined conversion models using a formalised matrix format (FOX – Formal Open matriX format) which is based on the matrix format propagated by the published IWA models (ASM1 - ASM3). The matrix

SIMBA® is a registered product of:
ifak e.V. Magdeburg, Steinfeldstr. 3, D-39-179 Barleben, Germany
tel: +49 39203 81920
fax: +49 39203 81939
e-mail: simba@ifak-system.com
web: www.ifak-system.com

description (ASCII file) is translated into a sparse matrix format and interpreted by the open reactor blocks during simulation. This technical solution ensures high model performance (comparable to blocks coded in C) and eliminates the need for users to manually code the model equations in C or FORTRAN (although this is still possible, if desired).

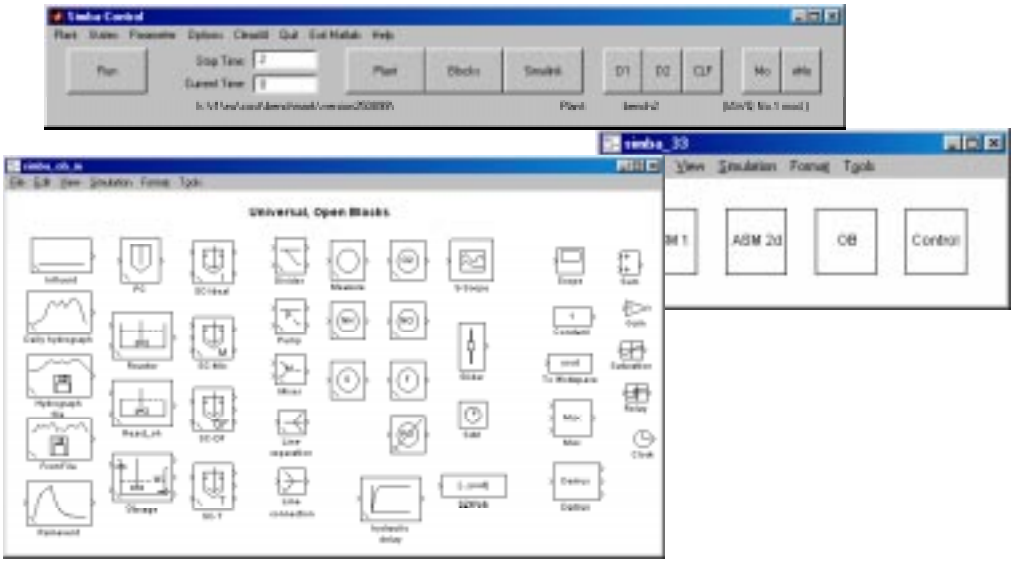

Figure 9.1: SIMBA control window and one of the SIMBA block libraries (open blocks).

As explained earlier, SIMBA is based on Matlab/Simulink, thus many of the issues previously discussed (Chapter 8) are also relevant here. Those issues will not be restated in this chapter, rather this chapter focuses on SIMBA-specific issues only.

9.1 MODEL ISSUES – SIMBA

9.1.1 Biological Process Model

The default ASM1 implementation in SIMBA contains some moderate modifications as compared to the original ASM1 publication. However, it is possible to implement user-defined models using so-called open blocks, where the conversion processes can be defined in a Matlab-Function (FOX-Format). For the *'simulation benchmark'*, the original ASM1 is required and therefore the default ASM1 implementation can not be used. Nevertheless, the original ASM1 model is available in SIMBA (Matlab-Function b_iawq1.m) and should be used for this implementation.

9.1.2 Settling Model

The Takács-Settler model is included in the SIMBA block libraries, but the included model is different to the *'simulation benchmark'* definition. That is, the SIMBA block does not lump all the particulate fractions together into one concentration but rather propagates all the particulate fractions separately to guarantee exact mass balances. To be in agreement with the *'simulation benchmark'* definition, two interface blocks should be built using Simulink to lump all particulate fractions into TSS and to estimate the particulate fractions back from the modelled TSS.

9.1.3 Dissolved Oxygen Modelling

Aeration per unit time using oxygen is calculated as follows:

$$Q_{O_2}(t) = \alpha_{O_2} \frac{S_{O_{SAT}} - S_O(t)}{S_{O_{SAT}}} Q_{air}(t) R_{AIR} h_{AIR} \qquad (1.1)$$

where:

α_{O_2} = oxygen transfer rate
$S_{o_{SAT}}$ = saturation of O_2-concentration (g m^{-3})
S_o = oxygen (negative COD) (g m^{-3})
R_{AIR} = O_2-flow per volume and immersion depth in clean water at $S_0 = 0$ (g $(m^3 m)^{-1}$)
h_{AIR} = immersion depth of pressurised air aeration (m)

The air flow rate is the default input signal of the reactor block in SIMBA. The equation used in SIMBA is comparable to the equation defined in the *'simulation benchmark'* description, but it is necessary to calculate the correct air flow rate to achieve the defined K_La value. To do this, a simple Matlab Function (kla2q.m) can be introduced.

9.2 CONFIGURATION ISSUES - SIMBA

Figure 9.2 shows one possible *'simulation benchmark'* implementation in SIMBA. To keep the flow scheme simple, several subsystems have been used. Figure 9.3 shows the subsystem for the biological reactors and Figure 9.4 shows the subsystem for the secondary settler.

Two special features should be noted in Figure 9.3. The first involves aeration and the second involves breaking an algebraic loop that is created by the configuration. To provide constant aeration (i.e. a constant K_La), a constant block utilising the function kla2q.m must be used and to avoid the detection of algebraic loops by Simulink, an artificial "hydraulic delay" block must be used. The method for implementing this hydraulic delay has been described elsewhere (Chapter 8) and will not be repeated here. Rather the reader is referred to the previous description.

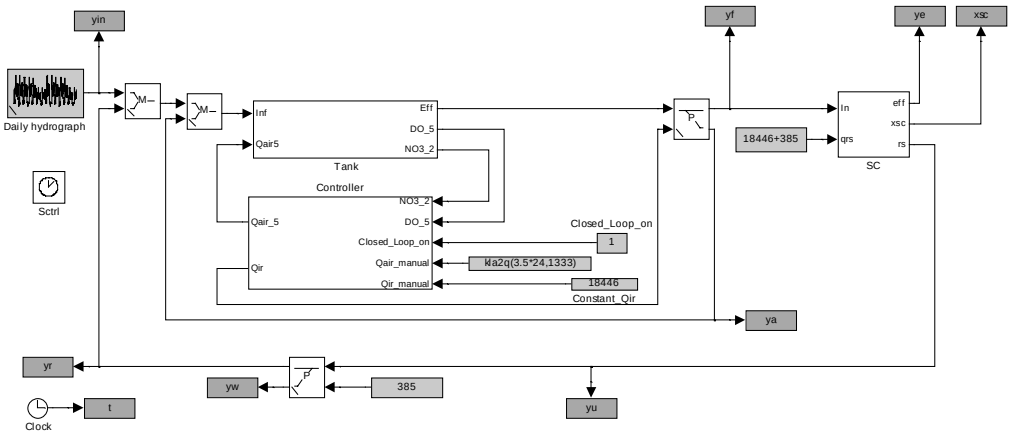

Figure 9.2: One possible implementation of the *'simulation benchmark'* in SIMBA

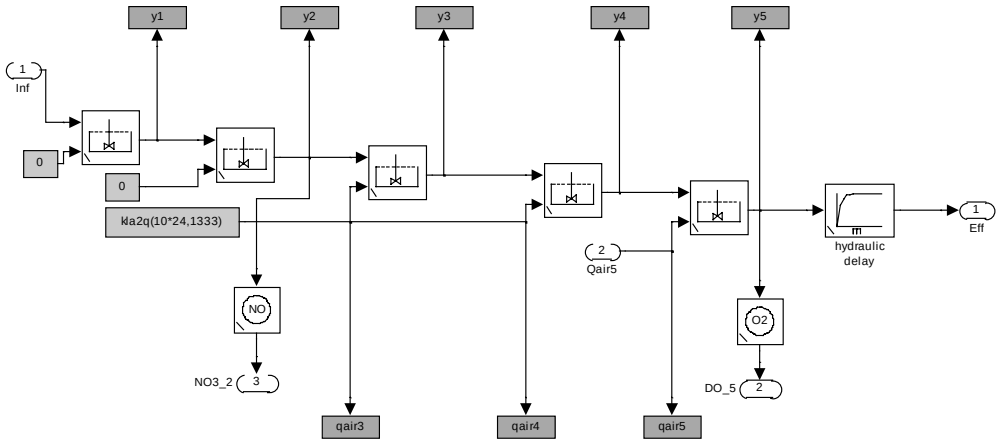

Figure 9.3: Subsystem for the biological reactors showing the aeration, sensor and hydraulic delay blocks.

Figure 9.4 shows the secondary settler subsystem and the necessary 'XF' and '1/XF' blocks used to lump and un-lump all the particulate fractions as described previously.

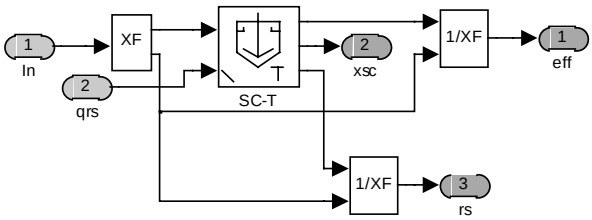

Figure 9.4: Subsystem for the secondary settler showing the settling and 'lumping' blocks.

The flow scheme of the controller subsystem is shown in Figure 9.5. Standard Simulink blocks and a special PI controller (part of SIMBA) are needed to build the benchmark controllers.

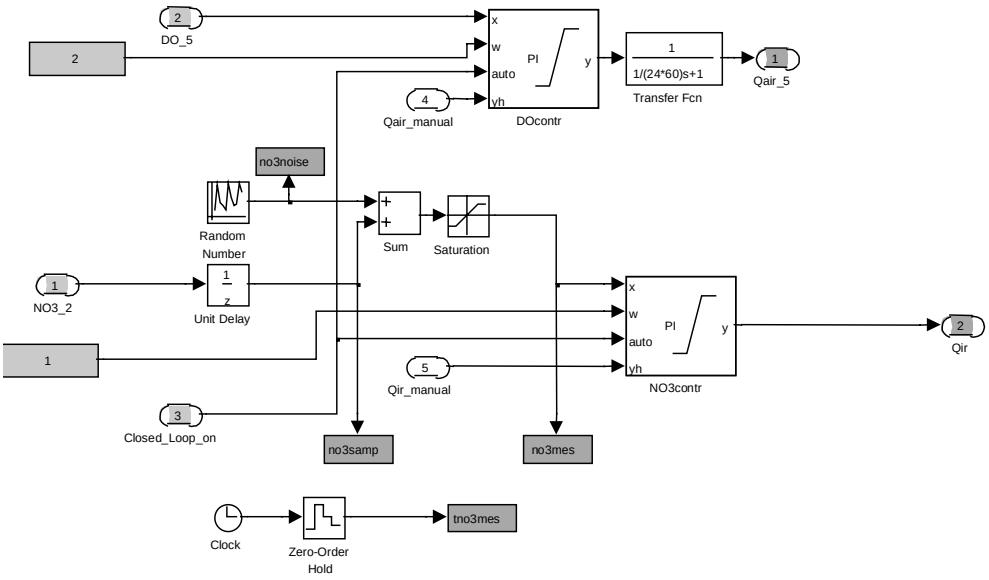

Figure 9.5: SIMBA representation of the controller subsystem.

9.3 SIMULATION ISSUES - SIMBA

NOTE: All the files described in this section are available at no cost from ifak System GmbH. Users who wish to receive these benchmark files should contact the company directly. Note also that SIMBA version 3.3^{+} (or higher) is required to run the benchmark in SIMBA.

The easiest way to simplify and to organise the *'simulation benchmark'* work is to prepare a number of Matlab script files. In this instance, to follow the benchmark procedures, several script files were created. Users are referred to the following script files for details of the parameter settings, simulation sequence and output evaluation.

Script name	Description
a2.m	• Runs the openloop steady state test, • Generates the start state for the dynamic openloop tests, • Creates HTML documentation of the steady state results
a3.m	• Runs the dynamic openloop tests for the different weather files, • Stores the results
a4.m	• Runs the dynamic closedloop tests for the different weather files, • Stores the results
aa.m	• Evaluates the open and closed loop results (performance index), • Generates HTML documentation of the results

For the evaluation of results, the following script files have been created:

Script name	Description
result1.m	• Creates HTML documentation of the steady state result
result2.m	• Evaluates performance index, • Generates HTML documentation

As an example, Figure 9.6 shows the HTML documentation generated after running a2.m. Similar files are created by the other HTML generating files.

Besides the script files described above that were developed specifically for the *'simulation benchmark'*, a number of standard SIMBA functions were also modified (as copies in the benchmark directory). The modified functions are listed below:

Function name	Description
i_iawq1.m	• Initialisation function of the activated sludge blocks, • DO saturation and temperature adjustment to the *'simulation benchmark'* settings
i_nkbto.m	• Initialisation function of the Takács-settler, • Definition of a reduced concentration vector (all particulate fractions lumped to TSS)
setpbio.m	• Function to set biological parameters of the activated sludge model to the *'simulation benchmark'* settings

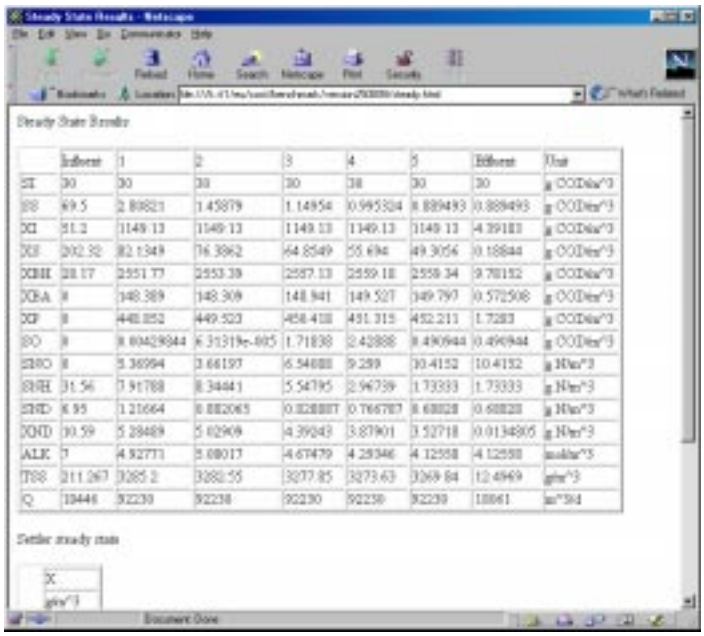

Figure 9.6: HTML documentation of the steady state results.

9.4 CONCLUSION

It is possible to implement the '*simulation benchmark*' in SIMBA without modifications to SIMBA itself. However, to fulfil all the '*simulation benchmark*' definitions, some effort is necessary to modify the default SIMBA settings and models. For instance, it is necessary to introduce an interface block to lump the particulate ASM1 fractions into a single TSS term for the settler model and introduce a second interface block to un-lump the TSS term back into ASM1 state variables after settling. The interaction with Matlab is essential as it makes the organisation of the benchmark procedures in scripts possible and makes the calculation of the performance criteria relatively easy. The prepared '*simulation benchmark*' files (and specifically the script files), give SIMBA users a very convenient way to start working with the '*simulation benchmark*' and hopefully will facilitate the use of SIMBA as a tool for the evaluation of a variety of control strategies.

10

STOAT™

described by Jeremy Dudley

NOTE: The issues and procedures outlined in this chapter are specific to the use of STOAT with the *'simulation benchmark'* and in no way should be interpreted as necessary procedures for using STOAT for any other purpose.

STOAT is a modular multipurpose modelling environment for the simulation of wastewater treatment systems. STOAT includes an implementation of ASM1, called *IAWQ #1*, and the Takács settler model, called *Generic*. Both of these models differ in several respects from the interpretation used by the benchmark. To make implementation of the *'simulation benchmark'* easier, STOAT has been updated at Version 4.1.6 to include benchmark-compatible implementations of these models. For both the aeration tank and the settling tank the models have been called *COST 682*. The standard models will not produce results that are comparable with the accepted *'simulation benchmark'* results so users will need to upgrade to Version 4.1.6 or later. Users should contact WRc for details.

10.1 MODEL ISSUES – STOAT

The default implementations of the ASM1 and Takács models in STOAT produce substantially different predictions from the accepted benchmark results. For this reason, the *COST 682* models were developed. These *'new'* models are consistent with those described in the COST *'simulation benchmark'*. The differences between the standard STOAT models and the benchmark models are described below.

STOAT™ is a trademarked products of:
WRc plc. Frankland Road, Swindon SN5 8YF, United Kingdom
tel: +44 1793 865185
fax: +44 1793 865001
e-mail: stoat@wrcplc.co.uk
web: www.wrcplc.co.uk

10.1.1 Biological Process Model

The state variables used in the STOAT model are not exactly the same as those used in ASM1. Table 10.1 shows the relationship between the STOAT and ASM1 variables. Note that STOAT does not use cryptic names for state variables, so there is no need to relate cryptic names.

The default biological process model in STOAT and the model specified in the *'simulation benchmark'* are essentially identical, but one difference, related to particulate degradable nitrogen (X_{ND}) in the influent, has been identified. In the benchmark, the value of X_{ND} in the influent is treated as being entirely available for degradation by the biomass. In the STOAT model, the value of X_{ND} available for degradation is reduced by the amount of nitrogen assumed to be held within the biomass:

$$X_{ND,available} = X_{ND,influent} - i_{XB} X_{B,H} \qquad (10.1)$$

Table 10.1: Comparison of state variables used in STOAT and ASM1.

STOAT	ASM1 equivalent
Inert soluble COD	S_I
Degradable soluble COD	S_S
Inert particulate COD	$X_I + X_P$
Degradable particulate COD	X_S
Heterotrophs	$X_{B,H}$
Autotrophs	$X_{B,A}$
Dissolved oxygen	S_O
Nitrate	S_{NO}
Ammonia	S_{NH}
Soluble degradable organic N	S_{ND}
Particulate degradable organic N	X_{ND}
Nonvolatile solids	*ignored*

This difference led to STOAT having a lower (and significantly so: 1.35 *versus* 1.73 mg/l) prediction of effluent ammonia. The benchmark implementation for STOAT has, therefore, been modified. However, it should be made clear that this is an inconsistency in the *'simulation benchmark'* description. That is, it is somewhat inconsistent to define a nitrogen content for the biomass (i.e. i_{XB}) in the reactor, but ignore that partitioning for the influent. Nevertheless, to remain consistent with the *'simulation benchmark'* description, the *'new'* model was developed.

It should further be noted that STOAT works in time units of **hours**, while the benchmark is set up in time units of **days**. This means that changes are required in the defined flow rates and calibration parameters to compensate for the time unit. The STOAT default stoichiometric and kinetic parameters correspond to the ASM1 and benchmark defined values. These are always adjusted for temperature and are quoted at a reference temperature of 15^0C.

10.1.2 Settling Model

In contrast to the biological model, many differences were noted when the defined *'simulation benchmark'* settler was compared to the original STOAT settling tank implementation. These differences are divided into two areas: the handling of nonsettleable solids, and the calculation of flux behaviour.

The Takács model separates out a value of suspended solids that is nonsettleable based on the instantaneous influent concentration, and calculates a settling velocity based on the total suspended solids concentration less the nonsettleable value; but the settling velocity is then applied to the nonsettleable fraction. In STOAT the nonsettleable fraction is assumed to have escaped the floc and does not have a settling velocity applied against it. In addition, the nonsettleable fraction is calculated as being a state variable, fed by the influent. Mathematically we have the following:

$$\text{Settling velocity } v(X) = \max(0, \min(v_{max}, v_0 [\, e^{-r_h(X - X_{\min})} - e^{-r_p(X - X_{\min})}) \qquad (10.2)$$

In the benchmark: $J = v(X)\, X$

In STOAT: $J = v(X)\, [X - X_{min}]$

In the benchmark: $X_{min} = f_{nss} \bullet X_f$

In STOAT: $dX_{min}/dt = Q\, (f_{nss} \bullet X_f - X_{min})]$; and this equation is implemented in the obvious way for each layer in the settler model.

The Takács model defines different settling flux models according to whether the location being looked at is above the feed point, at the feed point, or below the feed point. The STOAT implementation simplifies this and assumes that the settling process is independent of the feed point location, and only affected by the local concentration. Thus, in STOAT at every point the flux leaving a layer is calculated as:

$$\min(J_{clar,i}, J_{clar,i-1}) \qquad (10.3)$$

where:

$J_{clar,i}$ = if $X_{i-1} \leq X_{threshold}$ then J_i else $\min(J_i, J_{i-1})$
(and J is defined as above – remembering the STOAT and benchmark differences)

In the benchmark above the feed layer the flux leaving or entering a layer is calculated using the clarification flux, J_{clar}. At the feed layer, the flux entering is calculated using $J_{clar,}$ but the flux leaving is calculated as $\min(J_i, J_{i-1})$ and for all stages below the feed stage the flux leaving is calculated as $\min(J_i, J_{i-1})$. As with the biological model, these changes have been made for the STOAT benchmark model. The values used for the settling parameters are specified using hours as the time basis, and are given in Table 10.2.

Table 10.2: Comparative settling parameters as defined in the *'simulation benchmark'* and STOAT.

Parameter	Benchmark		STOAT	
	Value	Units	Value	Units
Maximum settling velocity	250	m day^{-1}	10.417	m hr^{-1}
Vesilind settling velocity	474	m day^{-1}	19.75	m hr^{-1}
Hindered settling parameter	0.000576	m^3 (g SS)$^{-1}$	0.000576	m^3 (g SS)$^{-1}$
Flocculant settling parameter	0.00286	m^3 (g SS)$^{-1}$	0.00286	m^3 (g SS)$^{-1}$
Nonsettleable fraction	0.00228	dimensionless	0.00228	dimensionless

In STOAT, layer 1 is at the surface of the settler. In the benchmark, layer 1 is at the base. The feed point is specified as the depth *from the surface*. Locating the feed 1.6 m from the surface places it in layer 6 from the base - layer 5 from the surface.

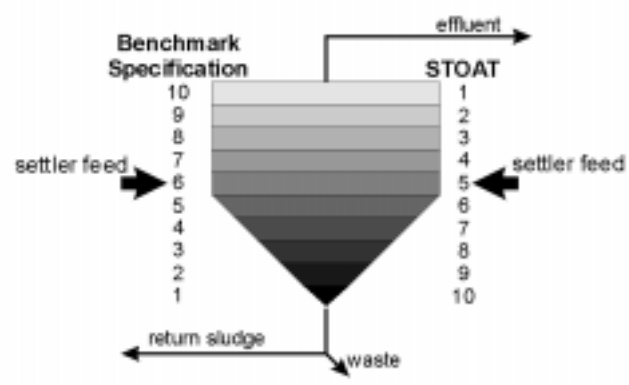

Figure 10.1: Settler layer numbering

10.2 CONFIGURATION ISSUES – STOAT

The *'simulation benchmark'* can be set up in several ways, but the easiest way is shown in Figure 10.2. This shows one influent, one aeration tank and one settler. The aeration tank is set up as 5 stages, while the settler is set up as 10 layers. The STOAT defaults are 1 stage for the aeration tank and 8 layers for the settler. The number of internal mixed liquor recycles is set as a property of the aeration tank — the default is zero. As discussed above, the *COST 682* process models should also be chosen for both the aeration tank and the settler.

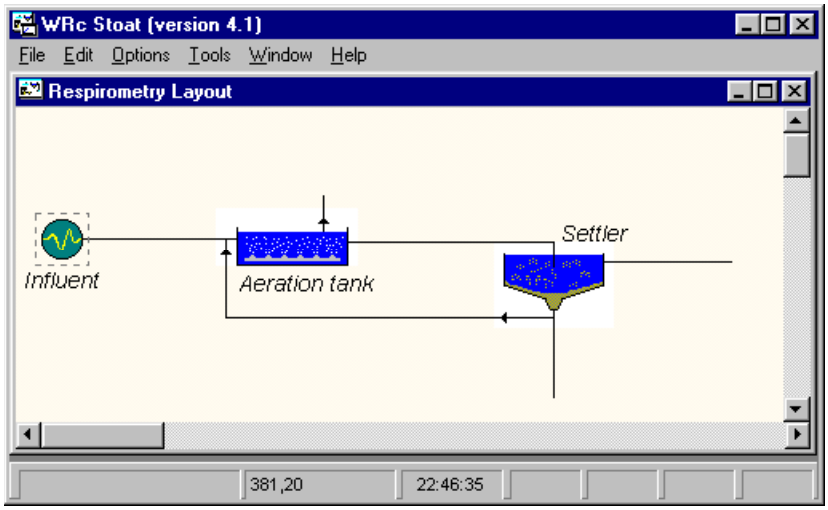

Figure 10.2: Interface layout of the COST *'simulation benchmark'* plant in STOAT.

10.3 SIMULATION ISSUES – STOAT

The default integration algorithm in STOAT is *RK-Cameron.* The benchmark recommends using a Gear solver and in STOAT, the recommended Gear solver is *MEDBF*. STOAT has no steady state solver, and the steady state solution must be achieved by running the system for 100 days, with the input constant and a maximum integration time step of 24 hours.

10.3.1 Influent Data Files

In STOAT, the *IAWQ #1* and *COST 682* biological process models treat total COD as the sum of four COD fractions: soluble biodegradable; soluble unbiodegradable; particulate biodegradable; and particulate unbiodegradable. Another variation users should be aware of involves the units used for influent heterotrophs and autotrophs. If the influent heterotroph and/or autotroph concentrations are non-zero then these are specified in *suspended solids* units, not *COD* units. Conversion between the two for the aeration model is made based on the user-defined value for the *COD to biomass ratio*. This has a default value of 0.5 (based on a BOD model) in STOAT up to Version 4.0 and 1.48 (when using a COD model) from Version 4.1. The particulate biodegradable COD *must* include the biomass COD, as the biomass COD will be subtracted from the influent COD when the influent enters the aeration tank.

The STOAT data file must contain the following data, comma separated, in the order listed. The computer must be set up to interpret the period (.) as the decimal separator. Table 10.3 shows the data file structure and flow-weighted *dry weather* data. The benchmark data can be entered using the STOAT influent data editor, in which case the values are limited to two decimal place

accuracy, or by creating the data file in a spreadsheet and saving the results as a comma-separated file.

Table 10.3: Illustrative example of STOAT influent data file structure showing the flow-weighted *dry weather* data and the applicable units.

Influent Component	Flow-weighted *dry weather* influent	Units
Flow	768.58	$m^3 hr^{-1}$
Temperature	15	^0C
pH	7	
COD of volatile fatty acids	[0 for benchmark]	
Soluble degradable COD	69.50	g COD m^{-3}
Soluble nondegradable COD	30.00	g COD m^{-3}
Particulate degradable COD	230.49	g COD m^{-3}
Particulate nondegradable COD	51.20	g COD m^{-3}
Volatile suspended solids	190.33	g SS m^{-3}
Nonvolatile suspended solids	14.70	g SS m^{-3}
Ammonia-N	31.56	g N m^{-3}
Nitrate-N	0	g N m^{-3}
Soluble organic N	6.95	g N m^{-3}
Particulate organic N	10.59	g N m^{-3}
Orthophosphate	[0 for benchmark]	
Dissolved oxygen	0	g COD m^{-3}
PHB in viable PAOs	[0 for benchmark]	
PHB in nonviable PAOs	[0 for benchmark]	
Poly-P in viable PAOs	[0 for benchmark]	
Poly-P in nonviable P users	[0 for benchmark]	
Viable nitrifiers	0	g SS m^{-3}
Nonviable nitrifiers	[0 for benchmark]	
Viable heterotrophs	19.03	g SS m^{-3}
Nonviable heterotrophs	[0 for benchmark]	
23 0s for components not relevant to the benchmark	↓	

10.3.2 Dissolved Oxygen Modelling

The benchmark specifies a saturation dissolved oxygen concentration (DO) of 8 g m^{-3}. In STOAT this is achieved by selecting, for the aeration tank, *Process Calibration* and making the calculation option *Specified by the user* rather than *Calculated automatically.* The user-specified DO can be set to 8 g m^{-3}.

10.4 BASIC CONTROL STRATEGY - STOAT

Implementing process control is fairly simple. The *'simulation benchmark'* basic control strategy makes use of two measured variables and two PI controllers. In STOAT, PI controllers for DO control are provided as part of the aeration object (providing that DO measurements are modelled as instantaneous with no measurement error, which fits the basic control strategy requirements). The control settings for DO control can be changed using the *Flow Distributions* menu. The edit dialogue is shown in Figure 10.3.

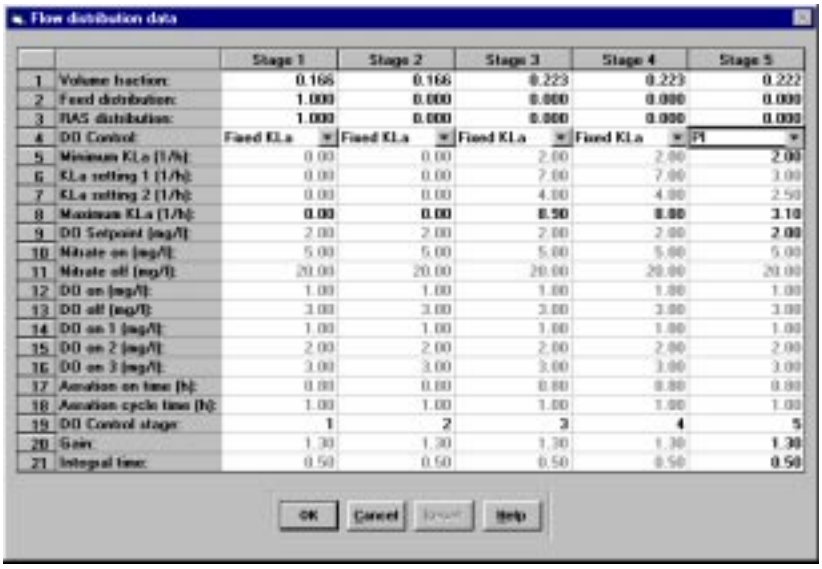

Figure 10.3: Flow distribution menu for DO control.

The minimum and maximum K_La values can be set, as can the DO setpoint and the PI parameters for gain and integral time. In STOAT, all PI controllers are implemented with integrator windup protection. That is, when the output reaches the minimum or maximum permitted values then the integration of the error term is discontinued, as continued integration would fail to provide any change in the output. This can provide more responsive control action at the extremes compared to PI schemes with no windup control. The PI controllers within the aeration object are implemented as differential equations, rather than difference equations, and are effectively equal to analogue, rather than digital, controllers.

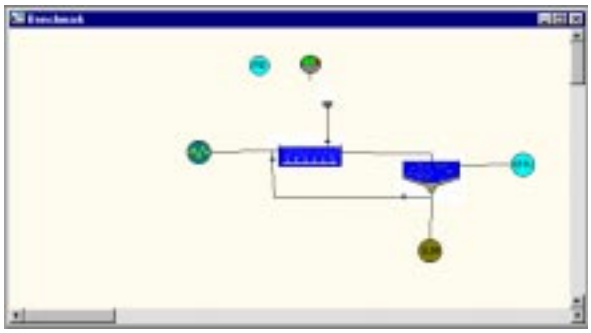

Figure 10.4: STOAT layout showing inclusion of a measuring point and PI controller.

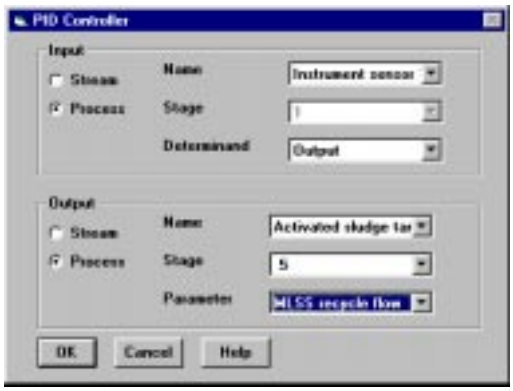

Figure 10.5: Nitrate PI controller signal set-up.

In contrast to the DO controller set-up, the nitrate controller requires the addition of a measuring point and a PI controller. The PI controller can be specified as analogue (integral term handled as a differential equation) or as digital (integral term handled as a difference equation) and the controller may be simple (shown in the following screen shots) or comprehensive. Figure 10.4 shows how the layout might look. A modification was added to STOAT 4.1.6 to provide support for a noisy instrument.

Figure 10.6: Nitrate PI controller settings.

The nitrate PI controller is set-up using the PID controller dialogue box as shown in Figure 10.5. This figure illustrates the input parameters for identifying the location of the input signal and control output. In this case, the control is based on whatever the instrument sensor reads, and is used to manipulate the flow of MLSS recycle leaving stage 5 of the aeration tank. The PID controller is then specified. P, PI and PID forms can be represented, and the controller gain,

integral time, and minimum and maximum outputs can be set. Figure 10.6 shows the applicable dialogue boxes.

The controller setpoint is set using the operation menu and the nitrate sensor connection is set-up as shown in Figure 10.7. In this instance, nitrate concentrations are taken from the sensor. There are several sensor models, and the required model for the basic control strategy is 'wet chemistry'. The location of the input signal is set up in a similar manner to that of the PI controller, as can be seen in Figure 10.7. Aspects of the sensor model can then specified including the sampling time, the sampling delay and the noise model (none, uniform or Normal). When using the 'Normal' model, the mean (for a biased sensor this is non-zero) and the standard deviation must be specified. To avoid the theoretical possibility of the infinite tails on a Normal distribution, minimum and maximum values are specified on the noise.

Figure 10.7: Nitrate PI controller sensor signal location.

10.5 CONCLUSION

Tuning STOAT to the *'simulation benchmark'* specifications requires that the user be aware of several STOAT-specific items. First, users must make use of the customised *COST 682* models for the aeration tank and final settler. And second, users must alter the defined *'simulation benchmark'* influent data files to allow for the inclusion of the biomass COD in the particulate biodegradable COD. With this knowledge it is then straightforward to implement the *'simulation benchmark'* in STOAT and achieve benchmark consistent results.

10.6 ACKNOWLEDGEMENTS

The information contained in this chapter was compiled from various sources including personal experience with the software and the STOAT manuals. Cristiano Bastianutti and Paolo Cirello provided additional help with the modelling.

11

WEST®

described by Peter A. Vanrolleghem & Sylvie Gillot

NOTE: The issues and procedures outlined in this chapter are specific to the use of WEST with the *'simulation benchmark'* and in no way should be interpreted as necessary procedures for using WEST for any other purpose.

WEST is a multi-platform modelling and experimentation system for the simulation of any system that can be represented by algebraic and differential equations. In WEST, ease of modelling a wide range of systems with maximum reuse of model knowledge is combined with high calculation performance. So far, WEST mainly has been used in the context of wastewater treatment research and is currently available for the following operating systems:

- Win 32: Windows 95, Windows 98, Windows NT 4.0 & Windows 2000
- Linux: Redhat, SuSE, Slackware, …
- SGI Irix 6.5
- IBM AIX 4.2
- Sun Solaris 2.6

The modelling environment is geared to support the rapid development of complex system models. Models are constructed by introducing model components (e.g. reactors, splitters, settlers, sensors, controllers, …) on a hierarchical graphical editor (HGE), linking the material and information flows between them and selecting the appropriate models from a modelbase. The open (plain text format) and hierarchically structured modelbase contains an extensive set of wastewater treatment component models and can be easily augmented with user-defined models. For instance, a new type of sensor can be quickly created by extending the basic sensor model available in the

WEST® is a registered product of:
HEMMIS N.V. Koning Leopold III-laan 2, B-8500 Kortrijk, Belgium
tel: +32 (0)56 37 26 37
fax: +32-(0)56 37 23 24
e-mail:info@hemmis.be
web: www.hemmiswest.com

modelbase with the specific properties of the new sensor to be modelled (e.g. the variable it measures, the noise type, measuring frequency). The model specification language (MSL Vs. 3.1) in which models are written, has been defined in such a way that particular information (e.g default parameter values or initial conditions, typical parameter ranges, variable constraint, units) can be included in the modelbase. In this way modelling is easier for non-expert users. Further, for the expert user procedures coded in C can be called from the MSL-code allowing, for instance, the use of existing high-performance numerical libraries.

Once the model is constructed, the WEST model transformer is launched and, in doing so, attention is diverted from user-friendly modelling support to raw calculation speed. WEST uses compiled code to achieve maximum calculation performance. Before compilation is initiated, however, the MSL-code of the different model components is combined into a set of algebraic and differential equations to which symbolic manipulation is applied. This allows a number of things including proper (dependency based) sorting of the equations, detection of algebraic loops (and if possible an immediate symbolic solution), and equation simplification. Finally, C-code is generated, compiled and linked.

The following sections describe the implementation of the *'simulation benchmark'* in WEST (Windows version 2.2 and higher), and outline the choices that are required to achieve benchmark consistent results.

The implementation of the benchmark into WEST is rather straightforward, as all benchmark specified models are available in WEST. Therefore, the implementation follows the normal procedure defined in detail on the web site (http://www.hemmiswest.com/) or in the WEST Quick Tour. Tuning WEST to the benchmark specifications simply consists in changing the default settings. A problem with the time interval of the output data, possibly influenced by the settings of the integration algorithm, is discussed.

11.1 CONFIGURATION ISSUES – WEST

The WEST layout of the *'simulation benchmark'* is shown in Figure 11.1. The benchmark configuration consists of five *activated_sludge_units*, two *two_splitters*, two *two_combiners*, a *secondary_clarifier*, two *loop_breakers*, a *CtoF*, a *FtoC*, a *sensor*, an *input*, an *output* and a *waste*. Once the layout has been created, models are assigned. Table 11.1 lists the WEST models that should be selected.

Table 11.1: WEST models to be used with the*'simulation benchmark'.*

Process	WEST Model
Biological (ASU)	FixVolumeASU
Settling (Clarifier)	SecondaryTakacsSolublesPropagator
Splitters	AbsTwoSplitter
Combiners	TwoCombiner
Loop breaker	DifferentialLoopBreaker
Waste	WasteFlux
Sensor	BenchmarkSensor

A benchmark specific model (*BenchmarkSensor*) has been created in order to calculate the effluent quality index, the effluent quality variance and the effluent violations (number of violations and percentage of time the plant is in violation) as the simulation runs.

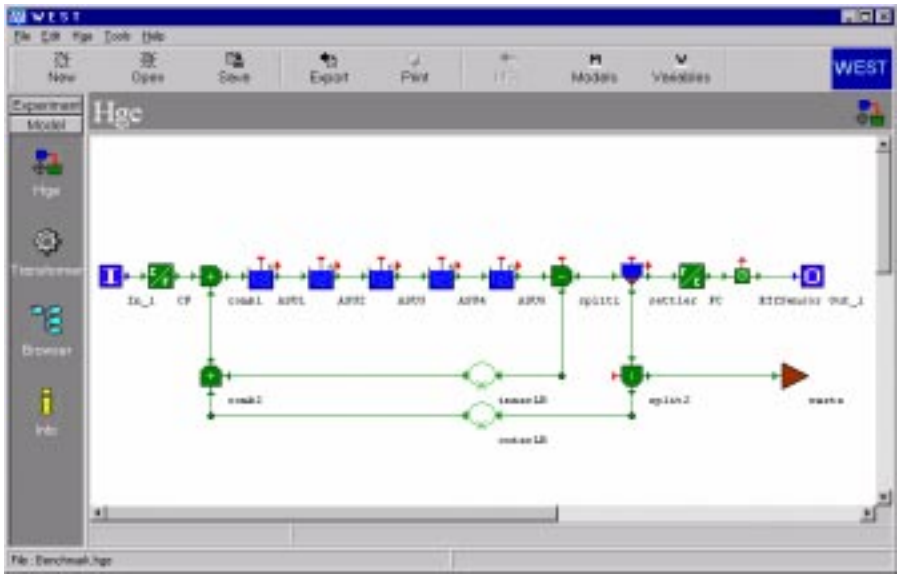

Figure 11.1: Interface layout of the open loop *'simulation benchmark'* in the Hierarchical Graphical Editor (HGE) in WEST.

Setting up the '*simulation benchmark'* configuration and assigning the models is done in the modelling environment of WEST. The model is then compiled and a dynamic link library (WEST model library, extension WML) is created using the transformer module. The WML library can then be loaded in the experimentation environment where the model parameters and initial conditions are set.

11.2 MODEL ISSUES – WEST

The *'simulation benchmark'* specifies two process models: the IAWQ Activated Sludge #1 – ASM1 (Henze *et al*., 1987) for the biological processes and the double-exponential settling function of Takács *et al*. (1991) for the settling process. Both process models are available in WEST, as seen in Table 11.1. The following sections outline the change in the default parameters required to meet the *'simulation benchmark'* specifications.

11.2.1 Biological Process Model

To work with the ASM1 state variables, *ASM1* should be selected as the WEST *Model Category* (Figure 11.2). Table 11.2 lists the state variables and symbols used in ASM1 and the WEST *FixVolumeASU* model. Note that in WEST, the state variables have mass units, but their concentrations are also available in the model.

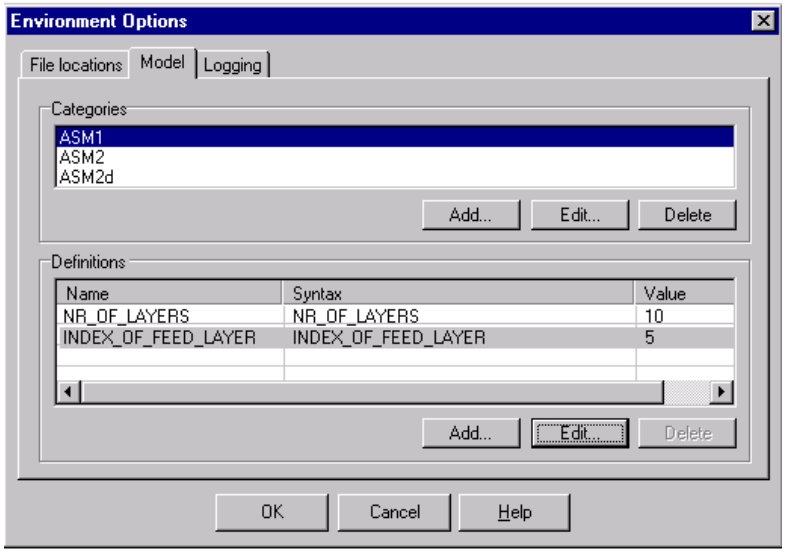

Figure 11.2: The definition of biological conversion model (ASM1) and settler properties in WEST.

Table 11.2: Comparison of the state variable symbols used in ASM1 and the WEST *FixVolumeASU* model.

State variable	ASM1 Symbol	WEST Symbol	WEST Units
Soluble inert organic matter	S_I	*S_I*	g COD
Readily biodegradable substrate	S_s	*S_S*	g COD
Particulate inert organic matter	X_I	*X_I*	g COD
Slowly biodegradable substrate	X_S	*X_S*	g COD
Active heterotrophic biomass	$X_{B,H}$	*X_BH*	g COD
Active autotrophic biomass	$X_{B,A}$	*X_BA*	g COD
Particulate products arising from biomass decay	X_P	*X_P*	g COD
Oxygen	S_O	*S_O*	g COD
Nitrate & nitrite nitrogen	S_{NO}	*S_NO*	g N
NH_4^+ & NH_3 nitrogen	S_{NH}	*S_NH*	g N
Soluble biodegradable organic nitrogen	S_{ND}	*S_ND*	g N
Particulate biodegradable organic nitrogen	X_{ND}	*X_ND*	g N

Table 11.3 and Table 11.4 list the stoichiometric and the kinetic parameters to be used with the WEST *FixVolumeASU* model. The parameter values are identical to those specified in the '*simulation benchmark*'.

Table 11.3: *'Simulation benchmark'* stoichiometric parameters and WEST *FixVolumeASU* model.

Description	ASM1 symbol	WEST symbol	WEST Value	Units
Fractions				
Fraction COD/TSS	f_{TC}	*F_TSS_COD*	0.75	g SS $gCOD^{-1}$
Heterotrophs				
Yield for heterotrophic biomass	Y_H	*Y_H*	0.67	gCOD $gCOD^{-1}$
Mass of nitrogen per mass of COD in biomass	i_{XB}	*i_X_B*	0.08	gN $gCOD^{-1}$
Fraction of biomass converted to inert matter	f_p	*f_P*	0.08	dimensionless
Mass of nitrogen per mass of COD in products	i_{XP}	*i_X_P*	0.06	gN $gCOD^{-1}$
Autotrophs				
Yield for autotrophic biomass	Y_A	*Y_A*	0.24	gCOD gN^{-1}

Figure 11.3 shows the '*simulation benchmark*' in the experimentation environment of WEST. From this window, the user can edit the model parameters and assign the *benchmark* specified values (Note in Figure 11.3 the properties that have been propagated from the modelbase into the experimentation environment).

Table 11.4: *'Simulation benchmark'* kinetic parameters and WEST *FixVolumeASU* model.

WEST description	ASM1 symbol	WEST symbol	WEST Value	Units
Heterotrophs				
Maximum specific growth rate for heterotrophic biomass	μ_H	*mu_H*	4.0	day^{-1}
Half-saturation coefficient for heterotrophic biomass	K_S	*K_S*	10.0	g COD m^{-3}
Half saturation coefficient for hydrolysis of slowly biodegradable substrate	K_X	*K_X*	0.1	g COD g COD^{-1}
Decay coefficient for heterotrophic biomass	b_H	*b_H*	0.3	day^{-1}
Correction factor for anoxic hydrolysis	η_h	*n_h*	0.8	dimensionless
Correction factor for anoxic growth of heterotrophs	η_g	*n_g*	0.8	dimensionless
Maximum specific ammonification rate	k_a	*k_a*	0.05	m^3 (g COD · day)$^{-1}$
Maximum specific hydrolysis rate	k_h	*k_h*	3.0	g X_S (g COD · day)$^{-1}$
Autotrophs				
Maximum specific growth rate for autotrophic biomass	μ_A	*mu_A*	0.5	day^{-1}
Decay coefficient for autotrophic biomass	b_A	*b_A*	0.05	day^{-1}
Ammonia half-saturation coefficient for autotrophic biomass	K_{NH}	*K_NH*	1.00	g NH_3-N m^{-3}
Switching Functions				
Oxygen half-saturation coefficient for heterotrophic biomass	$K_{O,H}$	*K_OH*	0.2	g O_2 m^{-3}
Nitrate half-saturation coefficient for denitrifying heterotrophic biomass	K_{NO}	*K_NO*	0.5	g NO_3-N m^{-3}
Oxygen half-saturation coefficient for autotrophic biomass	$K_{O,A}$	*K_OA*	0.4	g O_2 m^{-3}

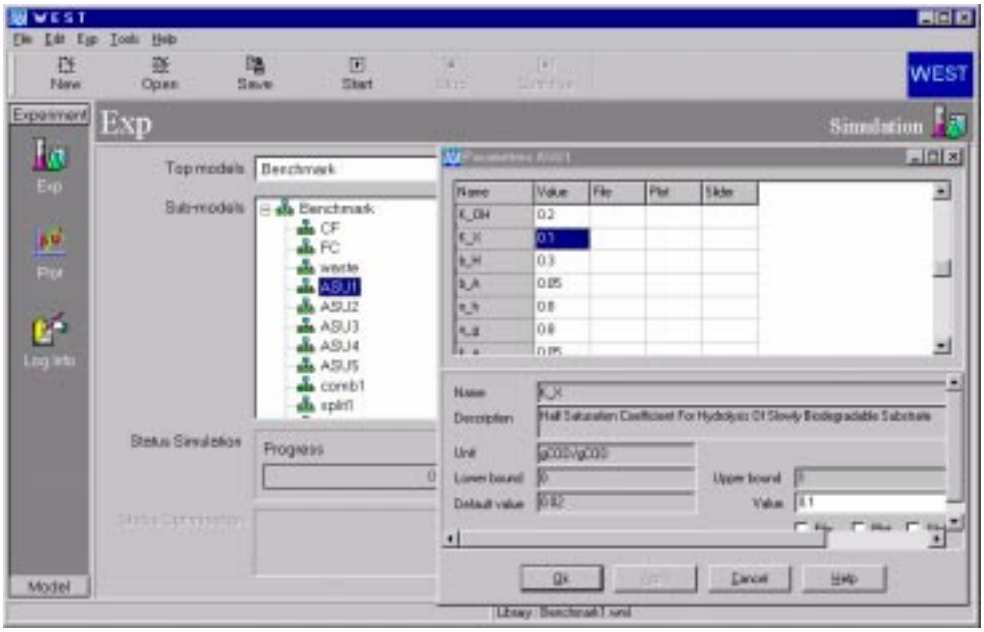

Figure 11.3: The COST Benchmark in the experimentation environment of WEST.

11.2.2 Settling Model

The double-exponential settling velocity function of Takács *et al*. (1991) was chosen as the '*simulation benchmark*' settling model due to its wide use and apparent acceptability as a fair representation of the settling process. The appropriate benchmark model is the *SecondaryTakacsSolublesPropagator* secondary clarifier model and needs to be selected from the secondary settler modelbase. The parameters in the Takács function and the associated values are listed in Table 11.5.

Table 11.5: Parameters and values of the settling model.

Parameter description	Takács Symbol	WEST Symbol	Value	Units
Maximum settling velocity	v'_0	*v_0*	250	m day^{-1}
Maximum Vesilind settling velocity	v_0	*v_00*	474	m day^{-1}
Hindered zone settling parameter	r_h	*r_h*	0.000576	m^3 (g SS)$^{-1}$
Flocculant zone settling parameter	r_p	*r_p*	0.00286	m^3 (g SS)$^{-1}$
Non-settleable fraction	f_{ns}	*f_ns*	0.00228	Dimensionless

By default, the feed to the settler is introduced at layer 5 in WEST (Figure 11.3); this is consistent with the *benchmark* description (as layers are counted from top to bottom in WEST).

11.2.3 Loop Breaker

The two recycles lead to algebraic loops in the model (direct feed through, i.e. when an input to a submodel is directly dependent on its own output, for instance with the recycle flows). During model transformation, the WEST algebraic loop detector will try to solve this loop problem by introducing an implicit loop solver (a symbolic solution is not found automatically), but this leads to a dramatic increase in computation time as this implicit solver uses iteration. The proper alternative is to introduce an explicit loop breaker. In WEST one can choose either a time delay or a first order loop breaker. In the current WEST benchmark a first order loop breaker was chosen with the time constant set to 0.0005 days to maximise calculation performance and minimise the deviation from the benchmark consistent simulation results.

11.3 SIMULATION ISSUES – WEST

Before running simulations, users have to define an *integrator*, an *input file* and an *output file* (and possibly, plots).

11.3.1 Integrator

As was shown also in the Matlab studies (Chapter 8), the numerical integrator preferred for this type of problem is the WEST default 4th order Runge-Kutta with variable time step size (*RK4ASC*). The parameters to be used for the integrator are shown in Figure 11.4.

Figure 11.4: Integrator window showing the required parameters for the integration algorithm.

11.3.2 Input File

Four different input files should be constructed: one for the steady state simulation (100 days) and three for the dynamic simulations (*dry weather*, *storm* and *rain*). Each dynamic file should consist of 14 days of *dry weather* data followed by 14 days of *dry weather*, *rain* or *storm* data, depending on the simulation.

11.3.3 Output File

The '*simulation benchmark*' recommends analysing the dynamic output data at 15-minute intervals. This corresponds to a *File communication interval* of 0.010417 d in WEST, as seen in Figure 11.5.

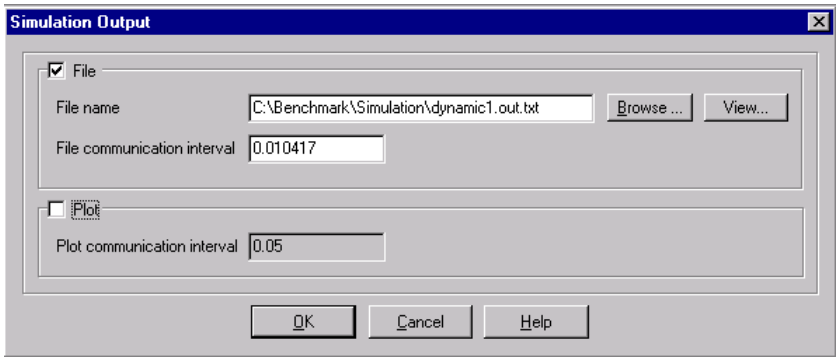

Figure 11.5: The WEST Simulation Output window.

NOTE: The RK4ASC integration method induces a variable integration step size. Therefore, the dynamic output data will not be given exactly at 15-minute intervals. To minimise the effect of this drawback, the user can either choose another integration algorithm (e.g. RK4) with fixed time step specified as an integer fraction of the desired communication interval. However, this would greatly increase the calculation time. So, it is suggested that a variable step size integration algorithm (with settings as in Figure 11.4) be used. It can be shown that the results (effluent quality index, operating costs) obtained with such an algorithm are consistent with the *'simulation benchmark'* results.

11.3.4 Data Processing

A dedicated Excel spreadsheet has been built (available upon request) to calculate the *'simulation benchmark'* performance criteria from the data compiled in the output file. Also, a benchmark-specific sensor (*BenchmarkSensor*) has been created to calculate, as the simulation runs, the effluent quality index, the effluent quality variance and the effluent violations (number of violations and percentage of time the plant is in violation). These values are reported when the simulation run has finished, but it should be noted that the availability of this information within the experimentation environment means that these quantities can be used directly in WEST optimisation experiments. That is, these quantities can be used directly for controller tuning where the object is to maximise performance based on the benchmark criteria.

11.4 BASIC CONTROL STRATEGY - WEST

The implementation of the basic control strategy is quite straightforward. Two additional sensors need to be incorporated in the graphical editor of the modelling environment, i.e. the standard *DO* sensor model and the *NO3GaussDelayHold* model (Figure 11.6). The dissolved oxygen and nitrate PI-controllers are included by incorporating two *PI* control blocks.

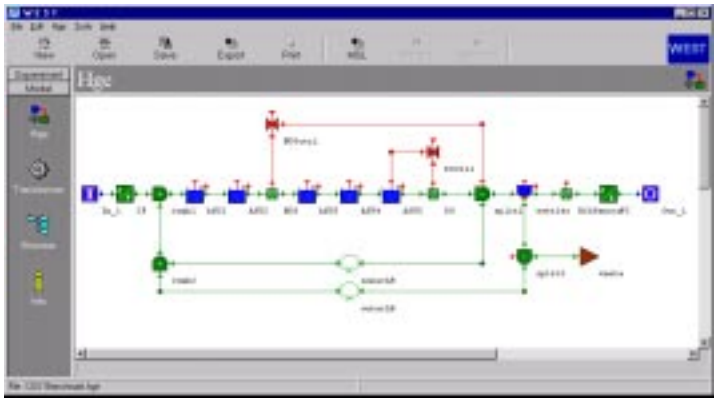

Figure 11.6: Interface layout of the closed loop *'simulation benchmark'* in WEST.

The sensor outputs are linked to the control inputs and the controller outputs are linked to the K_La of the 5th ASU (Figure 11.7) and the internal recycle flow rate, Q_a, respectively. Note that control loops are explicitly modelled in WEST (using sensor and controller blocks) and that information flows and material flows use different line colours (to avoid incorrect flow connections).

Figure 11.7: Connection specification window showing the linkage of the *DOctrl* PI-controller's output to *ASU5*'s K_La parameter.

After compiling and linking, a new experiment can be built with the new WEST Model Library. In addition to the parameters already specified in the openloop case, the parameters of the nitrate sensor

- delay time
- hold time
- standard deviation of the Gaussian white noise on the measurement
- minimum and maximum range of sensor output to avoid nonsensical negative nitrate concentrations due to the infinite noise tails

and the two PI-controllers

- DO and nitrate setpoints, *y_S*
- no error action, *u_0*
- proportional gain, *K_P*
- integrator time constant, *T_I*
- minimum and maximum control action, *u_min* and *u_max*

need to be specified. Note that for the DO sensor, no parameters need to be specified as the sensor is assumed to be ideal, reporting the actual dissolved oxygen concentration.

As the tight control of the DO and NO_3 controllers significantly affects the numerical properties of the model (the system becomes more stiff), it is necessary to change the accuracy of the *RK4ASC*

algorithm to 1.E-9 and to change the minimal step size to 0.0001 d. These changes are necessary to achieve sufficient simulation accuracy.

11.5 CONCLUSION

Setting up the '*simulation benchmark'* in WEST is relatively simple and the compiled code that is generated ensures high calculation performance. Users should be aware of several WEST-specific features including the pros and cons of using a variable step integrator. Use of the suggested integrator results in a deviation from the '*simulation benchmark'* definition, but the use of this integrator ensures that calculation times are minimised, with minimal variation in the output data communication interval and performance data. A *BenchmarkSensor* model was written for this study (and is now included in the standard WEST modelbase) and is used to perform standard benchmark data calculations. An important feature of this specific model is that the generated data can be used for the automatic tuning of user-defined controllers for optimal performance based on the benchmark criteria. However, it should be clear that users are not limited to the benchmark criteria only as the raw simulation data can still be exported from WEST as a standard text file for detailed data interpretation.

11.6 ACKNOWLEDGEMENTS

The work on the Benchmark in WEST is a joint effort by a team of Hemmis employees and researchers at the BIOMATH department of Ghent University (Belgium) that each contributed in different parts of its development. Especially Youri Amerlinck, Diedert Debusscher, Stefan De Grande, Matty Janssen, Jurgen Meirlaen, Dirk Stevens and Henk Vanhooren should be acknowledged for their contributions.

12

References

Alex J., Beteau J.F., Copp J.B., Hellinga C., Jeppsson U., Marsili-Libelli S., Pons M.N., Spanjers H. and Vanhooren H. (1999) Benchmark for evaluating control strategies in wastewater treatment plants. *Proceedings of the European Control Conference, ECC '99*, Karlsruhe, Germany, August 31-September 3, 1999.

Brenan, K.E., Campbell S.E. and Petzold L.R. (1989) Numerical solution of initial value problem in differential-algebraic equations. North-Holland, New-York. Code available at http://www.netlib.no/

Copp J. B. (2000) Defining a simulation benchmark for control strategies. *Water 21* (April), 44-49.

Copp J. B. (1999) Development of standardised influent files for the evaluation of activated sludge control strategies. *IAWQ Scientific and Technical Report Task Group: Respirometry in Control of the Activated Sludge Process – internal report.*

COST 682/624 Action website (http://www.ensic.u-nancy.fr/COSTWWTP) - *The European Co-operation in the field of Scientific and Technical Research.*

Dupont R. and Dahl (1995) A one-dimensional model for a secondary settling tank including density current and short-circuiting. *Wat. Sci. Technol.*, 31(2), pp.215-224

Henze M., Grady Jr C.P.L., Gujer W., Marais G.v.R. and Matsuo T. (1986) *Activated sludge model No.1, IAWQ Scientific and Technical Report No.1*, IAWQ, London.

Pons M. N., Spanjers H. and Jeppsson U. (1999) Towards a benchmark for evaluating control strategies in wastewater treatment plants by simulation. *Proceedings of 9th European Symposium on Computer Aided Process Engineering*, Budapest, Hungary, May 31-June 2, 1999.

Press W.H., Flannery S.A., Teukolsky S.A. and Vetterling W.T. (1992) Numerical recipes in FORTRAN. The art of scientific computing. Cambridge University Press, Cambridge

Spanjers H., Vanrolleghem P., Nguyen K. Vanhooren H. and Patry G.G. (1998) Towards a benchmark for evaluating control strategies in wastewater treatment plants by simulation. *Wat. Sci. Technol.*, 37(12), 219-226.

Spanjers H., Vanrolleghem P., Olsson G. and Dold P.L. (1998) *Respirometry in Control of the Activated Sludge Process: Principles, IAWQ Scientific and Technical Report No.7*, IAWQ, London.

Takács I., Patry G.G. and Nolasco D. (1991) A dynamic model of the clarification thickening process. *Wat. Res.*, 25, 10, 1263-1271.

Vanhooren H. and Nguyen K. (1996) Development of a simulation protocol for evaluation of respirometry-based control strategies, Technical Report, University of Gent, Gent, Belgium.

Vanrolleghem P., Jeppsson U., Carstensen J., Carlsson B. and Olsson G. (1996) Integration of wastewater treatment plant design and operation – a systematic approach to cost functions. *Wat. Sci. Technol.*, 34(3-4), 159-171.

13

Appendices

13.1 STEADY STATE RESULTS

Table 13.1: Steady state results generated by various simulation software packages.

PLATFORM	BioWin	EFOR	FORTRAN	GPS-X	MATLAB/ Simulink	SIMBA	STOAT	WEST
Anoxic tank #1								
Si	30	30	30	30	30	30	30	30
Ss	2.810	2.81	2.800	2.808	2.808	2.808	2.800	2.808
Xi	1149.110	1149.15	1150.000	1149.200	1149.125	1149.120	1598.100	1149.125
Xs	82.150	82.2	82.200	82.135	82.135	82.135	82.100	82.135
Xbh	2552.110	2553.49	2550.000	2551.800	2551.766	2551.760	2551.900	2551.766
Xba	148.340	148.37	148.000	148.390	148.389	148.389	148.400	148.389
Xp	448.980	449.16	449.000	448.860	448.852	448.850	(Xi = Xi + Xp)	448.852
So	0.000	0	0.004	0.004	0.004	0.004	-0.100	0.004
Sno	5.400	5.38	5.340	5.370	5.370	5.370	5.400	5.370
Snh	7.920	7.92	7.930	7.918	7.918	7.918	7.900	7.918
Snd	1.220	1.22	1.220	1.217	1.217	1.217	1.200	1.217
Xnd	5.350	5.29	5.290	5.285	5.285	5.285	5.300	5.285
Salk	4.930	4.93	4.930		4.928	4.928		
TSS	3289.830	3286.28	3284.400	3285.200	3285.200	3285.200	3283.697	3285.200
Anoxic tank #2								
Si	30	30	30	30	30	30	30	30
Ss	1.460	1.46	1.460	1.459	1.459	1.459	1.500	1.459
Xi	1149.110	1149.15	1150.000	1149.200	1149.125	1149.120	1598.800	1149.125
Xs	76.390	76.45	76.400	76.386	76.386	76.386	76.300	76.389
Xbh	2553.730	2555.12	2550.000	2553.400	2553.385	2553.380	2553.500	2553.385
Xba	148.260	148.29	148.000	148.310	148.309	148.309	148.300	148.309
Xp	449.660	450.06	450.000	449.530	449.523	449.521	(Xi = Xi + Xp)	449.523
So	0.000	0	0.000	0.000	0.000	0.000	0.000	0.000
Sno	3.690	3.67	3.640	3.662	3.662	3.662	3.700	3.662
Snh	8.350	8.34	8.350	8.344	8.344	8.344	8.300	8.344
Snd	0.880	0.88	0.883	0.882	0.882	0.882	0.900	0.882
Xnd	5.100	5.03	5.030	5.029	5.029	5.029	5.000	5.029
Salk	5.080	5.08	5.080		5.080	5.080		
TSS	3287.440	3283.62	3280.800	3282.500	3282.546	3282.540	3281.265	3282.548

Table 13.1 (cont'd): Steady state results generated by various simulation software packages.

PLATFORM	BioWin	EFOR	FORTRAN	GPS-X	MATLAB/ Simulink	SIMBA	STOAT	WEST
Aerobic tank #1								
Si	30	30	30	30	30	30	30	30
Ss	1.150	1.15	1.150	1.150	1.150	1.150	1.100	1.150
Xi	1149.110	1149.15	1150.000	1149.200	1149.125	1149.120	1599.700	1149.125
Xs	64.870	64.92	64.900	64.855	64.855	64.855	64.800	64.855
Xbh	2557.470	2558.87	2560.000	2557.100	2557.131	2557.130	2557.300	2557.131
Xba	148.890	148.92	149.000	148.940	148.941	148.941	149.000	148.941
Xp	450.550	0	451.000	450.430	450.418	450.416	(Xi = Xi + Xp)	450.418
So	1.710	1.72	1.720	1.718	1.718	1.718	1.700	1.718
Sno	6.560	6.55	6.510	6.541	6.541	6.541	6.600	6.541
Snh	5.560	5.54	5.560	5.548	5.548	5.548	5.500	5.548
Snd	0.830	0.83	0.830	0.829	0.829	0.829	0.800	0.829
Xnd	4.460	4.4	4.390	4.392	4.392	4.392	4.400	4.392
Salk	4.670	4.67	4.680		4.675	4.675		
TSS	3283.210	3278.93	3281.175	3277.800	3277.853	3277.850	3277.143	3277.853
Aerobic tank #2								
Si	30	30	30	30	30	30	30	30
Ss	1.000	1	0.993	0.995	0.995	0.995	1.000	0.995
Xi	1149.110	1149.15	1150.000	1149.200	1149.125	1149.120	1600.600	1149.125
Xs	55.720	55.76	55.700	55.694	55.694	55.694	55.700	55.694
Xbh	2559.520	2560.92	2560.000	2559.200	2559.186	2559.180	2559.300	2559.183
Xba	149.480	149.51	150.000	149.530	149.527	149.527	149.500	149.527
Xp	451.450	450.95	451.000	451.320	451.315	451.313	(Xi = Xi + Xp)	451.315
So	2.420	2.43	2.430	2.429	2.429	2.429	2.400	2.429
Sno	9.310	9.31	9.270	9.299	9.299	9.299	9.300	9.299
Snh	2.980	2.96	2.980	2.967	2.967	2.967	3.000	2.967
Snd	0.770	0.77	0.768	0.767	0.767	0.767	0.800	0.767
Xnd	3.950	3.88	3.880	3.879	3.879	3.879	3.900	3.879
Salk	4.290	4.29	4.300		4.294	4.293		
TSS	3279.410	3274.72	3275.025	3273.600	3273.633	3273.630	3273.292	3273.633
Aerobic tank #3								
Si	30	30	30	30	30	30	30	30
Ss	0.890	1	0.888	0.889	0.889	0.889	0.900	0.889
Xi	1149.110	1149.15	1150.000	1149.200	1149.125	1149.120	1601.500	1149.125
Xs	49.330	49.37	49.300	49.306	49.306	49.306	49.300	49.306
Xbh	2559.690	2561.09	2560.000	2559.400	2559.344	2559.340	2559.500	2559.344
Xba	149.750	149.78	150.000	149.800	149.797	149.797	149.800	149.797
Xp	452.340	451.85	452.000	452.220	452.211	452.209	(Xi = Xi + Xp)	452.211
So	0.500	0.49	0.490	0.491	0.491	0.491	0.500	0.491
Sno	10.450	10.42	10.400	10.415	10.415	10.415	10.400	10.415
Snh	1.740	1.73	1.750	1.733	1.733	1.733	1.700	1.733
Snd	0.690	0.69	0.689	0.688	0.688	0.688	0.700	0.688
Xnd	3.600	3.53	3.530	3.527	3.527	3.527	3.500	3.527
Salk	4.120	4.12	4.130		4.126	4.126		
TSS	3276.000	3270.92	3270.975	3269.800	3269.837	3269.830	3269.914	3269.837

Table 13.1 (cont'd): Steady state results generated by various simulation software packages.

PLATFORM	BioWin	EFOR	FORTRAN	GPS-X	MATLAB/ Simulink	SIMBA	STOAT	WEST
Effluent								
Si	30	30	30	30	30	30	30	30
Ss	0.890	0.89	0.888	0.889	0.889	0.889	0.900	0.889
Xi	4.270	4.39	4.39	4.390	4.392	4.392	6.100	4.392
Xs	0.210	0.19	0.189	0.188	0.188	0.188	0.140	0.188
Xbh	9.510	9.79	9.780	9.776	9.782	9.782	6.600	9.782
Xba	0.560	0.57	0.572	0.572	0.573	0.573	0.400	0.573
Xp	1.680	1.73	1.730	1.727	1.728	1.728	(Xi = Xi + Xp)	1.728
So	0.500	0.49	0.490	0.491	0.491	0.491	0.500	0.491
Sno	10.450	10.42	10.400	10.415	10.415	10.415	10.400	10.415
Snh	1.740	1.73	1.750	1.733	1.733	1.733	1.700	1.733
Snd	0.690	0.69	0.689	0.688	0.688	0.688	0.700	0.688
Xnd	0.010	0.01	0.014	0.013	0.013	0.013	0.000	0.013
Salk	4.120	4.12	4.130		4.126	4.126		
TSS	12.170	12.5	12.5	12.497	12.497	12.497	12.500	12.497
Settler interior (TSS) mg/l								
Layer 10	12.17	12.5	12.5	12.50	12.50	12.50	12.50	12.50
Layer 9	n/a	18.12	18.10	18.11	18.11	18.11	18.10	18.11
Layer 8	n/a	29.54	29.50	29.54	29.54	29.54	29.50	29.54
Layer 7	n/a	68.99	69.00	68.98	68.98	68.98	68.90	68.98
Layer 6	n/a	356.16	356.00	356.07	356.07	356.07	356.10	356.07
Layer 5	n/a	356.16	356.00	356.07	356.07	356.07	356.10	356.07
Layer 4	n/a	356.16	356.00	356.07	356.07	356.07	356.10	356.07
Layer 3	n/a	948.57	356.00	356.07	356.07	356.07	356.10	356.07
Layer 2	n/a	3275.91	356.00	356.07	356.07	356.07	356.10	356.07
Layer 1	6406.03	6396.11	6400.00	6393.90	6393.98	6393.98	6394.10	6393.98

13.2 DYNAMIC RESULTS

Table 13.2: Dynamic (openloop – i.e. without control) results generated with the *dry weather* file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Effluent average concentrations based on load								
Q =	m3/d	18061.33	18061.00	18061.33	18062.10	18078.75	18064.90	17.75
S_I =	g COD/m3	30.0000	30.0000	30.0000	30.0000	30.0000	30.0000	0.0000
S_S =	g COD/m3	0.9694	0.9730	0.9736	0.9735	0.9738	0.9727	0.0044
X_I =	g COD/m3	4.5878	4.5800	4.5779	4.5795	4.5865	4.5823	0.0099
X_S =	g COD/m3	0.2250	0.2290	0.2229	0.2229	0.2229	0.2245	0.0061
X_BH =	g COD/m3	10.2219	10.2000	10.2206	10.2209	10.2225	10.2172	0.0225
X_BA =	g COD/m3	0.5412	0.5420	0.5420	0.5422	0.5421	0.5419	0.0009
X_P =	g COD/m3	1.7580	1.7600	1.7560	1.7572	1.7570	1.7576	0.0040
S_O =	g (-COD)/m3	0.7978	n/a	0.7463	0.7463	0.7462	0.7592	0.0516
S_NO =	g N/m3	8.8464	8.8000	8.8231	8.8237	8.8182	8.8223	0.0464
S_NH =	g N/m3	4.8571	4.8000	4.7632	4.7590	4.7746	4.7908	0.0981
S_ND =	g N/m3	0.7260	0.7310	0.7291	0.7290	0.7292	0.7289	0.0050
X_ND =	g N/m3	0.0158	0.0157	0.0157	0.0157	0.0157	0.0157	0.0001
S_ALK =	kmol HCO3/m3	n/a	4.4600	4.4565	4.4562	n/a	4.4576	0.0038
TSS =	g SS/m3	13.0004	12.9780	12.9895	12.9919	12.9982	12.9916	0.0224
N_TKN =	g N/m3	6.8407	6.7900	6.7490	6.7450	6.7613	6.7772	0.0957
N_tot =	g N/m3	15.6871	15.6000	15.5721	15.5687	15.5796	15.6015	0.1184
COD_tot =	g COD/m3	48.3033	46.5000	48.2930	48.2961	48.3048	47.9394	1.8048
BOD5_tot =	g/m3	2.7741	2.7800	2.7745	2.7746	2.7750	2.7757	0.0059
Effluent average loads								
S_I =	kg COD/d	541.8400	542.0000	541.8400	541.8620	542.3625	541.9809	0.5226
S_S =	kg COD/d	17.5094	17.6000	17.5848	17.5837	17.6057	17.5767	0.0963
X_I =	kg COD/d	82.8610	82.7000	82.6833	82.7153	82.9177	82.7755	0.2344
X_S =	kg COD/d	4.0641	4.0300	4.0255	4.0253	4.0290	4.0348	0.0388
X_BH =	kg COD/d	184.6214	185.0000	184.5980	184.6110	184.8094	184.7280	0.4020
X_BA =	kg COD/d	9.7752	9.7900	9.7891	9.7928	9.8014	9.7897	0.0261
X_P =	kg COD/d	31.7515	31.7000	31.7151	31.7379	31.7642	31.7337	0.0642
S_O =	kg (-COD)/d	14.4093	n/a	13.4791	13.4803	13.4901	13.7147	0.9302
S_NO =	kg N/d	159.7777	159.0000	159.3566	159.3740	159.4225	159.3862	0.7777
S_NH =	kg N/d	87.7261	86.6000	86.0291	85.9579	86.3197	86.5266	1.7682
S_ND =	kg N/d	13.1116	13.2000	13.1682	13.1673	13.1834	13.1661	0.0884
X_ND =	kg N/d	0.2859	0.2840	0.2834	0.2834	0.2837	0.2841	0.0025
S_ALK =	kmol HCO3/d	n/a	80.5000	80.4910	80.4882	n/a	80.4931	0.0118
TSS =	kg N/d	234.8047	234.9200	234.6081	234.6610	234.9912	234.7970	0.3831
N_TKN =	kg N/d	123.5521	122.6300	121.8956	121.8280	122.2366	122.4285	1.7241
N_tot =	kg N/d	283.3298	281.7500	281.2522	281.2020	281.6592	281.8386	2.1278
COD_tot =	kg COD/d	872.4226	840.0000	872.2357	872.3270	873.2898	866.0550	33.2898
BOD5_tot =	kg/d	50.1046	50.2100	50.1116	50.1150	50.1691	50.1421	0.1054

Table 13.2 (cont'd): Dynamic (openloop – i.e. without control) results generated with the *dry weather* file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Performance Indices								
I.Q.-index =	kg poll.units/d	42042.81	42000.00	42042.81	42042.81	42042.82	42034.25	42.82
E.Q.-index =	kg poll.units/d	7108.84	7040.00	7066.72	7065.17	7066.68	7069.48	68.84
P_sludge =	kg SS	17071.10	17052.00	17051.79	17050.58	17056.26	17056.35	20.52
P_sludge per day =	kg SS/d	2438.70	2436.00	2435.97	2435.80	2436.61	2436.62	2.90
P_sludge_eff =	kg SS	1643.60	1631.00	1642.26	1642.63	1644.94	1640.89	13.94
P_sludge_eff per day =	kg SS/d	234.80	233.00	234.61	234.66	234.99	234.41	1.99
P_total_sludge =	kg SS	18714.80	18683.00	18694.05	18693.21	18701.20	18697.25	31.80
P_total_sludge per day =	kg SS/d	2673.50	2669.00	2670.58	2670.46	2671.60	2671.03	4.50
Aeration energy =	kWh/d	6476.00	6476.00	6476.11	6476.11	6476.11	6476.07	0.11
Pumping energy =	kWh/d	2967.00	2967.00	2966.76	2966.76	2966.76	2966.86	0.24
The max effluent N_tot level (18 g N/m3)								
was violated during:	d	0.06	0.58	0.57	0.57	0.58	0.47	0.52
i.e.:	% of the time	0.89	8.23	8.18	8.18	8.25	6.75	7.36
The limit was violated at:	occasions	1	5	5	5	5	4.20	4
The max effluent S_NH level (4 g N/m3)								
was violated during:	d	4.03	4.40	4.38	4.38	4.35	4.31	0.37
i.e.:	% of the time	57.60	62.90	62.50	62.50	62.21	61.54	5.30
The limit was violated at:	occasions	5	7	7	7	7	6.60	2

Table 13.3: Dynamic (openloop – i.e. without control) results generated with the *rain* weather file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Effluent average concentrations based on load								
Q =	m3/d	23808.18	23810.00	23808.18	23808.60	23830.01	23812.99	21.83
S_I =	g COD/m3	22.8329	22.8000	22.8388	22.8388	22.8460	22.8313	0.0460
S_S =	g COD/m3	1.1367	1.1300	1.1345	1.1343	1.1345	1.1340	0.0067
X_I =	g COD/m3	5.6458	5.6500	5.6372	5.6389	5.6460	5.6436	0.0128
X_S =	g COD/m3	0.3493	0.3460	0.3448	0.3447	0.3448	0.3459	0.0046
X_BH =	g COD/m3	12.8500	12.9000	12.8567	12.8565	12.8568	12.8640	0.0500
X_BA =	g COD/m3	0.6409	0.6430	0.6426	0.6428	0.6427	0.6424	0.0021
X_P =	g COD/m3	2.0681	2.0700	2.0666	2.0680	2.0675	2.0680	0.0034
S_O =	g (-COD)/m3	0.8860	n/a	0.8472	0.8470	0.8468	0.8567	0.0392
S_NO =	g N/m3	6.9683	6.9300	6.9585	6.9596	6.9581	6.9549	0.0383
S_NH =	g N/m3	5.1016	5.0100	4.9862	4.9820	4.9935	5.0147	0.1196
S_ND =	g N/m3	0.8151	0.8180	0.8157	0.8156	0.8157	0.8160	0.0029
X_ND =	g N/m3	0.0239	0.0237	0.0236	0.0236	0.0236	0.0237	0.0003
S_ALK =	kmol HCO3/m3	n/a	5.1500	5.1435	5.1431	n/a	5.1455	0.0069
TSS =	g SS/m3	16.1656	16.2100	16.1610	16.1633	16.1683	16.1736	0.0490
N_TKN =	g N/m3	7.4827	7.4000	7.3677	7.3636	7.3756	7.3979	0.1191
N_tot =	g N/m3	14.4510	14.3000	14.3262	14.3231	14.3337	14.3468	0.1510
COD_tot =	g COD/m3	45.5236	43.4000	45.5213	45.5242	45.5382	45.1015	2.1382
BOD5_tot =	g/m3	3.4744	3.4800	3.4747	3.4746	3.4747	3.4757	0.0056
Effluent average loads								
S_I =	kg COD/d	543.6091	544.0000	543.7504	543.7600	544.4198	543.9079	0.8108
S_S =	kg COD/d	27.0626	27.0000	27.0100	27.0066	27.0350	27.0228	0.0626
X_I =	kg COD/d	134.4152	135.0000	134.2121	134.2550	134.5443	134.4853	0.7879
X_S =	kg COD/d	8.3166	8.2400	8.2094	8.2079	8.2163	8.2380	0.1087
X_BH =	kg COD/d	305.9359	307.0000	306.0952	306.0950	306.3766	306.3006	1.0641
X_BA =	kg COD/d	15.2575	15.3000	15.2998	15.3051	15.3149	15.2955	0.0574
X_P =	kg COD/d	49.2380	49.3000	49.2025	49.2359	49.2685	49.2490	0.0975
S_O =	kg (-COD)/d	21.0932	n/a	20.1699	20.1668	20.1784	20.4021	0.9264
S_NO =	kg N/d	165.9032	165.0000	165.6694	165.6970	165.8120	165.6163	0.9032
S_NH =	kg N/d	121.4604	120.0000	118.7122	118.6150	118.9960	119.5567	2.8454
S_ND =	kg N/d	19.4049	19.5000	19.4204	19.4181	19.4383	19.4363	0.0951
X_ND =	kg N/d	0.5686	0.5640	0.5618	0.5618	0.5624	0.5637	0.0069
S_ALK =	kmol HCO3/d	n/a	123.0000	122.4574	122.4510	n/a	122.6361	0.5490
TSS =	kg N/d	384.8726	386.1300	384.7643	384.8240	385.2905	385.1763	1.3657
N_TKN =	kg N/d	178.1486	176.2000	175.4109	175.3160	175.7607	176.1673	2.8326
N_tot =	kg N/d	344.0518	340.4800	341.0803	341.0130	341.5727	341.6396	3.5718
COD_tot =	kg COD/d	1083.8349	1033.3000	1083.7794	1083.8700	1085.1755	1073.9920	51.8755
BOD5_tot =	kg/d	82.7193	82.8600	82.7257	82.7257	82.8019	82.7665	0.1407

Table 13.3 (cont'd): Dynamic (openloop – i.e. without control) results generated with the *rain* weather file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Performance Indices								
I.Q.-index =	kg poll.units/d	42042.81	42000.00	42042.81	42042.81	42042.82	42034.2501	42.8157
E.Q.-index =	kg poll.units/d	8900.06	8800.00	8840.37	8838.60	8843.96	8844.5975	100.0600
P_sludge =	kg SS	16492.60	16457.00	16469.13	16467.86	16536.97	16484.7113	79.9685
P_sludge per day =	kg SS/d	2356.10	2351.00	2352.73	2352.55	2362.42	2354.9613	11.4241
P_sludge_eff =	kg SS	2694.10	2688.00	2693.35	2693.77	2697.03	2693.2506	9.0332
P_sludge_eff per day =	kg SS/d	384.90	384.00	384.76	384.82	385.29	384.7550	1.2905
P_total_sludge =	kg SS	19186.70	19145.00	19162.48	19161.63	19234.00	19177.9619	89.0017
P_total_sludge per day =	kg SS/d	2741.00	2735.00	2737.50	2737.38	2747.71	2739.7183	12.7145
Aeration energy =	kWh/d	6476.00	6476.00	6476.11	6476.11	6476.11	6476.0668	0.1120
Pumping energy =	kWh/d	2967.00	2967.00	2966.76	2966.76	2966.76	2966.8560	0.2400
The max effluent N_tot level (18 g N/m3)								
was violated during:	d	0.00	0.31	0.31	0.30	0.32	0.2487	0.3150
i.e.:	% of the time	0.00	4.49	4.46	4.32	4.50	3.5549	4.5000
The limit was violated at:	occasions	0	3	3	3	3	2.4000	3
The max effluent S_NH level (4 g N/m3)								
was violated during:	d	4.38	4.46	4.44	4.43	4.45	4.4317	0.0780
i.e.:	% of the time	62.60	63.80	63.39	63.24	63.60	63.3266	1.2000
The limit was violated at:	occasions	7	7	7	7	7	7.0000	0

Table 13.4: Dynamic (openloop – i.e. without control) results generated with the *storm* weather file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Effluent average concentrations based on load								
Q =	m3/d	20658.10	20658.00	20658.10	20658.20	20677.34	20661.95	19.34
S_I =	g COD/m3	26.2778	26.3000	26.2999	26.3000	26.3015	26.2958	0.0237
S_S =	g COD/m3	1.1054	1.1100	1.1131	1.1129	1.1130	1.1109	0.0077
X_I =	g COD/m3	5.6398	5.6400	5.6355	5.6370	5.6430	5.6390	0.0075
X_S =	g COD/m3	0.3266	0.3240	0.3227	0.3226	0.3224	0.3236	0.0042
X_BH =	g COD/m3	11.8591	11.9000	11.8802	11.8801	11.8712	11.8781	0.0409
X_BA =	g COD/m3	0.5860	0.5890	0.5883	0.5884	0.5879	0.5879	0.0030
X_P =	g COD/m3	1.9113	1.9200	1.9125	1.9137	1.9119	1.9139	0.0087
S_O =	g (-COD)/m3	0.7990	n/a	0.7635	0.7634	0.7632	0.7723	0.0358
S_NO =	g N/m3	7.5002	7.4500	7.4800	7.4809	7.4791	7.4780	0.0502
S_NH =	g N/m3	5.4755	5.3900	5.3539	5.3495	5.3610	5.3860	0.1260
S_ND =	g N/m3	0.7981	0.8060	0.8035	0.8034	0.8036	0.8029	0.0079
X_ND =	g N/m3	0.0229	0.0227	0.0226	0.0226	0.0226	0.0227	0.0003
S_ALK =	kmol HCO3/m3	n/a	4.8800	4.8726	4.8722	n/a	4.8749	0.0078
TSS =	g SS/m3	15.2421	15.2800	15.2543	15.2563	15.2522	15.2570	0.0379
N_TKN =	g N/m3	7.7452	7.6690	7.6305	7.6261	7.6373	7.6616	0.1191
N_tot =	g N/m3	15.2454	15.1200	15.1105	15.1070	15.1164	15.1399	0.1384
COD_tot =	g COD/m3	47.7060	45.8400	47.7520	47.7546	47.7507	47.3607	1.9146
BOD5_tot =	g/m3	3.2204	3.2300	3.2267	3.2266	3.2244	3.2256	0.0096
Effluent average loads								
S_I =	kg COD/d	542.8501	543.0000	543.3052	543.3110	543.8448	543.2622	0.9947
S_S =	kg COD/d	22.8351	23.0000	22.9939	22.9914	23.0141	22.9669	0.1790
X_I =	kg COD/d	116.5072	117.0000	116.4188	116.4500	116.6821	116.6116	0.5812
X_S =	kg COD/d	6.7463	6.6900	6.6654	6.6643	6.6660	6.6864	0.0820
X_BH =	kg COD/d	244.9865	246.0000	245.4223	245.4210	245.4639	245.4587	1.0135
X_BA =	kg COD/d	12.1054	12.2000	12.1522	12.1561	12.1555	12.1539	0.0946
X_P =	kg COD/d	39.4846	39.6000	39.5078	39.5333	39.5320	39.5315	0.1154
S_O =	kg (-COD)/d	16.5062	n/a	15.7727	15.7713	15.7818	15.9580	0.7349
S_NO =	kg N/d	154.9403	154.0000	154.5230	154.5420	154.6487	154.5308	0.9403
S_NH =	kg N/d	113.1139	111.0000	110.6024	110.5120	110.8515	111.2160	2.6019
S_ND =	kg N/d	16.4868	16.7000	16.5990	16.5972	16.6160	16.5998	0.2132
X_ND =	kg N/d	0.4736	0.4690	0.4679	0.4678	0.4681	0.4693	0.0058
S_ALK =	kmol HCO3/d	n/a	101.0000	100.6591	100.6520	n/a	100.7704	0.3480
TSS =	kg N/d	314.8730	316.1200	315.1249	315.1680	315.3747	315.3321	1.2470
N_TKN =	kg N/d	160.0012	158.4300	157.6308	157.5420	157.9181	158.3044	2.4592
N_tot =	kg N/d	314.9415	312.3500	312.1538	312.0840	312.5668	312.8192	2.8575
COD_tot =	kg COD/d	985.5151	946.9600	986.4656	986.5260	987.3584	978.5650	40.3984
BOD5_tot =	kg/d	66.5265	66.7300	66.6570	66.6566	66.6725	66.6485	0.2035

Table 13.4 (cont'd): Dynamic (openloop – i.e. without control) results generated with the *storm* weather file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Performance Indices								
I.Q.-index =	kg poll.units/d	43758.11	43800.00	43758.11	43758.11	43758.12	43766.4902	41.8900
E.Q.-index =	kg poll.units/d	8047.14	7960.00	7993.11	7991.19	7994.79	7997.2443	87.1400
P_sludge =	kg SS	18223.60	18200.00	18197.38	18196.14	18188.33	18201.0895	35.2732
P_sludge per day =	kg SS/d	2603.40	2600.00	2599.63	2599.45	2598.33	2600.1616	5.0676
P_sludge_eff =	kg SS	2204.10	2191.00	2205.87	2206.18	2207.62	2202.9553	16.6226
P_sludge_eff per day =	kg SS/d	314.90	313.00	315.12	315.17	315.37	314.7139	2.3747
P_total_sludge =	kg SS	20427.70	20391.00	20403.25	20402.31	20395.95	20404.0428	36.7000
P_total_sludge per day =	kg SS/d	2918.20	2913.00	2914.75	2914.62	2913.71	2914.8556	5.2000
Aeration energy =	kWh/d	6476.00	6476.00	6476.11	6476.11	6476.11	6476.0668	0.1120
Pumping energy =	kWh/d	2967.00	2967.00	2966.76	2966.76	2966.76	2966.8560	0.2400
The max effluent N_tot level (18 g N/m3)								
was violated during:	d	0.17	0.60	0.59	0.59	0.58	0.5057	0.4304
i.e.:	% of the time	2.38	8.53	8.48	8.48	8.25	7.2244	6.1500
The limit was violated at:	occasions	3	4	4	4	4	3.8000	1
The max effluent S_NH level (4 g N/m3)								
was violated during:	d	4.47	4.55	4.51	4.51	4.52	4.5104	0.0840
i.e.:	% of the time	63.80	65.00	64.43	64.43	64.50	64.4329	1.2000
The limit was violated at:	occasions	6	7	7	7	7	6.8000	1
The max effluent TSS level (30 g SS/m3)								
was violated during:	d	0.00	0.01	0.01	0.01	0.01	0.0084	0.0105
i.e.:	% of the time	0.00	0.15	0.15	0.15	0.15	0.1198	0.1500
The limit was violated at:	occasions	0	1	1	1	1	0.8000	1

13.3 BASIC CONTROL STRATEGY RESULTS

Table 13.5: Dynamic (closedloop – i.e. basic control strategy operational) results generated with the *dry weather* file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Effluent average concentrations based on load								
Q =	m3/d	18061.33	18061.00	18061.26	18085.60	18064.24	18066.69	24.60
S_I =	g COD/m3	30.0000	30.0000	30.0000	30.0000	30.0000	30.0000	0.0000
S_S =	g COD/m3	0.8726	0.8880	0.8817	0.8852	0.8779	0.8811	0.0154
X_I =	g COD/m3	4.5812	4.5700	4.5718	4.5735	4.5725	4.5738	0.0112
X_S =	g COD/m3	0.2018	0.2020	0.2007	0.2017	0.1993	0.2011	0.0027
X_BH =	g COD/m3	10.2315	10.2000	10.2308	10.2303	10.2110	10.2207	0.0315
X_BA =	g COD/m3	0.5788	0.5800	0.5783	0.5723	0.5781	0.5775	0.0077
X_P =	g COD/m3	1.7569	1.7600	1.7548	1.7556	1.7545	1.7564	0.0055
S_O =	g (-COD)/m3	1.9902	n/a	1.9997	1.9867	1.9989	1.9939	0.0130
S_NO =	g N/m3	12.4125	13.2000	12.4394	12.6486	12.3351	12.6071	0.8649
S_NH =	g N/m3	2.5285	2.4500	2.5287	2.5893	2.5020	2.5197	0.1393
S_ND =	g N/m3	0.7012	0.7130	0.7066	0.7083	0.7052	0.7069	0.0118
X_ND =	g N/m3	0.0145	0.0146	0.0144	0.0145	0.0143	0.0145	0.0003
S_ALK =	kmol HCO3/m3	n/a	3.9800	4.0387	4.0287	n/a	4.0158	0.0587
TSS =	g SS/m3	13.0126	12.9800	13.0023	13.0000	12.9866	12.9963	0.0326
N_TKN =	g N/m3	4.4892	4.4200	4.4940	4.5561	4.4643	4.4847	0.1361
N_tot =	g N/m3	16.9017	17.6000	16.9334	17.2047	16.7994	17.0878	0.8006
COD_tot =	g COD/m3	48.2227	48.2400	48.2181	48.2185	48.1934	48.2185	0.0466
BOD5_tot =	g/m3	2.7550	2.7600	2.7567	2.7563	2.7508	2.7558	0.0092
Effluent average loads								
S_I =	kg COD/d	541.8400	542.0000	541.8378	542.5690	541.9273	542.0348	0.7312
S_S =	kg COD/d	15.7598	16.0000	17.9247	16.0094	15.8590	16.3106	2.1649
X_I =	kg COD/d	82.7417	82.6000	82.5732	82.7148	82.5996	82.6459	0.1685
X_S =	kg COD/d	3.6450	3.6500	3.6252	3.6477	3.5999	3.6335	0.0501
X_BH =	kg COD/d	184.7941	185.0000	184.7804	185.0220	184.4535	184.8100	0.5685
X_BA =	kg COD/d	10.4537	10.5000	10.4439	10.3498	10.4433	10.4381	0.1502
X_P =	kg COD/d	31.7321	31.7000	31.6946	31.7503	31.6940	31.7142	0.0563
S_O =	kg (-COD)/d	35.9453	n/a	36.1178	35.9303	36.1080	36.0253	0.1875
S_NO =	kg N/d	224.1856	238.0000	224.6717	228.7580	222.8250	227.6881	15.1750
S_NH =	kg N/d	45.6680	44.2000	45.6707	46.8300	45.1966	45.5131	2.6300
S_ND =	kg N/d	12.6638	12.9000	12.7621	12.8100	12.7396	12.7751	0.2362
X_ND =	kg N/d	0.2614	0.2630	0.2603	0.2620	0.2585	0.2611	0.0045
S_ALK =	kmol HCO3/d	n/a	71.9000	72.9446	72.8614	n/a	72.5687	1.0446
TSS =	kg N/d	235.0254	234.5000	234.8380	235.1140	234.5927	234.8140	0.6140
N_TKN =	kg N/d	81.0816	79.8300	81.1672	82.3997	80.6441	81.0245	2.5697
N_tot =	kg N/d	305.2672	317.9000	305.8388	311.1570	303.4691	308.7264	14.4309
COD_tot =	kg COD/d	870.9663	871.2000	870.8798	872.0640	870.5766	871.1373	1.4874
BOD5_tot =	kg/d	49.7582	49.8500	49.7891	49.8498	49.6910	49.7876	0.1590

Table 13.5 (cont'd): Dynamic (closedloop – i.e. basic control strategy operational) results generated with the *dry weather* file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Performance Indices								
I.Q.-index =	kg poll.units/d	42042.81	42000.00	42042.81	42042.81	42042.82	42034.25	42.82
E.Q.-index =	kg poll.units/d	7545.88	7760.00	7556.91	7665.14	7501.02	7605.79	258.98
P_sludge =	kg SS	17104.30	17087.00	17085.46	17080.74	17142.68	17100.03	61.94
P_sludge per day =	kg SS/d	2443.50	2441.00	2440.78	2440.11	2448.95	2442.87	8.84
P_sludge_eff =	kg SS	1645.20	1631.00	1643.87	1645.80	1642.15	1641.60	14.80
P_sludge_eff per day =	kg SS/d	235.00	233.00	234.84	235.11	234.59	234.51	2.11
P_total_sludge =	kg SS	18749.50	18718.00	18729.32	18726.54	18784.82	18741.64	66.82
P_total_sludge per day =	kg SS/d	2678.50	2674.00	2675.62	2675.22	2683.55	2677.38	9.55
Aeration energy =	kWh/d	7241.00	7262.00	7241.27	7253.21	7231.50	7245.80	30.50
Pumping energy =	kWh/d	1524.00	1328.00	1488.14	1430.31	1521.88	1458.47	196.00
The max effluent N_tot level (18 g N/m3)								
was violated during:	d	0.99	2.03	1.28	1.46	1.12	1.38	1.04
i.e.:	% of the time	14.14	29.00	18.30	20.83	16.05	19.66	14.86
The limit was violated at:	occasions	5	9	7	7	7	7.00	4
The max effluent S_NH level (4 g N/m3)								
was violated during:	d	1.21	1.16	1.21	1.30	1.20	1.22	0.14
i.e.:	% of the time	17.30	16.60	17.26	18.60	17.10	17.37	2.00
The limit was violated at:	occasions	5	5	5	6	5	5.20	1
Nitrate controller for second anoxic reactor								
Controller type		velocity PI	disc PI with aw	cont PI with aw	cont PI	PI		
Proportional gain (K)	m3/d/(g N/m3)	7500	5040	15000	10000	10000	9508.00	9960.00
Integral time constant (Ti)	d	0.0125	0.007	0.05	0.08	0.01	0.03	0.07
Anti-windup time constant (Tt)	d	not used	not used	0.03	not used	not used		
Controlled variable, SNO2								
Setpoint	g N/m3	1	1	1	1	1	1.00	0.00
Integral of absolute error (IAE)	(g N/m3)*d	0.1851641	4.04	1.4818	2.57325	0.8287897	1.82	3.85
Integral of square error (ISE)	(g N/m3)**2*d	0.0662959	3.35	0.59844	1.61783	0.1892283	1.16	3.28
Max dev from setpoint (max e)	g N/m3	0.8827	1.59	0.88729	0.9	0.6515534	0.98	0.94
Standard deviation of error (std e)	g N/m3	0.1790472	0.38	0.29234	0.480315	0.1644458	0.30	0.32
Variance of error (var e)	(g N/m3)**2	0.0320579	0.144	0.085463	0.230702	0.0270424	0.10	0.20
Manipulated variable (MV), Qintrec								
Max deviation of MV (max-min)	m3/d	49531	92232	36691.4502	30887.5	46724.644	51213.32	61344.50
Max dev in MV (delta)	m3/d	10677	12000	8077.7893	6218.02	9880.633	9370.69	5781.98
Std deviation of MV (delta)	m3/d	1622.53	285.6	1661.9725	1741.16	1554.0036	1373.05	1455.56
Variance of MV (delta)	(m3/d)**2	2632603.5	81567	2762152	3031000	2414927.1	2184450	2949433

Table 13.5 (cont'd): Dynamic (closedloop – i.e. basic control strategy operational) results generated with the *dry weather* file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Oxygen controller for third aerobic reactor								
Controller type		velocity PI	disc PI with aw	cont PI with aw	cont PI	PI		
Proportional gain (K)	1/d/(g (-COD)/m3)	20.8	16.8	500	100	5000	1127.52	4983.20
Integral time constant (Ti)	d	0.002	0.0025	0.001	0.01	0.05	0.01	0.05
Anti-windup time constant (Tt)	d	not used	not used	0.0002	not used	not used	0.00	0.00
Controlled variable, SO5								
Setpoint	g (-COD)/m3	2	2	2	2	2	2.00	0.00
Integral of absolute error (IAE)	(g (-COD)/m3)*d	0.0451658	0.454	0.0075065	0.302823	0.0266098	0.17	0.45
Integral of square error (ISE)	(g (-COD)/m3)**2*d	0.0049911	0.067	1.74E-05	0.029381	0.000191	0.02	0.07
Max dev from setpoint (max e)	g (-COD)/m3	0.3408	0.41	0.0069017	0.259716	0.0175857	0.21	0.40
Standard deviation of error (std e)	g (-COD)/m3	0.0543653	0.0731	0.0015756	0.064787	0.005226	0.04	0.07
Variance of error (var e)	(g (-COD)/m3)**2	0.0029556	0.0053	2.4824E-06	0.004197	2.731E-05	0.00	0.01
Manipulated variable (MV), Kla5								
Max deviation of MV (max-min)	1/d	186.737	240	186.4117	186.99	182.00036	196.43	58.00
Max dev in MV (delta)	1/d	36.229	12	36.7723	29.7157	38.65462	30.67	26.65
Std deviation of MV (delta)	1/d	5.9096592	0.41	5.7745	6.92972	5.9642918	5.00	6.52
Variance of MV (delta)	(1/d)**2	34.924072	168	33.3452	48.021	35.572777	63.97	134.65

Table 13.6: Dynamic (closedloop – i.e. basic control strategy operational) results generated with the *rain* weather file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Effluent average concentrations based on load								
Q =	m3/d	23808.18	23810.00	23808.12	23833.10	23830.46	23817.97	24.98
S_I =	g COD/m3	22.8323	22.8000	22.8387	22.8450	22.8456	22.8323	0.0456
S_S =	g COD/m3	1.0238	1.0500	1.0296	1.0372	1.0241	1.0329	0.0262
X_I =	g COD/m3	5.6352	5.6300	5.6271	5.6277	5.6239	5.6288	0.0113
X_S =	g COD/m3	0.3127	0.3190	0.3110	0.3138	0.3082	0.3130	0.0108
X_BH =	g COD/m3	12.8760	12.9000	12.8810	12.8758	12.8497	12.8765	0.0503
X_BA =	g COD/m3	0.6854	0.6890	0.6856	0.6777	0.6851	0.6846	0.0114
X_P =	g COD/m3	2.0624	2.0600	2.0610	2.0611	2.0597	2.0608	0.0027
S_O =	g (-COD)/m3	1.9932	n/a	1.9998	1.9910	1.9992	1.9958	0.0088
S_NO =	g N/m3	9.1307	9.7400	9.1748	9.3130	9.0782	9.2873	0.6618
S_NH =	g N/m3	3.2586	3.1600	3.2172	3.2872	3.1952	3.2236	0.1272
S_ND =	g N/m3	0.7842	0.8000	0.7875	0.7907	0.7865	0.7898	0.0158
X_ND =	g N/m3	0.0216	0.0221	0.0215	0.0217	0.0213	0.0216	0.0008
S_ALK =	kmol HCO3/m3	n/a	4.8200	4.8589	4.8540	n/a	4.8443	0.0389
TSS =	g SS/m3	16.1788	16.2000	16.1744	16.1671	16.1450	16.1731	0.0550
N_TKN =	g N/m3	5.6112	5.5300	5.5729	5.6453	5.5468	5.5812	0.1153
N_tot =	g N/m3	14.7420	15.2700	14.7477	14.9582	14.6250	14.8686	0.6450
COD_tot =	g COD/m3	45.4279	45.4400	45.4341	45.4383	45.3963	45.4273	0.0437
BOD5_tot =	g/m3	3.4533	3.4650	3.4555	3.4551	3.4461	3.4550	0.0189
Effluent average loads								
S_I =	kg COD/d	543.5964	544.0000	543.7459	544.4670	544.4212	544.0461	0.8706
S_S =	kg COD/d	24.3751	25.0000	24.5120	24.7192	24.4038	24.6020	0.6249
X_I =	kg COD/d	134.1647	134.0000	133.9717	134.1250	134.0204	134.0564	0.1930
X_S =	kg COD/d	7.4449	7.6000	7.4050	7.4798	7.3455	7.4550	0.2545
X_BH =	kg COD/d	306.5551	307.0000	306.6719	306.8700	306.2145	306.6623	0.7855
X_BA =	kg COD/d	16.3182	16.4000	16.3236	16.1505	16.3265	16.3038	0.2495
X_P =	kg COD/d	49.1021	49.2000	49.0695	49.1233	49.0827	49.1155	0.1305
S_O =	kg (-COD)/d	47.4548	n/a	47.6122	47.4520	47.6427	47.5404	0.1907
S_NO =	kg N/d	217.3860	232.0000	218.4344	221.9580	216.3371	221.2231	15.6629
S_NH =	kg N/d	77.5822	75.3000	76.5961	78.3443	76.1428	76.7931	3.0443
S_ND =	kg N/d	18.6708	19.1000	18.7496	18.8455	18.7430	18.8218	0.4292
X_ND =	kg N/d	0.5143	0.5270	0.5122	0.5176	0.5079	0.5158	0.0191
S_ALK =	kmol HCO3/d	n/a	115.0000	118.6819	115.6870	n/a	116.4563	3.6819
TSS =	kg N/d	385.1882	385.6500	385.0813	385.3130	384.7422	385.1949	0.9078
N_TKN =	kg N/d	133.5932	131.6700	132.6800	134.5440	132.1832	132.9341	2.8740
N_tot =	kg N/d	350.9792	364.5800	351.1144	356.5020	348.5203	354.3392	16.0597
COD_tot =	kg COD/d	1081.5565	1082.1000	1081.6997	1082.940	1081.8146	1082.0222	1.3835
BOD5_tot =	kg/d	82.2158	82.5010	82.2682	82.3446	82.1217	82.2903	0.3793

Table 13.6 (cont'd): Dynamic (closedloop – i.e. basic control strategy operational) results generated with the *rain* weather file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Performance Indices								
I.Q.-index =	kg poll.units/d	42042.81	42000.00	42042.81	42042.81	42042.82	42034.25	42.82
E.Q.-index =	kg poll.units/d	9035.95	9250.00	9038.69	9148.28	8976.96	9089.98	273.04
P_sludge =	kg SS	16518.30	16492.00	16504.47	16499.89	16629.17	16528.77	137.17
P_sludge per day =	kg SS/d	2359.80	2356.00	2357.78	2357.13	2375.60	2361.26	19.60
P_sludge_eff =	kg SS	2696.30	2695.00	2695.57	2697.18	2693.20	2695.45	3.98
P_sludge_eff per day =	kg SS/d	385.20	385.00	385.08	385.31	384.74	385.07	0.57
P_total_sludge =	kg SS	19214.60	19187.00	19200.04	19197.07	19322.37	19224.22	135.37
P_total_sludge per day =	kg SS/d	2744.90	2741.00	2742.86	2742.44	2760.34	2746.31	19.34
Aeration energy =	kWh/d	7168.00	7198.00	7169.77	7179.71	7159.81	7175.06	38.19
Pumping energy =	kWh/d	2009.00	1647.00	1927.53	1840.03	2004.24	1885.56	362.00
The max effluent N_tot level (18 g N/m3)								
was violated during:	d	0.42	1.16	0.79	0.91	0.68	0.79	0.74
i.e.:	% of the time	5.95	16.60	11.31	12.95	9.75	11.31	10.65
The limit was violated at:	occasions	3	7	5	5	5	5.00	4
The max effluent S_NH level (4 g N/m3)								
was violated during:	d	2.00	1.86	1.90	1.96	1.84	1.91	0.16
i.e.:	% of the time	28.60	26.60	27.08	27.98	26.25	27.30	2.35
The limit was violated at:	occasions	8	8	8	8	8	8.00	0
Nitrate controller for second anoxic reactor								
Controller type		velocity PI	disc PI with aw	cont PI with aw	cont PI	PI		
Proportional gain (K)	m3/d/(g N/m3)	7500	5040	15000	10000	10000	9508.00	9960.00
Integral time constant (Ti)	d	0.0125	0.007	0.05	0.08	0.01	0.03	0.07
Anti-windup time constant (Tt)	d	not used	not used	0.03	not used	not used		
Controlled variable, SNO2								
Setpoint	g N/m3	1	1	1	1	1	1.00	0.00
Integral of absolute error (IAE)	(g N/m3)*d	0.21702	4.375	1.8182	2.57325	0.9874789	1.99	4.16
Integral of square error (ISE)	(g N/m3)**2*d	0.0903118	4.14	0.84205	1.61783	0.278571	1.39	4.05
Max dev from setpoint (max e)	g N/m3	0.920014	2.51	0.9092	0.9	0.732948	1.19	1.78
Standard deviation of error (std e)	g N/m3	0.2080349	0.445	0.3468	0.480315	0.19955	0.34	0.28
Variance of error (var e)	(g N/m3)**2	0.0432785	0.198	0.12027	0.230702	0.0398202	0.13	0.19
Manipulated variable (MV), Qintrec								
Max deviation of MV (max-min)	m3/d	92232	46845	77424.58	70176.3	92232	75781.98	45387.00
Max dev in MV (delta)	m3/d	10135	12000	8897.2944	6157.58	9998.969	9437.77	5842.42
Std deviation of MV (delta)	m3/d	1769.7859	288	1641.798	1754.69	1764.9688	1443.85	1481.79
Variance of MV (delta)	(m3/d)**2	3132142.1	82944	2695501	3078930	3115114.9	2420926	3049198

Table 13.6 (cont'd): Dynamic (closedloop – i.e. basic control strategy operational) results generated with the *rain* weather file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Oxygen controller for third aerobic reactor								
Controller type		velocity PI	disc PI with aw	cont PI with aw	cont PI	PI		
Proportional gain (K)	1/d/(g (-COD)/m3)	20.8	16.8	500	100	5000	1127.52	4983.20
Integral time constant (Ti)	d	0.002	0.0025	0.001	0.01	0.05	0.01	0.05
Anti-windup time constant (Tt)	d	not used	not used	0.0002	not used	not used	0.00	0.00
Controlled variable, SO5								
Setpoint	g (-COD)/m3	2	2	2	2	2	2.00	0.00
Integral of absolute error (IAE)	(g (-COD)/m3)*d	0.0389263	0.395	0.0069977	0.262958	0.0231699	0.15	0.39
Integral of square error (ISE)	(g (-COD)/m3)**2*d	0.0037126	0.052	5.83E-05	0.022519	0.0001474	0.02	0.05
Max dev from setpoint (max e)	g (-COD)/m3	0.3068	0.386	0.065537	0.249905	0.0173045	0.21	0.37
Standard deviation of error (std e)	g (-COD)/m3	0.0469105	0.0654	0.0028853	0.056719	0.0045911	0.04	0.06
Variance of error (var e)	(g (-COD)/m3)**2	0.0022006	0.00428	8.33E-06	0.00321	2.11E-05	0.00	0.00
Manipulated variable (MV), Kla5								
Max deviation of MV (max-min)	1/d	189.993	240	220.2254	189.837	182.44765	204.50	57.55
Max dev in MV (delta)	1/d	31.43	12	35.7611	28.5663	35.78925	28.71	23.79
Std deviation of MV (delta)	1/d	5.0835676	0.365	5.2715	6.06096	5.1246793	4.38	5.70
Variance of MV (delta)	(1/d)**2	25.84266	0.133	27.7882	36.7353	26.262338	23.35	36.60

Table 13.7: Dynamic (closedloop – i.e. basic control strategy operational) results generated with the *storm* weather file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Effluent average concentrations based on load								
Q =	m3/d	20658.10	20658.00	20658.06	20685.70	20655.22	20663.02	30.48
S_I =	g COD/m3	26.2761	26.3000	26.2992	26.2970	26.2984	26.2941	0.0239
S_S =	g COD/m3	0.9884	1.0100	0.9995	1.0028	0.9963	0.9994	0.0216
X_I =	g COD/m3	5.6359	5.6400	5.6320	5.6313	5.6253	5.6329	0.0147
X_S =	g COD/m3	0.2911	0.2890	0.2874	0.2873	0.2857	0.2881	0.0054
X_BH =	g COD/m3	11.8813	11.9000	11.9024	11.8936	11.8643	11.8883	0.0381
X_BA =	g COD/m3	0.6299	0.6330	0.6311	0.6241	0.6302	0.6297	0.0089
X_P =	g COD/m3	1.9045	1.9100	1.9066	1.9058	1.9040	1.9062	0.0060
S_O =	g (-COD)/m3	1.9895	n/a	1.9975	1.9897	1.9984	1.9938	0.0088
S_NO =	g N/m3	10.5409	11.2000	10.5639	10.7358	10.4571	10.6995	0.7429
S_NH =	g N/m3	3.0915	3.0000	3.0529	3.1230	3.0303	3.0595	0.1230
S_ND =	g N/m3	0.7689	0.7850	0.7767	0.7787	0.7756	0.7770	0.0161
X_ND =	g N/m3	0.0207	0.0206	0.0204	0.0204	0.0203	0.0205	0.0004
S_ALK =	kmol HCO3/m3	n/a	4.4400	4.4881	4.4815	n/a	4.4699	0.0481
TSS =	g SS/m3	15.2571	15.2790	15.2696	15.2565	15.2322	15.2589	0.0468
N_TKN =	g N/m3	5.3344	5.2610	5.3051	5.3758	5.2775	5.3108	0.1148
N_tot =	g N/m3	15.8753	16.4700	15.8689	16.1115	15.7346	16.0121	0.7354
COD_tot =	g COD/m3	47.6073	47.6700	47.6583	47.6418	47.6042	47.6363	0.0658
BOD5_tot =	g/m3	3.1975	3.2100	3.2044	3.2016	3.1942	3.2015	0.0158
Effluent average loads								
S_I =	kg COD/d	542.8150	543.0000	543.2904	543.9700	543.1991	543.2549	1.1550
S_S =	kg COD/d	20.4181	20.8000	20.6486	20.7427	20.5786	20.6376	0.3819
X_I =	kg COD/d	116.4269	117.0000	116.3466	116.4860	116.1913	116.4902	0.8087
X_S =	kg COD/d	6.0145	5.9800	5.9374	5.9427	5.9022	5.9554	0.1123
X_BH =	kg COD/d	245.4447	246.0000	245.8812	246.0280	245.0606	245.6829	0.9674
X_BA =	kg COD/d	13.0132	13.1000	13.0371	12.9100	13.0160	13.0153	0.1900
X_P =	kg COD/d	39.3440	39.4000	39.3860	39.4223	39.3282	39.3761	0.0941
S_O =	kg (-COD)/d	41.0999	n/a	41.2644	41.1577	41.2764	41.1996	0.1765
S_NO =	kg N/d	217.7556	232.0000	218.2292	222.0770	215.9938	221.2111	16.0062
S_NH =	kg N/d	63.8652	62.0000	63.0676	64.6016	62.5913	63.2251	2.6016
S_ND =	kg N/d	15.8840	16.2000	16.0454	16.1075	16.0208	16.0515	0.3160
X_ND =	kg N/d	0.4269	0.4260	0.4219	0.4224	0.4193	0.4233	0.0076
S_ALK =	kmol HCO3/d	n/a	91.8000	92.7160	92.7032	n/a	92.4064	0.9160
TSS =	kg N/d	315.1829	316.1100	315.4412	315.5920	314.6237	315.3900	1.4863
N_TKN =	kg N/d	110.1989	108.6800	109.5924	111.2010	109.0087	109.7362	2.5210
N_tot =	kg N/d	327.9545	340.2400	327.8216	333.2780	325.0025	330.8593	15.2375
COD_tot =	kg COD/d	983.4764	984.6400	984.5272	985.5020	983.2759	984.2843	2.2261
BOD5_tot =	kg/d	66.0535	66.3100	66.1977	66.2270	65.9778	66.1532	0.3322

Table 13.7 (cont'd): Dynamic (closedloop – i.e. basic control strategy operational) results generated with the *storm* weather file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Performance Indices								
I.Q.-index =	kg poll.units/d	43758.11	43800.00	43758.11	43758.11	43758.12	43766.49	41.89
E.Q.-index =	kg poll.units/d	8305.04	8520.00	8304.24	8414.70	8236.28	8356.05	283.72
P_sludge =	kg SS	18260.00	18242.00	18239.73	18233.98	18279.50	18251.04	45.52
P_sludge per day =	kg SS/d	2608.60	2606.00	2605.68	2604.85	2611.36	2607.30	6.51
P_sludge_eff =	kg SS	2206.30	2198.00	2208.09	2209.14	2202.37	2204.78	11.14
P_sludge_eff per day =	kg SS/d	315.20	314.00	315.44	315.59	314.62	314.97	1.59
P_total_sludge =	kg SS	20466.20	20440.00	20447.81	20443.12	20481.86	20455.80	41.86
P_total_sludge per day =	kg SS/d	2923.70	2920.00	2921.12	2920.45	2925.98	2922.25	5.98
Aeration energy =	kWh/d	7284.00	7309.00	7286.06	7298.36	7276.88	7290.86	32.12
Pumping energy =	kWh/d	1818.00	1498.00	1727.31	1645.26	1795.62	1696.84	320.00
The max effluent N_tot level (18 g N/m3)								
was violated during:	d	0.89	1.64	1.10	1.30	0.96	1.18	0.75
i.e.:	% of the time	12.65	23.40	15.77	18.60	13.65	16.81	10.75
The limit was violated at:	occasions	6	8	7	8	6	7.00	2
The max effluent S_NH level (4 g N/m3)								
was violated during:	d	1.93	1.86	1.88	1.91	1.86	1.88	0.07
i.e.:	% of the time	27.50	26.50	26.79	27.23	26.55	26.91	1.00
The limit was violated at:	occasions	7	7	7	7	7	7.00	0
The max effluent TSS level (30 g SS/m3)								
was violated during:	d	0.00	0.02	0.02	0.02	0.01	0.01	0.02
i.e.:	% of the time	0.00	0.30	0.30	0.30	0.15	0.21	0.30
The limit was violated at:	occasions	0	2	2	2	1	1.40	2
Nitrate controller for second anoxic reactor								
Controller type		velocity PI	disc PI with aw	cont PI with aw	cont PI	PI		
Proportional gain (K)	m3/d/(g N/m3)	7500	5040	15000	10000	10000	9508.00	9960.00
Integral time constant (Ti)	d	0.0125	0.007	0.05	0.08	0.01	0.03	0.07
Anti-windup time constant (Tt)	d	not used	not used	0.03	not used	not used		
Controlled variable, SNO2								
Setpoint	g N/m3	1	1	1	1	1	1.00	0.00
Integral of absolute error (IAE)	(g N/m3)*d	0.1851641	4.21	1.761	2.57322	1.0274977	1.95	4.02
Integral of square error (ISE)	(g N/m3)**2*d	0.0662959	3.84	0.83314	1.61778	0.2935525	1.33	3.77
Max dev from setpoint (max e)	g N/m3	0.8827	2.09	1.068	0.9	0.7086099	1.13	1.38
Standard deviation of error (std e)	g N/m3	0.1790472	0.435	0.34494	0.480308	0.2048457	0.33	0.30
Variance of error (var e)	(g N/m3)**2	0.0320579	0.189	0.11899	0.230695	0.2048457	0.16	0.20
Manipulated variable (MV), Qintrec								
Max deviation of MV (max-min)	m3/d	92232	49531	39120	57782.9	92232	66179.58	53112.00
Max dev in MV (delta)	m3/d	10135	10677	12000	5861.37	9382.8445	9611.24	6138.63
Std deviation of MV (delta)	m3/d	1769.7859	1622.53	290	1768.08	1811.8472	1452.45	1521.85
Variance of MV (delta)	(m3/d)**2	3132142.1	2632603.5	84100	3126120	3282790.3	2451551	3198690

Table 13.7 (cont'd): Dynamic (closedloop – i.e. basic control strategy operational) results generated with the *storm* weather file by various simulation software packages.

	Units	GPS-X	FORTRAN	MATLAB/ Simulink	SIMBA	WEST	Average	max-min
Oxygen controller for third aerobic reactor								
Controller type		velocity PI	disc PI with aw	cont PI with aw	cont PI	PI		
Proportional gain (K)	1/d/(g (-COD)/m3)	20.8	16.8	500	100	5000	1127.52	4983.20
Integral time constant (Ti)	d	0.002	0.0025	0.001	0.01	0.05	0.01	0.05
Anti-windup time constant (Tt)	d	not used	not used	0.0002	not used	not used	0.00	0.00
Controlled variable, SO5								
Setpoint	g (-COD)/m3	2	2	2	2	2	2.00	0.00
Integral of absolute error (IAE)	(g (-COD)/m3)*d	0.0451658	0.454	0.024638	0.320457	0.0601024	0.18	0.43
Integral of square error (ISE)	(g (-COD)/m3)**2*d	0.0049911	0.065	3.24E-03	0.033660	0.00585	0.02	0.06
Max dev from setpoint (max e)	g (-COD)/m3	0.3408	0.411	0.22993	0.259643	0.2252782	0.29	0.19
Standard deviation of error (std e)	g (-COD)/m3	0.0543653	0.713	0.021358	0.069344	0.0289232	0.18	0.69
Variance of error (var e)	(g (-COD)/m3)**2	0.0029556	0.508	4.56E-04	0.004809	8.37E-04	0.10	0.51
Manipulated variable (MV), Kla5								
Max deviation of MV (max-min)	1/d	186.737	240	193.166	193.19	193.37321	201.29	53.26
Max dev in MV (delta)	1/d	36.229	12	36.7754	29.7863	39.85488	30.93	27.85
Std deviation of MV (delta)	1/d	5.9096592	0.396	5.5011	6.69429	5.684237	4.84	6.30
Variance of MV (delta)	(1/d)**2	34.924072	0.157	30.2624	44.8135	32.31055	28.49	44.66