Benchmark Simulation Model no. 2 (BSM2)

J. Alex, L. Benedetti, J. Copp, K.V. Gernaey, U. Jeppsson,
I. Nopens, M.N. Pons, C. Rosen, J.P. Steyer and
P. Vanrolleghem

Summary

This document presents in details the final state of Benchmark Simulation Model no. 2 (BSM2). The model equations to be implemented for the proposed layout, the procedure to test the implementation and the performance criteria to be used are described, as well as the sensors and control handles. Further explanations concerning the reasoning which ended up with the choices made can be found in the Corresponding Technical Reports. Furthermore supplementary informations are given in the documents accompanying the software.

Table of Contents

1. INTRODUCTION

Wastewater treatment plants (WWTPs) are large non-linear systems subject to large perturbations in influent flow rate and pollutant load, together with uncertainties concerning the composition of the incoming wastewater. Nevertheless these plants have to be operated continuously, meeting stricter and stricter regulations.

Many control strategies have been proposed in the literature but their evaluation and comparison, either practical or based on simulation, is difficult. This is due to a number of reasons, including: (1) the variability of the influent; (2) the complexity of the biological and biochemical phenomena; (3) the large range of time constants (varying from a few minutes to several days); (4) the lack of standard evaluation criteria (among other things, due to region specific effluent requirements and cost levels).

It is thus difficult to judge the particular influence of an applied control strategy on reported plant performance increase, as the reference situation is often not properly characterized. Due to the complexity of the systems it takes much effort to develop alternative controller approaches and, as a consequence, a fair comparison between different control strategies is only made seldomly. And even if this is done, it remains difficult to conclude to what extent the proposed solution is process or location specific.

To enhance the acceptance of innovating control strategies, the performance evaluation should be based on a rigorous methodology including a reference simulation model, a precise plant layout, well-defined controllers, performance criteria and test procedures.

From 1998 to 2004, the development of benchmark tools for simulation-based evaluation of control strategies for activated sludge plants has been undertaken in Europe by Working Groups of COST Action 682 and 624 (Alex *et al*., 1999). This development work is now continued under the umbrella of the IWA Task Group on Benchmarking of Control Strategies for WWTPs.

The benchmark is a simulation environment defining a plant layout, a simulation model, influent loads, test procedures and evaluation criteria. For each of these items, compromises were pursued to combine plainness with realism and accepted standards. Once the user has validated the simulation code, any control strategy can be applied and the performance can be evaluated according to a defined set of criteria. The benchmark is not linked to a particular simulation platform: direct coding (C/C++, Fortran, Matlab®) as well as commercial WWTP simulation software packages (such as Simba®, West®, GPS-X®) can be used. For this reason, the full set of equations and all the parameter values are available in the present document.

The first layout (BSM1) (Alex *et al*., 2009) was relatively simple (Figure 1). The BSM1 plant is composed of a five-compartment activated sludge reactor consisting of two anoxic tanks followed by three aerobic tanks. It thus combines nitrification with predenitrification in a configuration that is commonly used for achieving biological nitrogen removal in full-scale plants. The activated sludge reactors are followed by a secondary clarifier.

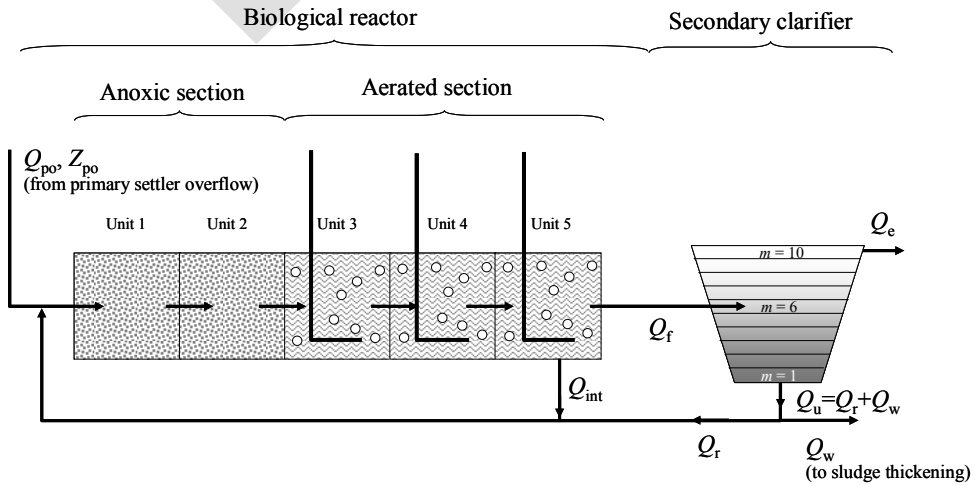


Figure 1: General overview of the BSM1 plant within BSM2

The BSM2 layout (Figure 2) includes BSM1 for the biological treatment of the wastewater and the sludge treatment. A primary clarifier, a thickener for the sludge wasted from the BSM1 clarifier, a digester for treatment of the solids wasted from the primary clarifier and the thickened secondary sludge as well as a dewatering unit have been also added. The liquids collected in the thickening and dewatering steps are recycled ahead of the primary settler. Different possible control handles such as pumps, valves, aeration, etc. are also shown in Figure 2.

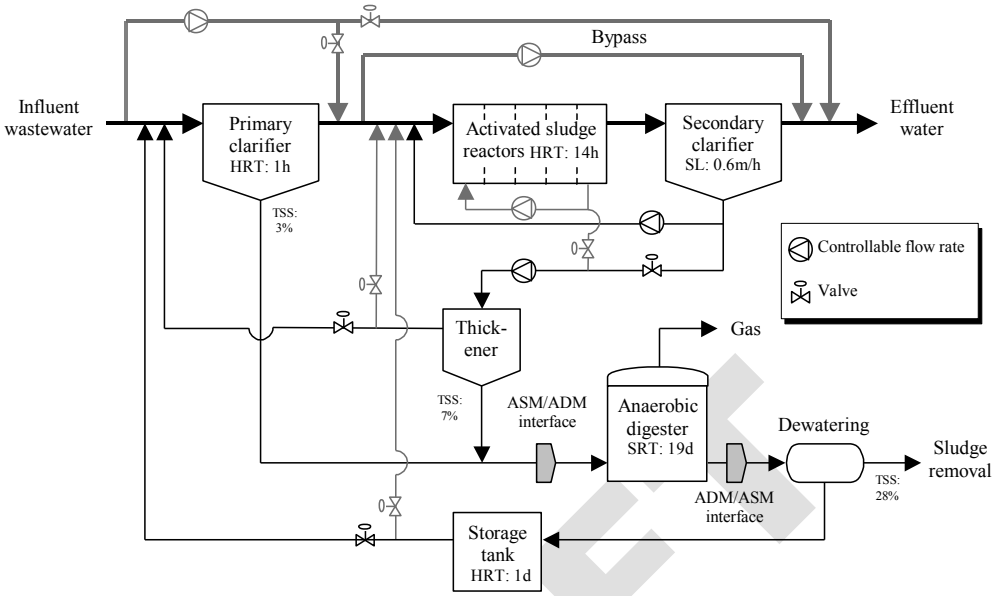


Figure 2: General overview of the BSM2 plant

The purpose of the present document is to describe in details the BSM2 benchmark system. An important part of the development of BSM2 has been to implement the ADM1 model for the anaerobic digester. This has implied some slight changes with respect to the original version of ADM1 as well as the development of calculation procedures in order to have a reasonable calculation time for the whole BSM2 plant (Rosen *et al.*, 2006; Rosen and Jeppsson, 2009). Furthermore, interfaces to transform the ASM1 variables into ADM1 variables (and vice-versa) had to be implemented (Nopens *et al.*, 2009). More details on the model development for some units can be found in the other sections of the Technical Report. Finally, to facilitate the understanding of the modelling, Figure 3 summarizes the notations used for the various flow rates throughout the BSM2 plant.

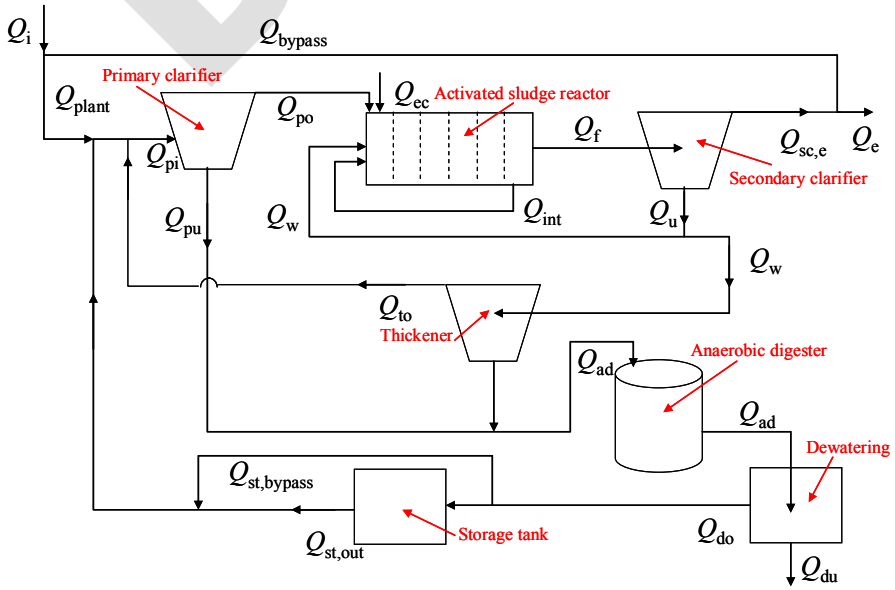


Figure 3: BSM2 plant simplified layout with notation used for flow rates

2. MODELING OF THE ACTIVATED SLUDGE SECTION

2.1. General characteristics

The plant is designed for an average influent dry-weather flow rate of 20,648.36 $m^3.d^{-1}$ and an average biodegradable COD in the influent of 592.53 $g.m^{-3}$. Its hydraulic retention time (based on average dry weather flow rate and total tank volume – i.e. primary clarifier (900 m^3) + biological reactor (12,000 m^3) + secondary clarifier (6,000 m^3) – of 18,900 m^3) is 22 hours.

The influent dynamics are defined for 609 days by means of a single file, which takes into account rainfall effect and temperature.

2.2. Bioprocess model

The Activated Sludge Model no. 1 (ASM1; Henze *et al.*, 1987) has been selected to describe the biological phenomena taking place in the biological reactor (Figure 4).

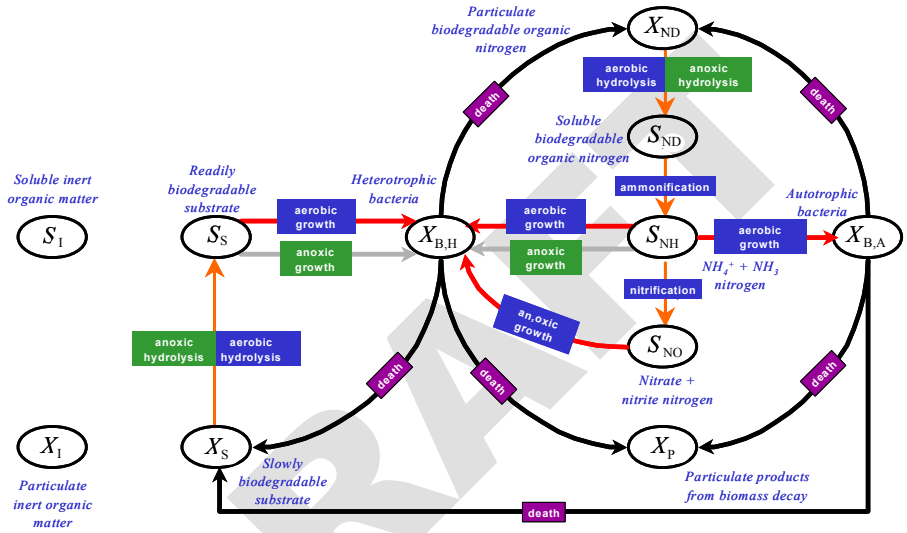


Figure 4: General overview of ASM1

2.2.1. List of variables

The list of state variables, with their definition and appropriate notation, is given in Table 1. To these variables should be added T_{as} for temperature.

Table 1: List of ASM1 variables

Definition	Notation
Soluble inert organic matter	S_I
Readily biodegradable substrate	S_S
Particulate inert organic matter	X_I
Slowly biodegradable substrate	X_S
Active heterotrophic biomass	$X_{B,H}$
Active autotrophic biomass	$X_{B,A}$
Particulate products arising from biomass decay	X_P
Oxygen	S_O
Nitrate and nitrite nitrogen	S_{NO}
$NH_4^+ + NH_3$ nitrogen	S_{NH}
Soluble biodegradable organic nitrogen	S_{ND}
Particulate biodegradable organic nitrogen	X_{ND}
Alkalinity	S_{ALK}

2.2.2. List of processes

Eight basic processes (ρ_k, k = 1 to 8) are used to describe the biological behavior of the system.

- *j = 1*: Aerobic growth of heterotrophs

$$\rho_{1,as} = \mu_{HT}\left(\frac{S_S}{K_S + S_S}\right)\left(\frac{S_O}{K_{O,H} + S_O}\right)X_{B,H} \quad (1)$$

with:

$$\mu_{HT} = \mu_H \cdot \exp\left(\left(Ln\left(\frac{\mu_H}{3}\right)\Big/5\right)\cdot(T_{as} - 15)\right) \quad (2)$$

- *j = 2*: Anoxic growth of heterotrophs

$$\rho_{2,as} = \mu_{HT}\left(\frac{S_S}{K_S + S_S}\right)\left(\frac{K_{O,H}}{K_{O,H} + S_O}\right)\left(\frac{S_{NO}}{K_{NO} + S_{NO}}\right)\eta_g X_{B,H} \quad (3)$$

- *j = 3*: Aerobic growth of autotrophs

$$\rho_{3,as} = \mu_{AT}\left(\frac{S_{NH}}{K_{NH} + S_{NH}}\right)\left(\frac{S_O}{K_{O,A} + S_O}\right)X_{B,A} \quad (4)$$

with:

$$\mu_{AT} = \mu_A \cdot \exp\left(\left(Ln\left(\frac{\mu_A}{0.3}\right)\Big/5\right)\cdot(T_{as} - 15)\right) \quad (5)$$

- *j = 4*: Decay of heterotrophs

$$\rho_{4,as} = b_{HT} X_{B,H} \quad (6)$$

with:

$$b_{HT} = b_H \cdot \exp\left(\left(Ln\left(\frac{b_H}{0.2}\right)\Big/5\right)\cdot(T_{as} - 15)\right) \quad (7)$$

- *j = 5*: Decay of autotrophs

$$\rho_{5,as} = b_{AT} X_{B,A} \quad (8)$$

$$\text{with } b_{AT} = b_A \cdot \exp\left(\left(Ln\left(\frac{b_A}{0.03}\right)\Big/5\right)\cdot(T_{as} - 15)\right) \quad (9)$$

- *j = 6*: Ammonification of soluble organic nitrogen

(10)

$$\text{with } k_{aT} = k_a \cdot \exp\left(\left(Ln\left(\frac{k_a}{0.04}\right)\Big/5\right)\cdot(T_{as} - 15)\right) \quad (11)$$

- *j = 7*: Hydrolysis of entrapped organics

(12)

$$\text{with } k_{hT} = k_h \cdot \exp\left(\left(Ln\left(\frac{k_h}{2.5}\right)\Big/5\right)\cdot(T_{as} - 15)\right) \quad (13)$$

- *j = 8*: Hydrolysis of entrapped organic nitrogen

(14)

2.2.3. Observed conversion rates

The observed conversion rates (r_k) result from combinations of the basic processes:

- S_I (k = 1)

(15)

- S_S (k= 2)

(16)

- X_I (k = 3)

(17)

- X_S (k = 4)

(18)

- $X_{B,H}$ (k = 5)

(19)

- $X_{B,A}$ (k = 6)

(20)

- X_P (k = 7)

(21)

- S_O (k = 8)

(22)

- S_{NO} (k = 9)

(23)

- S_{NH} (k = 10)

(24)

- S_{ND} (k = 11)

(25)

- X_{ND} (k = 12)

(26)

- S_{ALK} (k = 13)

(27)

2.2.4. Biological parameter values

The base biological parameter values used in the activated sludge section of BSM2 correspond to a temperature of 15°C. The stoichiometric parameters are listed in Table 2 and the kinetic parameters in Table 3.

Table 2: Stoichiometric parameters

Parameter	Unit	Value
Y_A	g cell COD formed.(g N oxidized)$^{-1}$	0.24
Y_H	g cell COD formed.(g COD oxidized)$^{-1}$	0.67
f_P	dimensionless	0.08
i_{XB}	g N.(g COD)$^{-1}$ in biomass	0.08
i_{XP}	g N.(g COD)$^{-1}$ in particulate products	0.06

Table 3: Kinetic parameters

Parameter	Unit	Value
μ_H	d^{-1}	4.0
K_S	g COD.m^{-3}	10.0
$K_{O,H}$	g (–COD).m^{-3}	0.2
K_{NO}	g NO_3-N.m^{-3}	0.5
b_H	d^{-1}	0.3
η_g	dimensionless	0.8
η_h	dimensionless	0.8
k_h	g slowly biodegradable COD.(g cell COD . d)$^{-1}$	3.0
K_X	g slowly biodegradable COD.(g cell COD)$^{-1}$	0.1
μ_A	d^{-1}	0.5
K_{NH}	g NH_3-N.m^{-3}	1.0
b_A	d^{-1}	0.05
$K_{O,A}$	g (–COD).m^{-3}	0.4
k_a	m^3.(g COD.d)$^{-1}$	0.05

2.3. Detailed activated sludge section layout

2.3.1. Bioreactor (general characteristics)

According to Figure 1, the general characteristics of the bioreactor for the default case are:
Number of compartments: 5
Non-aerated compartments: compartments 1-2
Aerated compartments: compartments 3-5

For each compartment, the following variables have been defined (k = 1 to 5):

- Flow rate: Q_k
- Concentration: $Z_{as,k}$
- Volume:
  Non-aerated compartments: $V_{as,1} = V_{as,2}$ = 1,500 m^3
  Aerated compartments: $V_{as,3} = V_{as,,4} = V_{as,5}$ = 3,000 m^3
- Reaction rate: r_k

2.3.2. Reactor mass balances (general formula)

The general equations for mass balancing are as follows:

- For k = 1 (compartment 1)

$$\frac{dZ_{as,1}}{dt} = \frac{1}{V_{as,1}}\left(Q_{int}Z_{int} + Q_r Z_r + Q_{po}Z_{po} + r_{Z,1}V_{as,1} - Q_1 Z_{as,1}\right) \quad (28)$$

$$Q_1 = Q_{int} + Q_r + Q_{po} \quad (29)$$

where Q_{int} is the internal recycle from compartment 5, Q_r the external recycle from the underflow of the secondary clarifier and Q_{po} the overflow rate of the primary clarifier.

- For k = 2 to 5

$$\frac{dZ_{as,k}}{dt} = \frac{1}{V_{as,k}}\left(Q_{k-1}Z_{as,k-1} + r_{Z,k}V_{as,k} - Q_k Z_{as,k}\right) \quad (30)$$

$$Q_k = Q_{k-1} \quad (31)$$

- Special case for oxygen ($S_{O,as,k}$)

$$\frac{dS_{O,as,k}}{dt} = \frac{1}{V_{as,k}}\left(Q_{k-1}S_{O,as,k-1} + r_{Z,k}V_{as,k} + (K_L a)_k V_{as,k}\left(S_{O,as}^{sat} - S_{O,as,k}\right) - Q_k S_{O,as,k}\right) \quad (32)$$

where the saturation concentration for oxygen is $S_{O,as}^{sat}$ and is a function of temperature (cf §2.4).

$r_{Z,k}$ stands for the appropriate conversion rate, depending upon the state variable considered (cf §§ 2.2.3).

- Temperature (T_{as})

An adiabatic temperature balance is assumed (no heat exchange with the environment)

$$\frac{dT_{as,k}}{dt} = \frac{1}{V_{as,k}}\left(Q_{k-1}T_{as,k-1} - Q_k T_{as,k}\right) \quad (33)$$

- Miscellaneous

$$Z_{int} = Z_{as,5} \quad (34)$$

$$Z_f = Z_{as,5} \quad (35)$$

$$Z_w = Z_r \quad (36)$$

$$Q_f = Q_{sc,e} + Q_r + Q_w = Q_{sc,e} + Q_{sc,u} \quad (37)$$

Where $Q_{sc,e}$ and $Q_{sc,u}$ are the overflow and underflow rates from the secondary clarifier respectively and Q_w is the wastage flow rate.

2.3.3. Secondary clarifier

The secondary clarifier is modeled as a 10 layers non-reactive unit (i.e. no biological reaction). The 6th layer (counting from bottom to top) is the feed layer. The secondary clarifier has an area (A) of 1,500 m^2. The height of each layer m (z_m) is equal to 0.4 m, for a total height of 4 m. Therefore the secondary clarifier volume is equal to 6,000 m^3.

The solid flux due to gravity is $J_s = v_s(X_{sc})X_{sc}$ where X_{sc} is the total sludge concentration. A double-exponential settling velocity function (Takács *et al.,* 1991) has been selected:

$$v_s(X_{sc}) = \max\left[0, \min\left\{v_0', v_0\left(e^{-r_h(X_{sc}-X_{min})} - e^{-r_p(X_{sc}-X_{min})}\right)\right\}\right] \quad (38)$$

with $X_{min} = f_{ns}X_f$. X_f is the total solid concentration from the biological reactor. The parameter values for the settling velocity function are given in Table 4.

Table 4: Settling parameters

	Parameter	Units	Value
Maximum settling velocity	v_0'	m.d^{-1}	250.0
Maximum Vesilind settling velocity	v_0	m.d^{-1}	474
Hindered zone settling parameter	r_h	m^3.(g SS)$^{-1}$	0.000576
Flocculant zone settling parameter	r_p	m^3.(g SS)$^{-1}$	0.00286
Non-settleable fraction	f_{ns}	dimensionless	0.00228

The upward (v_{up}) and downward (v_{dn}) velocities are calculated as:

$$v_{dn} = \frac{Q_{sc,u}}{A} = \frac{Q_r + Q_w}{A} \quad (39)$$

$$v_{up} = \frac{Q_{sc,e}}{A} \quad (40)$$

According to these notations, the mass balances for the sludge are written as:
For the feed layer (m = 6):

$$\frac{dX_{sc,m}}{dt} = \frac{\frac{Q_f X_f}{A} + J_{sc,m+1} - (v_{up} + v_{dn}) X_{sc,m} - \min(J_{s,m}, J_{s,m-1})}{z_m} \quad (41)$$

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

$$\frac{dX_{sc,m}}{dt} = \frac{v_{dn}(X_{sc,m+1} - X_{sc,m}) + \min(J_{s,m}, J_{s,m+1}) - \min(J_{s,m}, J_{s,m-1})}{z_m} \quad (42)$$

For the bottom layer (m = 1):

$$\frac{dX_{sc,1}}{dt} = \frac{v_{dn}(X_{sc,2} - X_{sc,1}) + \min(J_{s,2}, J_{s,1})}{z_1} \quad (43)$$

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

$$\frac{dX_{sc,m}}{dt} = \frac{v_{up}(X_{sc,m-1} - X_{sc,m}) + J_{sc,m+1} - J_{sc,m}}{z_m} \quad (44)$$

$$J_{sc,j} = \begin{cases} \min(v_{s,j} X_{sc,j}, v_{s,j-1} X_{sc,j-1}) \; if \; X_{sc,j-1} > X_t \\ \text{or} \\ v_{s,j} X_{sc,j} \quad if \quad X_{sc,j-1} \le X_t \end{cases} \quad (45)$$

For the top layer (m = 10)

$$\frac{dX_{sc,10}}{dt} = \frac{v_{up}(X_{sc,9} - X_{sc,10}) - J_{sc,10}}{z_{10}} \quad (46)$$

$$\text{with } J_{sc,10} = \begin{cases} \min(v_{s,10} X_{sc,10}, v_{s,9} X_{sc,9}) \; if \; X_{sc,9} > X_t \\ \text{or} \\ v_{s,10} X_{sc,10} \quad if \quad X_{sc,9} \le X_t \end{cases} \quad (47)$$

The threshold concentration X_t is equal to 3,000 g.m^{-3}

For the soluble components (including dissolved oxygen and temperature (T_{sc})), each layer represents a completely mixed volume and the concentrations of soluble components are calculated accordingly:

For the feed layer (m = 6)

$$\frac{dZ_{sc,m}}{dt} = \frac{\frac{Q_f Z_f}{A} - (v_{dn} + v_{up}) Z_{sc,m}}{z_m} \quad (48)$$

For the layers m = 1 to 5

$$\frac{dZ_{sc,m}}{dt} = \frac{v_{dn}(Z_{sc,m+1} - Z_{sc,m})}{z_m} \quad (49)$$

For the layers m = 7 to 10

$$\frac{dZ_{sc,m}}{dt} = \frac{v_{up}(Z_{sc,m-1} - Z_{sc,m})}{z_m} \quad (50)$$

The concentrations in the recycle and wastage flow are equal to those of the first layer (bottom layer):

$$Z_u = Z_{sc,1} \quad (51)$$

Calculation of the sludge concentration is straightforward from the concentrations in compartment 5 of the activated sludge reactor:

$$X_f = \frac{1}{fr_{COD-SS}}\left(X_{S,as,5} + X_{P,as,5} + X_{I,as,5} + X_{B,H,as,5} + X_{B,A,as,5}\right) = 0.75\left(X_{S,as,5} + X_{P,as,5} + X_{I,as,5} + X_{B,H,as,5} + X_{B,A,as,5}\right) \quad (52)$$

given a COD to SS conversion factor, fr_{COD-SS}, equal to 4/3. The same principle is applied for X_u (in the clarifier underflow) and X_e (at the secondary clarifier exit).

To calculate the distribution of particulate concentrations in the recycle and the wastage flows, their ratios with respect to the total solid concentration are assumed to remain constant across the clarifier:

$$\frac{X_{S,as,5}}{X_f} = \frac{X_{S,sc,1}}{X_u} \quad (53)$$

Similar equations hold for $X_{P,sc,1}$, $X_{I,sc,1}$, $X_{B,H,sc,1}$, $X_{B,A,sc,1}$, and $X_{ND,sc,1}$. Note that this assumption means that the dynamics of the fractions of particulate concentrations in the inlet of the clarifier will be directly propagated to the clarifier underflow and overflow, without taking into account the normal retention time in the clarifier.

The sludge age calculation is based on the total amount of biomass present in the system, i.e. the reactor and the settler:

$$SRT = \frac{TX_{as} + TX_{sc}}{\phi_e + \phi_w} \quad (54)$$

where TX_{as} is the total amount of biomass present in the reactor:

$$TX_{as} = \sum_{k=1}^{k=n}\left(X_{B,H,as,k} + X_{B,A,as,k}\right)\cdot V_{as,k} \text{ with } n = 5 \quad (55)$$

TX_{sc} is the total amount of biomass present in the secondary clarifier:

$$TX_{sc} = \sum_{j=1}^{j=m}\left(X_{B,H,sc,j} + X_{B,A,sc,j}\right)\cdot z_j \cdot A \text{ with } m = 10 \quad (56)$$

ϕ_e is the loss rate of biomass in the secondary clarifier overflow:

$$\phi_e = \left(X_{B,H,sc,m} + X_{B,A,sc,m}\right)\cdot Q_{sc,e} \quad (57)$$

and ϕ_w is the loss rate of biomass in the wastage flow:

$$\phi_w = \left(X_{B,H,sc,1} + X_{B,A,sc,1}\right)\cdot Q_w \quad (58)$$

In an actual plant the sludge age is measured based on the total amount of solids present in the system:

$$SRT_{meas} = \frac{TSS_{as} + TSS_{sc}}{\psi_e + \psi_w} \quad (59)$$

where TSS_{as} is the total amount of solids present in the reactor:

$$TSS_{as} = \sum_{k=1}^{k=n} TSS_{as,k}\cdot V_{as,k} \quad (60)$$

with $n = 5$ and $TSS_{as,k} = \frac{1}{fr_{COD-SS}}\left(X_{S,as,k} + X_{P,as,k} + X_{I,as,k} + X_{B,H,as,k} + X_{B,A,as,k}\right)$ (61)

TSS_{sc} is the total amount of solids present in the clarifier:

$$TSS_{sc} = \sum_{j=1}^{j=m} TSS_{sc,j}\cdot z_j \cdot A \quad (62)$$

with $m = 10$ and $TSS_{sc,j} = \frac{1}{fr_{COD-SS}}\left(X_{S,sc,j} + X_{P,sc,j} + X_{I,sc,j} + X_{B,H,sc,j} + X_{B,A,sc,j}\right)$ (63)

ψ_{fe} is the loss rate of solids in the secondary clarifier overflow:

$$\psi_e = TSS_{sc,m}\cdot Q_{sc,e} \quad (64)$$

with $TSS_{sc,m} = \frac{1}{fr_{COD-SS}}\left(X_{S,sc,m}+X_{P,sc,m}+X_{I,sc,m}+X_{B,H,sc,m}+X_{B,A,sc,m}\right)$ (65)

for $m = 10$.

ψ_w is the loss rate of solids in the wastage flow:

$\psi_w = TSS_{sc,1} \cdot Q_w$ (66)

with $TSS_{sc,1} = \frac{1}{fr_{COD-SS}}\left(X_{S,sc,1}+X_{P,sc,1}+X_{I,sc,1}+X_{B,H,sc,1}+X_{B,A,sc,1}\right)$ (67)

2.3.4. Plant effluent composition

In BSM2, the plant effluent composition is based on the secondary clarifier overflow and the raw wastewater that might be by-passed at the inlet of the plant. For any composition state variable:

$Z_e = \left(Q_{sc,e} \cdot Z_{sc,10} + Q_{bypass} \cdot Z_i\right) / \left(Q_{sc,e} + Q_{bypass}\right)$ (68)

where Z_i is the concentration in the raw wastewater.

2.4. Oxygen transfer coefficient

The oxygen transfer coefficient, K_La, depends on temperature. ASCE (1993) presents the generally accepted dependency of the oxygen transfer coefficient K_La on temperature:

$K_La(T) = 1.024^{(T-15.)} \cdot K_La(15°C)$ with K_La in d^{-1} and T in °C. (69)

2.5. Oxygen concentration at saturation

Solubility of oxygen is dependent on temperature, decreasing with decreasing temperature.

$S_{O,as}^{sat}(T_{as}) = 0.9997743214 \cdot \frac{8}{10.5} \cdot 6791.5 \cdot K(T_K)$ (70)

With $K(T_K) = 56.12\, e^{A+B/T^*+C\ln T^*}$ (71)

The formula is valid in the range 273.15 K —348.15 K, where $T^*=T_K/100$ (K), $T_K = T_{as}$(°C) + 273.15, A = –66.7354, B = 87.4755 C = 24.4526.

3. MODELING OF THE ANAEROBIC DIGESTER

3.1. Introduction

The ADM1 implementation deviates somewhat from the model description in Batstone *et al.* (2002). There are mainly three reasons for this. Firstly, the ADM1 is implemented so that it is consistent with the other sections of BSM2. Secondly, the computational requirements must be regarded. Thirdly, no explicit values are given in Batstone *et al.* (2002) with regard to carbon and nitrogen contents of some state variables.

3.1.1. Acid-base equations

The acid-base equilibrium equations play an important role in ADM1 (e.g. for pH calculations). For persons primarily familiar with AS models, these equations may create a problem as they do not normally appear in those. Moreover, (Batstone *et al.,* 2002) focus more on how the implementation should be done by implicit algebraic equations and is not completely clear on the ODE implementation. The general model matrix describes the transformations of valerate ($S_{va,total}$), butyrate, propionate, acetate, inorganic carbon and inorganic nitrogen. However, all these substances are made up by acid-base pairs (e.g. $S_{va,total} = S_{va^-} + S_{hva}$). It is suggested in Batstone *et al.* (2002) that when using ODEs, the equations are defined for each acid and base, respectively. Based on our experiences it is more advantageous to implement the ODEs based on the total and one of the acid-base components instead. The remaining part can always be calculated as the total minus the calculated part. This approach actually makes the model more understandable also in other respects and due to numerical issues (we are subtracting very small and similar sized numbers) the error of calculated outputs are much closer to the solution a differential-algebraic equation (DAE) set-up would provide (when using a numerical solver with the same tolerance to integrate the ODEs). Using valerate as an example, the process rate (A4) in (Batstone *et al.,* 2002) is:

$$K_{A,Bva}\left(S_{va^-}S_{H^+} - K_{a,va}S_{hva}\right) \quad (72)$$

and herein we replace S_{hva} by $S_{va,total} - S_{va^-}$ and get

$$K_{A,Bva}\left(S_{va^-}\left(K_{a,va} + S_{H^+}\right) - K_{a,va}S_{hva}\right) \quad (73)$$

and, consequently, change the stoichiometry since S_{va} is not affected when the equilibrium of S_{va^-} is changing. If using the suggested implicit solver to calculate the pH (or S_{H^+}) at every integration step (see below) then the above problem will no longer be an issue.

The choice of a ODE or DAE system for modeling the pH should not affect the overall results of the model. The DAE can be said to be an approximation of the ODE since, naturally, the pH dynamics are not instantaneous. However, it is very common to model the dynamics as a DAE system in biochemical/chemical engineering. Thus, the rate coefficients $k_{A,Bi}$ should be defined in such a way that the ODE produces the same results as the DAE. In Batstone *et al.* (2002), it is recommended that the coefficients should be chosen so that they are at least one order of magnitude faster (larger) than the fastest time constant of the remaining system and the value 1.10^8 $M^{-1}d^{-1}$ is recommended. However, this is not sufficient. For the ODE to yield identical results, the rate coefficients need to be larger and a value of 1.10^{10} $M^{-1}d^{-1}$ is more appropriate.

3.1.2. Temperature dependencies

In order to better allow for reasonable results for different temperatures within the digester, the benchmark ADM1 implementation now uses the complete information as stated in the ADM1 STR with regard to temperature dependency of several physiochemical parameters (see the table for physiochemical parameters). This means that a model user can work with different temperatures when investigation the system without having to recalculate these parameters. The parameters that are now considered to be functions of temperature are: K_W, $K_{a,co2}$, $K_{a,IN}$, $K_{H,co2}$, $K_{H,ch4}$, $K_{H,h2}$ and $p_{gas,h2o}$ (i.e. water vapor saturation pressure).

The K_a values for the organic acids are not assumed to vary within the selected temperature range (0 - 60 °C) and are assumed to be constants (see also Batstone *et al.* (2002), p. 39). For an even better temperature dependency of the AD model many of the biochemical parameter values would also need to be described as functions of temperature.

3.2. Model equations

3.2.1. Process rates

The biochemical process rates are defined as:

Disintegration:

$$\rho_{1,ad} = k_{dis} \cdot X_c \quad (74)$$

Hydrolysis of carbohydrates:

$$\rho_{2,ad} = k_{hyd,ch} \cdot X_{ch} \quad (75)$$

Hydrolysis of proteins:

$$\rho_{3,ad} = k_{hyd,pr} \cdot X_{pr} \quad (76)$$

Hydrolysis of lipids:

$$\rho_{4,ad} = k_{hyd,li} \cdot X_{li} \quad (77)$$

Uptake of sugars:

$$\rho_{5,ad} = k_{m,su} \cdot \frac{S_{su}}{K_{S,su} + S_{su}} \cdot X_{su} \cdot I_5 \quad (78)$$

Uptake of amino-acids:

$$\rho_{6,ad} = k_{m,aa} \cdot \frac{S_{aa}}{K_{S,aa} + S_{aa}} \cdot X_{aa} \cdot I_6 \quad (79)$$

Uptake of LCFA:

$$\rho_{7,ad} = k_{m,fa} \cdot \frac{S_{fa}}{K_{S,fa} + S_{fa}} \cdot X_{fa} \cdot I_7 \quad (80)$$

Uptake of valerate:

$$\rho_{8,ad} = k_{m,c4} \cdot \frac{S_{va}}{K_{S,c4} + S_{va}} \cdot X_{c4} \cdot \frac{S_{va}}{S_{bu} + S_{va}} \cdot I_8 \quad (81)$$

Uptake of butyrate:

$$\rho_{9,ad} = k_{m,c4} \cdot \frac{S_{bu}}{K_{S,c4} + S_{bu}} \cdot X_{c4} \cdot \frac{S_{bu}}{S_{va} + S_{bu}} \cdot I_9 \quad (82)$$

Uptake of propionate:

$$\rho_{10,ad} = k_{m,pro} \cdot \frac{S_{pro}}{K_{S,pro} + S_{pro}} \cdot X_{pro} \cdot I_{10} \quad (83)$$

Uptake of acetate:

$$\rho_{11,ad} = k_{m,ac} \cdot \frac{S_{ac}}{K_{S,ac} + S_{ac}} \cdot X_{ac} \cdot I_{11} \quad (84)$$

Uptake of hydrogen:

$$\rho_{12,ad} = k_{m,h2} \cdot \frac{S_{h2}}{K_{S,h2} + S_{h2}} \cdot X_{h2} \cdot I_{12} \quad (85)$$

Decay of X_{su}:

$$\rho_{13,ad} = k_{dec,Xsu} \cdot X_{su} \quad (86)$$

Decay of X_{aa}:

$$\rho_{14,ad} = k_{dec,Xaa} \cdot X_{aa} \quad (87)$$

Decay of X_{fa}:

$$\rho_{15,ad} = k_{dec,Xfa} \cdot X_{fa} \quad (88)$$

Decay of X_{c4}:

$$\rho_{16,ad} = k_{dec,Xc4} \cdot X_{c4} \quad (89)$$

Decay of X_{pro}:

$$\rho_{17,ad} = k_{dec,Xpro} \cdot X_{pro} \quad (90)$$

Decay of X_{ac}:

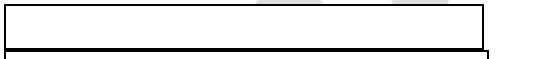
$$\rho_{18,ad} = k_{dec,Xac} \cdot X_{ac} \quad (91)$$

Decay of X_{h2}:

$$\rho_{19,ad} = k_{dec,Xh2} \cdot X_{h2} \quad (92)$$

In the expressions for $\rho_{8,ad}$ and $\rho_{9,ad}$ (Eqs. 81 and 82), a small constant (1.10^{-6}) can be been added at the denominator to the sum (S_{va} + S_{bu}) in order to avoid division by zero in the case of poor choice of initial conditions for S_{va} and S_{bu}, respectively.

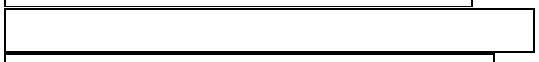
The acid-base rates for the ODE implementation are as follows:

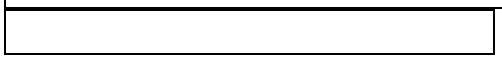

(93)


(94)

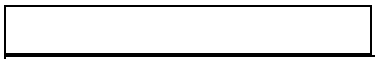

(95)

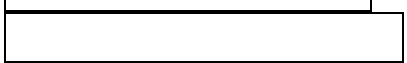

(96)

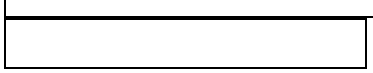

(97)


(98)

The gas transfer rates (note that S_{co2} is used in the expression for $\rho_{T,10}$, not S_{IC}) are written as:


(99)


(100)


(101)

3.2.2. Process inhibition

The process inhibition terms are expressed as :

$$I_5 = I_6 = I_{pH,aa} \cdot I_{IN,lim} \quad (102)$$

$$I_7 = I_{pH,aa} \cdot I_{IN,lim} \cdot I_{h2,fa} \quad (103)$$

$$I_8 = I_9 = I_{pH,aa} \cdot I_{IN,lim} \cdot I_{h2,c4} \quad (104)$$

$$I_{10} = I_{pH,aa} \cdot I_{IN,lim} \cdot I_{h2,pro} \quad (105)$$

$$I_{11} = I_{pH,ac} \cdot I_{IN,lim} \cdot I_{nh3} \quad (106)$$

$$I_{12} = I_{pH,h2} \cdot I_{IN,lim} \quad (107)$$

$$I_{IN,lim} = \frac{1}{1 + K_{S,IN}/S_{IN}} \quad (108)$$

$$I_{h2,fa} = \frac{1}{1 + S_{h2}/K_{I,h2,fa}} \quad (109)$$

$$I_{h2,c4} = \frac{1}{1 + S_{h2}/K_{I,h2,c4}} \quad (110)$$

$$I_{h2,pro} = \frac{1}{1 + S_{h2}/K_{I,h2,pro}} \quad (111)$$

$$I_{nh3} = \frac{1}{1 + S_{nh3}/K_{I,nh3}} \quad (112)$$

Batstone *et al.* (2002) used switch functions to account for inhibition due to pH. These functions are, however, discontinuous and in a stiff system, such a switch can favour numerical instabilities. To reduce this risk, a number of alternative functions can be used to express the inhibition due to pH. Siegrist *et al.* (2002) used a Hill inhibition function based on the hydrogen ion concentration. This solution has been chosen for BSM2. For the ADM1, this gives the following expressions:

$$I_{pH,aa} = \frac{K_{pH}^{n_{aa}}}{S_{H^+}^{n_{aa}} + K_{pH}^{n_{aa}}} \text{ with } K_{pH} = 10^{-\frac{pH_{LL,aa} + pH_{UL,aa}}{2}} \text{ and } n_{aa} = \frac{3.0}{pH_{UL,aa} - pH_{LL,aa}} \quad (113)$$

$$I_{pH,ac} = \frac{K_{pH}^{n_{ac}}}{S_{H^+}^{n_{ac}} + K_{pH}^{n_{ac}}} \text{ with } K_{pH} = 10^{-\frac{pH_{LL,ac} + pH_{UL,ac}}{2}} \text{ and } n_{ac} = \frac{3.0}{pH_{UL,ac} - pH_{LL,ac}} \quad (114)$$

$$I_{pH,h2} = \frac{K_{pH}^{n_{h2}}}{S_{H^+}^{n_{h2}} + K_{pH}^{n_{h2}}} \text{ with } K_{pH} = 10^{-\frac{pH_{LL,h2} + pH_{UL,h2}}{2}} \text{ and } n_{h2} = \frac{3.0}{pH_{UL,h2} - pH_{LL,h2}} \quad (115)$$

3.2.3. Liquid phase equations

The influent liquid flow rate to the anaerobic digester is calculated as:

$$Q_{ad} = Q_{tu} + Q_{pu} \quad (116)$$

where Q_{tu} is the flow rate from the thickener underflow and Q_{pu} is the flow rate from the primary settler underflow.

The water-phase equations are written as:

Differential equations 117 to 128 (soluble matter)

$$\frac{dS_{su}}{dt} = \frac{Q_{ad}}{V_{ad,liq}}(S_{su,i} - S_{su}) + \rho_{2,ad} + (1 - f_{fa,li})\rho_{4,ad} - \rho_{5,ad} \quad (117)$$

$$\frac{dS_{aa}}{dt} = \frac{Q_{ad}}{V_{ad,liq}}(S_{aa,i} - S_{aa}) + \rho_{3,ad} - \rho_{6,ad} \quad (118)$$

$$\frac{dS_{fa}}{dt} = \frac{Q_{ad}}{V_{ad,liq}}\left(S_{fa,i} - S_{fa}\right) + f_{fa,li}\rho_{4,ad} - \rho_{7,ad} \quad (119)$$

$$\frac{dS_{va}}{dt} = \frac{Q_{ad}}{V_{ad,liq}}\left(S_{va,i} - S_{va}\right) + \left(1 - Y_{aa}\right)f_{va,aa}\rho_{6,ad} - \rho_{8,ad} \quad (120)$$

$$\frac{dS_{bu}}{dt} = \frac{Q_{ad}}{V_{ad,liq}}\left(S_{bu,i} - S_{su}\right) + \left(1 - Y_{su}\right)f_{bu,su}\rho_5 + \left(1 - Y_{aa}\right)f_{bu,aa}\rho_{6,ad} - \rho_{9,ad} \quad (121)$$

$$\frac{dS_{pro}}{dt} = \frac{Q_{ad}}{V_{ad,liq}}\left(S_{pro,i} - S_{pro}\right) + \left(1 - Y_{su}\right)f_{pro,su}\rho_{5,ad} + \left(1 - Y_{aa}\right)f_{pro,aa}\rho_{6,ad} + \left(1 - Y_{c4}\right)0.54\rho_{8,ad} - \rho_{10,ad} \quad (122)$$

$$\begin{aligned}\frac{dS_{ac}}{dt} = {} & \frac{Q_{ad}}{V_{ad,liq}}\left(S_{ac,i} - S_{ac}\right) + \left(1 - Y_{su}\right)f_{ac,su}\rho_{5,ad} + \left(1 - Y_{aa}\right)f_{ac,aa}\rho_{6,ad} + \left(1 - Y_{fa}\right)0.7\rho_{7,ad} \\ & + \left(1 - Y_{c4}\right)0.31\rho_{8,ad} + \left(1 - Y_{c4}\right)0.8\rho_{9,ad} + \left(1 - Y_{pro}\right)0.57\rho_{10,ad} - \rho_{11,ad}\end{aligned} \quad (123)$$

$$\begin{aligned}\frac{dS_{h2}}{dt} = {} & \frac{Q_{ad}}{V_{ad,liq}}\left(S_{h2,i} - S_{h2}\right) + \left(1 - Y_{su}\right)f_{h2,su}\rho_{5,ad} + \left(1 - Y_{aa}\right)f_{h2,aa}\rho_{6,ad} + \left(1 - Y_{fa}\right)0.3\rho_{7,ad} \\ & + \left(1 - Y_{c4}\right)0.15\rho_{8,ad} + \left(1 - Y_{c4}\right)0.2\rho_{9,ad} + \left(1 - Y_{pro}\right)0.43\rho_{10,ad} - \rho_{12,ad} - \rho_{T,8}\end{aligned} \quad (124)$$

$$\frac{dS_{ch4}}{dt} = \frac{Q_{ad}}{V_{ad,liq}}\left(S_{ch4,i} - S_{ch4}\right) + \left(1 - Y_{ac}\right)\rho_{11,ad} + \left(1 - Y_{h2}\right)\rho_{12,ad} - \rho_{T,9} \quad (125)$$

$$\frac{dS_{IC}}{dt} = \frac{Q_{ad}}{V_{ad,liq}}\left(S_{IC,i} - S_{IC}\right) - \sum_{j=1}^{19}\left(\sum_{k=1-9,11-24} C_k \nu_{k,j}\rho_{j,ad}\right) - \rho_{T,10} \quad (126)*$$

(127)

(128)

*More specifically, the sum in equation 126 is calculated as:

(129)

where

(130)

(131)

(132)

(133)

(134)

$$s_6 = -C_{aa} + (1 - Y_{aa})(f_{va,aa}C_{va} + f_{bu,aa}C_{bu} + f_{pro,aa}C_{pro} + f_{ac,aa}C_{ac}) + Y_{aa}C_{bac} \quad (135)$$

$$s_7 = -C_{fa} + (1 - Y_{fa})0.7C_{ac} + Y_{fa}C_{bac} \quad (136)$$

$$s_8 = -C_{va} + (1 - Y_{c4})0.54C_{pro} + (1 - Y_{c4})0.31C_{ac} + Y_{c4}C_{bac} \quad (137)$$

$$s_9 = -C_{bu} + (1 - Y_{c4})0.8C_{ac} + Y_{c4}C_{bac} \quad (138)$$

$$s_{10} = -C_{pro} + (1 - Y_{pro})0.57C_{ac} + Y_{pro}C_{bac} \quad (139)$$

$$s_{11} = -C_{ac} + (1 - Y_{ac})C_{ch4} + Y_{ac}C_{bac} \quad (140)$$

$$s_{12} = (1 - Y_{h2})C_{ch4} + Y_{h2}C_{bac} \quad (141)$$

$$s_{13} = -C_{bac} + C_{xc} \quad (142)$$

Differential equations 143 to 154 (particulate matter)

$$\frac{dX_c}{dt} = \frac{Q_{ad}}{V_{ad,liq}}(X_{c,i} - X_c) - \rho_{1,ad} + \rho_{13,ad} + \rho_{14,ad} + \rho_{15,ad} + \rho_{16,ad} + \rho_{17,ad} + \rho_{18,ad} + \rho_{19,ad} \quad (143)$$

$$\frac{dX_{ch}}{dt} = \frac{Q_{ad}}{V_{ad,liq}}(X_{ch,i} - X_{ch}) + f_{ch,xc}\rho_{1,ad} - \rho_{2,ad} \quad (144)$$

$$\frac{dX_{pr}}{dt} = \frac{Q_{ad}}{V_{ad,liq}}(X_{pr,i} - X_{pr}) + f_{pr,xc}\rho_{1,ad} - \rho_{3,ad} \quad (145)$$

$$\frac{dX_{li}}{dt} = \frac{Q_{ad}}{V_{ad,liq}}(X_{li,i} - X_{li}) + f_{li,xc}\rho_{1,ad} - \rho_{4,ad} \quad (146)$$

$$\frac{dX_{su}}{dt} = \frac{Q_{ad}}{V_{ad,liq}}(X_{su,i} - X_{su}) + Y_{su}\rho_{5,ad} - \rho_{13,ad} \quad (147)$$

(148)

(149)

(150)

(151)

(152)

(153)

$$\frac{dX_I}{dt} = \frac{Q_{ad}}{V_{ad,liq}}(X_{I,i} - X_I) + f_{xi\_xc}\rho_{1,ad} \quad (154)$$

Differential equations 155 and 156 (cations and anions)

$$\frac{dS_{cat^+}}{dt} = \frac{Q_{ad}}{V_{ad,liq}}\left(S_{cat^+,i} - S_{cat^+}\right) \quad (155)$$

$$\frac{dS_{an^-}}{dt} = \frac{Q_{ad}}{V_{ad,liq}}\left(S_{an^-,i} - S_{an^-}\right) \quad (156)$$

Differential equations 157 to 162 (ion states for ODE implementation)

$$\frac{dS_{va^-}}{dt} = -\rho_{A,4} \quad (157)$$

$$\frac{dS_{bu^-}}{dt} = -\rho_{A,5} \quad (158)$$

$$\frac{dS_{pro^-}}{dt} = -\rho_{A,6} \quad (159)$$

$$\frac{dS_{ac^-}}{dt} = -\rho_{A,7} \quad (160)$$

$$\frac{dS_{hco3^-}}{dt} = -\rho_{A,10} \quad (161)$$

$$\frac{dS_{nh3}}{dt} = -\rho_{A,11} \quad (162)$$

The algebraic equations involved for pH calculation are given as:

(163)

(164)

(165)

(166)

3.2.4. Gas phase equations

Differential equations 167 to 169 describe the fate of the gas phase components :

(167)

$$\frac{dS_{gas,ch4}}{dt} = -\frac{S_{gas,ch4}Q_{gas}}{V_{ad,gas}} + \rho_{T,9} \cdot \frac{V_{ad,liq}}{V_{ad,gas}} \quad (168)$$

(169)

The necessary algebraic equations are:

(170)

(171)

(172)

(173)

A problem with this way of calculating the gas flow rate is that is may give rise to numerical problems in the solution of the equations. Multiple steady states as well as numerical instability have been reported among users. An alternative way of calculating the gas flow rate is also given in Batstone *et al.* (2002):

(174)

with (175)

The alternative expression assumes an overpressure in the head space. Consequently, the flow rate is calculated at a higher pressure compared to the first expression. To compensate for this, the expression needs to rewritten into

$$Q_{gas} = k_p \left(P_{gas} - P_{atm} \right) \frac{P_{gas}}{P_{atm}} \quad (176)$$

to obtain the flow rate at atmospheric pressure. Although this compensation factor is included, the two expressions will not yield to identical results. Depending on the operational overpressure, which is a function of the value of parameter k_p (related to the friction in the gas outlet), the alternative expression results in a slightly smaller flow rate. The reason for this is that the liquid-gas transfer rates ($\rho_{T,8}$, $\rho_{T,9}$, $\rho_{T,10}$) will be different. A comparison of the two expressions when the same overpressure is applied shows very similar results (the relative error in the range of 1.10^{-5}). For BSM2, the alternative way (assuming an overpressure in the head space) of calculating the gas flow rate is used. Also note that if the physical or operational conditions of the digester model are changed (volume, load etc.), for example if applying the ADM1 as a stand-alone model outside the framework of BSM2, then the parameter k_p will have to be adjusted to achieve a reasonable overpressure in the head space.

3.3. ADM1 DAE implementation

It has been realized that the ODE implementation may be problematic for use in the BSM2 framework. The model must be able to handle dynamic inputs, time discrete and event-driven control actions as well as stochastic inputs or noise and still be sufficiently efficient and fast to allow for extensive simulations. The ADM1 is a very stiff system[1] with time constants ranging from fractions of a second to months. This makes the simulation of such a system challenging and in order to avoid excessively long simulation times, one needs to be somewhat creative when implementing the model.

Some solvers are so called stiff solvers and, consequently, capable of solving stiff systems. However, a problem common to all stiff solvers is the difficulty to handle dynamic input - including noise. The more stochastic or random an input variable behaves, the more problematic is the simulation using a stiff solver. The reason for this is that in stiff solvers, predictions of future state values are carried out. However, predictions of future state

[1] A system is called stiff, when the range of the time constants is large. This means that some of the system states react quickly whereas some others react sluggishly.

values affected by stochastic inputs will result in poor results, slowing down the solver by limiting its ability to use long integration steps. Simulation of the BSM2 is, thus, subject to the following dilemma. BSM2, which includes ASM1 and ADM1 models, is a very stiff system and, consequently, a stiff solver should be used. However, since BSM2 is a control simulation benchmark, noise must be included, calling for an explicit (i.e. non-stiff) solver.

In this section, the differential algebraic equation model implementation of ADM1 is presented. Two different DAE models are discussed: a model with algebraic pH (S_{H+}) calculations and a model with algebraic pH and S_{h2} calculations ($DAE_{pH, Sh2}$).

3.3.1 ODE and DAE systems

When the states of a system are described only by ordinary differential equations, the system is said to be an ODE system. If the system is stiff, it is sometimes possible to rewrite some of the system equations in order to omit the fastest states. The rationale for this is that from the slower state's point of view, the fast states can be considered instantaneous and possible to describe by algebraic equations. Such systems are normally referred to as differential algebraic equation (DAE) systems. By rewriting an ODE system to a DAE system, the stiffness can be decreased, allowing for explicit solvers to be used and for stochastic elements to be incorporated. The drawback is that the DAE system is only an approximation of the original system and the effect of this approximation must be considered and investigated for each specific simulation model.

3.3.2 Time constants in ADM1

As mentioned before, the ADM1 includes time constants in a wide range; from milliseconds for pH to weeks or months for the states describing various fractions of active biomass. Since most control actions affecting the anaerobic digester are fairly slow, it makes sense to investigate which fast states can be approximated by algebraic equations. In Batstone *et al.* (2002), it is suggested that the pH (S_{H+}) state is calculated by algebraic equations. However, this will only partially solve the stiffness problem. There are other fast states and a closer investigation suggests that the state describing hydrogen (S_{h2}) also needs to be approximated by an algebraic equation.

3.3.3. pH and S_{h2} solvers

As mentioned above, stiffness of the ADM1 can be reduced by approximating the differential equations of the pH and S_{h2} states by algebraic equations. Different solutions can be proposed to solve them.

An implicit algebraic equation for the pH calculation is given in (Batstone *et al.,* 2002). It has been suggested to calculate the S_{H+} and, consequently, the pH from the sum of all charges, which is supposed to be zero. The obtained implicit algebraic equations are non-linear and therefore can be solved only by an iterative numerical method. In the MATLAB-Simulink implementation of BSM2, the Newton-Raphson method used in Volcke (2006) for calculation of the pH and equilibrium concentrations was implemented. By using this method the new value of the unknown state is calculated at each iteration step *k* as:

$$S_{H^+,k+1} = S_{H^+,k} - \frac{E(S_{H^+,k})}{dE(S_{H^+})/dS_{H^+}\big|_{S_{H^+,k}}} \tag{177}$$

where $S_{H+,k}$ is the value of the state obtained from the previous iteration step and $E(S_{H+,k})$ is the value of the algebraic equation that has to be zero for the equilibrium, i.e.:

$$E(S_{H^+,k}) = S_{cat+,k} + S_{nh4+,k} + S_{H^+,k} - S_{hco3-,k} - \frac{S_{ac-,k}}{64} - \frac{S_{pr-,k}}{112} - \frac{S_{bu-,k}}{160} - \frac{S_{va-,k}}{208} - \frac{K_W}{S_{H^+,k}} - S_{an-,k} \tag{178}$$

The gradient of the algebraic equation, $dE(S_{H^+})/dS_{H^+}\big|_{S_{H^+,k}}$, is also needed for calculation of the new state value. Since this expression is rather complicated, it is not presented here. The iteration is repeated as long as $E(S_{H+,k})$ remains larger than the predefined tolerance value, which in the present case is set to 10^{-12}. Normally only two or three iterations are required to solve the equation at each time step.

In the FORTRAN implementation, a one-dimension optimization routine (Golden section) is used (Pons *et al*., 1983) to find the minimum of

$$O(S_{H^+,k}) = (S_{cat+,k} + S_{nh4+,k} + S_{H^+,k}) - \left(S_{hco3-,k} + \frac{S_{ac-,k}}{64} + \frac{S_{pr-,k}}{112} + \frac{S_{bu-,k}}{160} + \frac{S_{va-,k}}{208} + \frac{K_W}{S_{H^+,k}} + S_{an-,k}\right) \quad (179)$$

in the pH = 0 to 14 interval with a relative tolerance of 10^{-7}.

The differential equation for the S_{h2} state, explicitly given in the ODE implementation of this report, can be approximated by an algebraic equation in a similar way as was the case for the S_{H+} state, simply by setting its differential to zero (assuming fast dynamics). In the MATLAB-SIMULINK implementation, the iteration is carried out in the same way as for the S_{H+} calculation, this time using:

$$\begin{aligned} E(S_{h2,k}) = {} & \frac{Q_{ad}}{V_{ad,liq}}(S_{h2,i} - S_{h2,k}) + (1 - Y_{su})f_{h2,su}\rho_{5,ad} + (1 - Y_{aa})f_{h2,aa}\rho_{6,ad} + (1 - Y_{fa})0.3\rho_{7,ad} \\ & + (1 - Y_{c4})0.15\rho_{8,ad} + (1 - Y_{c4})0.2\rho_{9,ad} + (1 - Y_{pro})0.43\rho_{10,ad} - \rho_{12,ad} - \rho_{T,8} \end{aligned} \quad (180)$$

and the gradient of $E(S_{h2,k+1})$.

In the FORTRAN implementation, the minimum of

$$\begin{aligned} O(S_{h2,k}) = {} & \frac{Q_{ad}}{V_{ad,liq}}(S_{h2,i} - S_{h2,k}) + (1 - Y_{su})f_{h2,su}\rho_{5,ad} + (1 - Y_{aa})f_{h2,aa}\rho_{6,ad} + (1 - Y_{fa})0.3\rho_{7,ad} \\ & + (1 - Y_{c4})0.15\rho_{8,ad} + (1 - Y_{c4})0.2\rho_{9,ad} + (1 - Y_{pro})0.43\rho_{10,ad} - \rho_{12,ad} - \rho_{T,8} \end{aligned} \quad (181)$$

is searched for in the interval $[0, 10^{-5}]$ with a relative tolerance of 10^{-7}.

The expression of the gradient is quite complex. To obtain the gradients for the S_{H+} and S_{h2} equations, it is recommended that a tool for handling mathematics symbolically is used (e.g. Maple or Mathematica).

3.4. ADM1 benchmark model parameters

Tables 5 to 8 summarize the ADM1 model parameters used in BSM2.

4. MODELING OF THE PRIMARY CLARIFIER

The flow rate at the inlet of the primary clarifier (Q_{pi}) is given by:

$$Q_{pi} = Q_{plant} + Q_{to} + Q_{st,bypass} + Q_{st,out} \quad (182)$$

where Q_{plant} is the flow rate of raw wastewater which will be treated in the plant, Q_{to} is the overflow rate from the thickener, $Q_{st,bypass}$ is the flow rate bypassed from the sludge tank and $Q_{st,out}$ is the flow rate from the sludge tank. If the flow rate into the BSM2 system Q_i is larger than 60,000 $m^3.d^{-1}$

$$Q_{bypass} = Q_i - 60{,}000 \quad (183)$$

$$Q_{plant} = 60{,}000 \quad (184)$$

where Q_{bypass} the flow rate of raw wastewater which is bypassed.

For any influent fraction as well as temperature, the following equation holds:

$$Z_{pi} \cdot Q_{pi} = Z_{plant} \cdot Q_{plant} + Q_{to} \cdot Z_{to} + Q_{st,bypass} \cdot Z_{st,bypass} + Q_{st,out} \cdot Z_{st,out} \quad (185)$$

with

$$Z_{plant} = Z_i \quad (186)$$

The model proposed by Otterpohl and Freund (1992) and Otterpohl *et al.* (1994) can be described by one completely mixed tank and a separation of the effluent of the tank into a primary clarifier effluent and a primary sludge (Figure 4). The model description originally does not consider primary sludge. The formulas regarding primary sludge are added by simple mass balance considerations. Table 9 summarizes the variables and their assumed values.

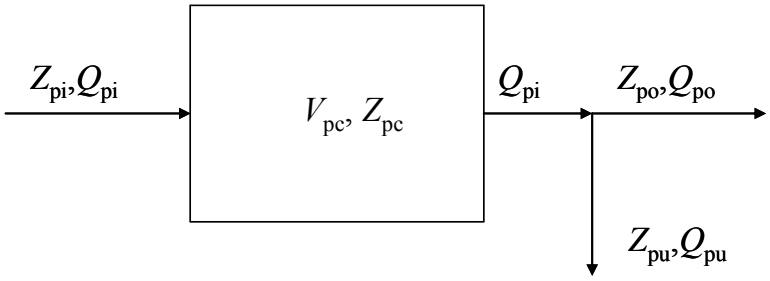


Figure 4: Primary clarifier model assumption

Table 5: Stoichimetric parameter values

Parameter	Value	Unit	Process(es)	Comment
$f_{SI,xc}$	0.1	-	1	
$f_{XI,xc}$	0.2	-	1	
$f_{ch,xc}$	0.2	-	1	
$f_{pr,xc}$	0.2	-	1	
$f_{li,xc}$	0.3	-	1	Note: $1-f_{ch,xc}-f_{pr,xc}-f_{SI,xc}-f_{li,xc}-f_{XI,xc} = 0$
N_{xc}	0.0376/14	c	1, 13-19	To maintain N balance for disintegration
N_I	0.06/14	kmole N.(kg COD)$^{-1}$	1	6% on weight basis in the ASM section
N_{aa}	0.007	kmole N.(kg COD)$^{-1}$	1, 6	
C_{xc}	0.02786	kmole C.(kg COD)$^{-1}$	1,13-19	C_{13} in Eq. (10)
C_{SI}	0.03	kmole C.(kg COD)$^{-1}$	1	C_{12} in Eq. (126)
C_{ch}	0.0313	kmole C.(kg COD)$^{-1}$	1,2	C_{14} in Eq. (126)
C_{pr}	0.03	kmole C.(kg COD)$^{-1}$	1,3	C_{15} in Eq. (126)
C_{li}	0.022	kmole C.(kg COD)$^{-1}$	1,4	C_{16} in Eq. (126)
C_{XI}	0.03	kmole C.(kg COD)$^{-1}$	1	C_{24} in Eq. (126)
C_{su}	0.0313	kmole C.(kg COD)$^{-1}$	2, 5	C_1 in Eq. (126)
C_{aa}	0.03	kmole C.(kg COD)$^{-1}$	3, 6	C_2 in Eq. (126)
$f_{fa,li}$	0.95	-	4	
C_{fa}	0.0217	kmole C.(kg COD)$^{-1}$	4, 7	C_3 in Eq. (126)
$f_{h2,su}$	0.19	-	5	
$f_{bu,su}$	0.13	-	5	
$f_{pro,su}$	0.27	-	5	
$f_{ac,su}$	0.41		5	
N_{bac}	0.08/14	kmole N.(kg COD)$^{-1}$	5-19	8% on weight basis in the ASM section
C_{bu}	0.025	kmole C.(kg COD)$^{-1}$	5, 6, 9	C_5 in Eq. (126)
C_{pro}	0.0268	kmole C.(kg COD)$^{-1}$	5, 6, 8, 10	C_6 in Eq. (126)
C_{ac}	0.0313	kmole C.(kg COD)$^{-1}$	5-11	C_7 in Eq. (126)
C_{bac}	0.0313	kmole C.(kg COD)$^{-1}$	5-19	$C_{17\text{-}23}$ in Eq. (126)
Y_{su}	0.1	-	5	kmole COD_X.(kg COD_S)$^{-1}$
$f_{h2,aa}$	0.06	-	6	
$f_{va,aa}$	0.23	-	6	
$f_{bu,aa}$	0.26	-	6	
$f_{pro,aa}$	0.05	-	6	
$f_{ac,aa}$	0.40	-	6	
C_{va}	0.024	kmole C.(kg COD)$^{-1}$	6, 8	C_4 in Eq. (126)
Y_{aa}	0.08	-	6	kmole COD_X.(kg COD_S)$^{-1}$
Y_{fa}	0.06	-	7	kmole COD_X.(kg COD_S)$^{-1}$
Y_{c4}	0.06	-	8, 9	kmole COD_X.(kg COD_S)$^{-1}$
Y_{pro}	0.04	-	10	kmole COD_X.(kg COD_S)$^{-1}$
C_{ch4}	0.0156	kmole C.(kg COD)$^{-1}$	11, 12	C_9 in Eq. (10)
Y_{ac}	0.05	-	11	kmole COD_X.(kg COD_S)$^{-1}$
Y_{h2}	0.06	-	12	kmole COD_X.(kg COD_S)$^{-1}$

Note that C_{h2} and C_{IN}, i.e. C_8 and C_{11}, are equal to zero in Eq. 126.

Table 6: Biochemical parameter values. The unit M is defined as kmole.m^{-3} according to Batstone *et al.* (2002)

Parameter	Value	Unit	Process(es)	Comment
k_{dis}	0.5	d^{-1}	1	
$k_{hyd,ch}$	10	d^{-1}	2	
$k_{hyd,pr}$	10	d^{-1}	3	
$k_{hyd,li}$	10	d^{-1}	4	
$K_{S,IN}$	1.10^{-4}	M	5-12	
$k_{m,su}$	30	d^{-1}	5	
$K_{S,su}$	0.5	kg COD.m^{-3}	5	
$pH_{UL,aa}$	5.5	-	5-10	in $I_{5\text{-}10}$
$pH_{LL,aa}$	4	-	5-10	in $I_{5\text{-}10}$
$k_{m,aa}$	50	d^{-1}	6	
$K_{S,aa}$	0.3	kg COD.m^{-3}	6	
$k_{m,fa}$	6	d^{-1}	7	
$K_{S,fa}$	0.4	kg COD.m^{-3}	7	
$K_{I,h2,fa}$	5.10^{-6}	kg COD.m^{-3}	7	in I_7
$k_{m,c4}$	20	d^{-1}	8, 9	
$K_{S,c4}$	0.2	kg COD.m^{-3}	8, 9	
$K_{I,h2,c4}$	1.10^{-5}	kg COD.m^{-3}	8, 9	in I_8 and I_9
$k_{m,pro}$	13	d^{-1}	10	
$K_{S,pro}$	0.1	kg COD.m^{-3}	10	
$K_{I,h2,pro}$	$3.5 \cdot 10^{-6}$	kg COD.m^{-3}	10	in I_8 and I_9
$k_{m,ac}$	8	d^{-1}	11	
$K_{S,ac}$	0.15	kg COD.m^{-3}	11	
$K_{I,NH3}$	0.0018	M	11	in I_{11}
$pH_{UL,ac}$	7	-	11	in I_{11}
$pH_{LL,ac}$	6	-	11	in I_{11}
$k_{m,h2}$	35	d^{-1}	12	
$K_{S,h2}$	7.10^{-6}	kg COD.m^{-3}	12	
$pH_{UL,h2}$	6	-	12	in I_{12}
$pH_{LL,h2}$	5	-	12	in I_{12}
$k_{dec,Xsu}$	0.02	d^{-1}	13	
$k_{dec,Xaa}$	0.02	d^{-1}	14	
$k_{dec,Xfa}$	0.02	d^{-1}	15	
$k_{dec,Xc4}$	0.02	d^{-1}	16	
$k_{dec,Xpro}$	0.02	d^{-1}	17	
$k_{dec,Xac}$	0.02	d^{-1}	18	
$k_{dec,Xh2}$	0.02	d^{-1}	13	

Table 7: Physiochemical parameter values

Parameter	Value	Unit	Comment
R	0.083145	$Bar.M^{-1}.K^{-1}$	
T_{base}	298.15	K	
T_{ad}	308.15	K	= 35°C
K_W	$10^{-14.0}$	M	$\approx 2.08 \cdot 10^{-14}$
$K_{a,va}$	$10^{-4.86}$	M	$\approx 1.38 \cdot 10^{-5}$
$K_{a,bu}$	$10^{-4.82}$	M	$\approx 1.5 \cdot 10^{-5}$
$K_{a,pro}$	$10^{-4.88}$	M	$\approx 1.32 \cdot 10^{-5}$
$K_{a,ac}$	$10^{-4.76}$	M	$\approx 1.74 \cdot 10^{-5}$
$K_{a,co2}$	$10^{-6.35}$	M	$\approx 4.94 \cdot 10^{-7}$
$K_{a,IN}$	$10^{-9.25}$	M	$\approx 1.11 \cdot 10^{-9}$
$k_{A,Bva}$	1.10^{10}	$M^{-1}.d^{-1}$	Set to be at least three orders of magnitude higher than the fastest time constant of the system
$k_{A,Bbu}$	1.10^{10}	$M^{-1}.d^{-1}$	
$k_{A,Bpro}$	1.10^{10}	$M^{-1}.d^{-1}$	
$k_{A,Bac}$	1.10^{10}	$M^{-1}.d^{-1}$	
$k_{A,Bco2}$	1.10^{10}	$M^{-1}.d^{-1}$	
$k_{A,BIN}$	1.10^{10}	$M^{-1}.d^{-1}$	
P_{atm}	1.013	bar	
$p_{gas,h2o}$		bar	≈ 0.0557
k_p	5.10^4	$m^3.d^{-1}.bar^{-1}$	
K_La	200	d^{-1}	
$K_{H,co2}$		$M_{liq}.bar^{-1}$	≈ 0.0271
$K_{H,ch4}$		$M_{liq}.bar^{-1}$	≈ 0.00116
$K_{H,h2}$		$M_{liq}.bar^{-1}$	$\approx 7.38 \cdot 10^{-4}$

Table 8: Physical parameter values

Parameter	Value	Unit
$V_{ad,liq}$	3400	m^3
$V_{ad,gas}$	100	m^3

For the evaluation of the concentrations within the tank simple CSTR formulas holds:

for all fractions k. (187)

Given the ”mean” fraction of particulate COD from overall COD:

(188)

the COD removal efficiency is calculated as:

[%] (189)

with the hydraulic retention time t_h (d) and a correction factor $f_{corr.}$ The parameter f_X is used as a constant parameter describing the mean value of the COD particulate to COD total ratio. For the calculation of the "mean" hydraulic retention time a first order low pass is used to calculate the mean influent flow rate:

(190)

with the "smoothing" time constant t_m=3/24 d. The hydraulic retention time is then calculated as

(191)

The removal efficiency with respect to the particulate COD is written as:

(192)

as the soluble COD is not effected. With this factor, the effluent concentrations calculates as:

(193)

with the factor

(194)

where the factor $f_{sx,k}$ is 0 for all soluble fractions, is 1 for all particulate fractions except X_S and is f_{XS} for X_S.

The primary sludge concentration follows from the mass balance:

(195)

For the primary sludge flow rate, a proportional flow rate to the influent flow rate is used:

$$Q_{pu}(t) = f_{PS}\ q(t) \quad \text{with } f_{PS} = 0.007 \tag{196}$$

Table 9: Main primary settler parameters

Parameter	Description	Value
f_{corr}	correction factor removal efficiency (tuning parameter)	0.65
f_k	effluent concentration factor	-
f_{PS}	ration of primary sludge flow rate to the influent flow rate	0.007
$f_{SX,k}$	structural factor defining the fraction of particulate matter of each fraction	-
f_X	ratio of particulate COD from total COD (mean value)	0.85
f_{XS}	particulate fraction of XS	0.5
$Z_{pc,k}$ (t)	concentrations in the mixing tank (CSTR)	
$Z_{pi,k}(t)$	influent concentrations	
$Z_{po,k}(t)$	effluent concentrations	
$Z_{pu,k}(t)$	primary sludge concentrations	
$\eta_{COD}(t)$	COD removal efficiency (total)	%
$\eta_{CODp}(t)$	COD removal efficiency (particulate)	%
$Q_{pi}(t)$	influent flow rate	$m^3.d^{-1}$
Q_m(t)	mean influent flow rate	$m^3.d^{-1}$
Q_{pu}(t)	primary sludge flow rate	$m^3.d^{-1}$
Q_{po}(t)	primary settler overflow, to the activated sludge section	$m^3.d^{-1}$
$t_h(t)$	hydraulic retention time	d
t_m	smoothing time constant for the q_m calculation	d
V_{pc}	volume of primary clarifier (900 m^3)	m^3

5. MODELING OF THE THICKENER AND DEWATERING UNIT

For simplicity, these two units are supposed to have an ideal behavior and have no volume.

5.1. Thickener

The thickener thickens the sludge wasted from the bottom of the clarifier prior to its mixing with the primary sludge from the primary clarifier and its digestion (Figure 5).

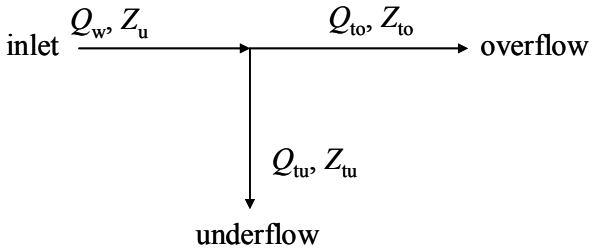


Figure 5: Thickener model assumption

The suspended solid concentration at the inlet of the thickener is $TSS_{sc,1}$ (Eq. 67). The percentage of suspended solids in the underflow of the thickener is p_{thick} (= 7%). The percentage of suspended solids removed is TSS_{rem} (= 98%). The thickening factor (f_{thick}) is calculated as:

(197)

Let (198)

Then, the thinning factor is equal to:

(199)

If the thickening factor is larger than 1, the variables in the underflow are calculated as:

For any particulate fraction:

For any soluble fraction and temperature: (200)

Underflow rate:

The variables in the overflow are calculated as:

For any particulate fraction:

For any soluble fraction and temperature: (201)

Overflow rate:

If the thickening factor is lower than 1, there is an error.

5.2. Dewatering unit

The dewatering unit thickens the digested sludge from the digester (flow rate Q_{ad}). The reject water is recycled to the inlet of the primary settler (Figure 6). The model of the dewatering unit is similar to the model of the thickener.

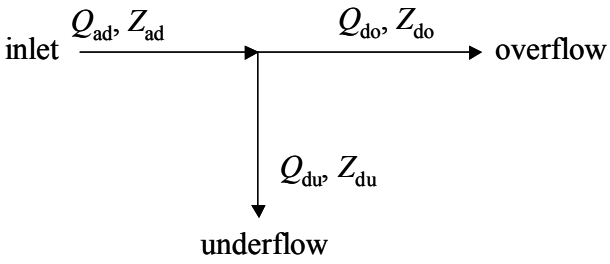


Figure 6: Dewatering unit model assumption

The suspended solid concentration at the inlet of the dewatering unit is:

(202)

where $X_{S,ad}$, $X_{P,ad}$, $X_{I,ad}$, $X_{B,H,ad}$ and $X_{B,A,ad}$ are outputs from the ADM-ASM interface (see §§7).

The percentage of suspended solids in the underflow of the thickener is p_{dewat} (= 28%). The percentage of suspended solids removed is TSS_{rem} (= 98%). The dewatering factor (f_{dewat}) is calculated as:

(203)

Let (204)

Then the dewatering factor is equal to:

(205)

If the dewatering factor is larger than 1, the variables in the underflow are calculated as:

For any particulate fraction: $Z_{du} = Z_{ad} \cdot f_{dewat}$

For any soluble fraction and temperature: (206)

Underflow rate:

The variables in the overflow are calculated as:

For any particulate fraction:

For any soluble fraction and temperature: (207)

Overflow rate:

If the dewatering factor is lower than 1, there is an error.

6. MODELING OF THE REJECT WATER STORAGE TANK

A storage tank has been set on the recycle line from the dewatering unit to the inlet of the primary clarifier, after the ADM to ASM interface. Its behaviour depends upon the flow rate from the dewatering unit, the available storage volume and the fate of the stored reject water. The reject water cannot be stored in the tank when the tank is full. In such a case, the reject water has to be recycled directly to the inlet of the primary clarifier until the storage tank has been emptied and can receive water again. Furthermore, a limit has been set for the reject water outflow rate from the tank ($Q_{st,set}$).

6.1 General definitions

The various notations used for the model are given in Table 10, in agreement with Figure 7. The height tolerance is used to define the minimal and maximal volumes that can be handled in the tank.

Table 10: Reject water storage tank variables

Definition	Notation
Total tank volume	$V_{st,total}$
Liquid volume in tank	V_{st}
Maximal liquid volume in tank	$V_{st,max}$
Minimal liquid volume in tank	$V_{st,min}$
Height	H_{st}
Area	A_{st}
Liquid flow rate from the dewatering unit overflow	Q_{do}
Liquid flow rate at the inlet of the storage tank	$Q_{st,in}$
Liquid flow rate at the outlet of the storage tank	$Q_{st,out}$
Liquid flow rate in the bypass of the storage tank	$Q_{st,bypass}$
Liquid flow rate setpoint at the outlet of the storage tank	$Q_{st,set}$

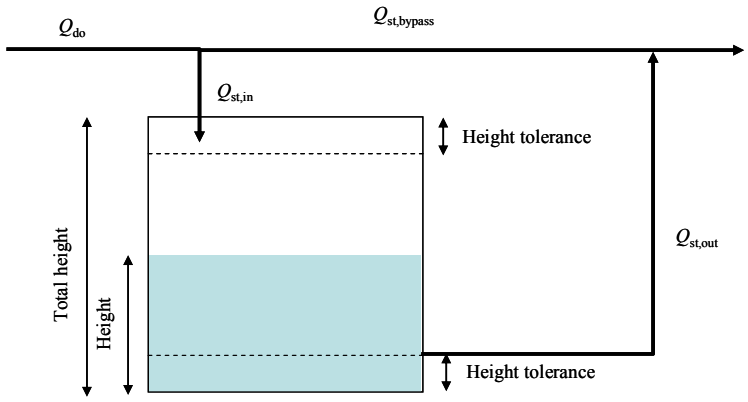


Figure 7: Storage tank

6.2. Storage tank behaviour

6.2.1. Variation of liquid volume

The following relations define the various flow rates needed for the mass balances.

When the liquid volume in the storage tank is between the minimal and the maximal values, the total reject water flow rate from the dewatering unit is admitted to the storage tank and the outflow rate is equal to the set point. No reject water is bypassed:

If ($V_{st} \leq V_{st,max}$ **and** $V_{st} \geq V_{st,min}$) **then**

$$Q_{st,in} = Q_{do}$$
$$Q_{st,out} = Q_{st,set} \quad (208)$$
$$Q_{st,bypass} = 0$$

When the liquid volume in the storage tank is larger than the maximal value and the reject water flow rate from the dewatering unit is larger than the outflow rate set point, the reject water from the dewatering unit is bypassed to the primary clarifier and the outflow rate from the storage tank is set to 0:

If ($V_{st} \geq V_{st,max}$ **and** $Q_{do} > Q_{st,set}$) **then**

$$Q_{st,in} = 0$$
$$Q_{st,out} = 0 \quad (209)$$
$$Q_{st,bypass} = Q_{do}$$

When the liquid volume in the storage tank is larger than the maximal value and the reject water flowrate from the dewatering unit is smaller than the ouflow rate set point, the reject water from the dewatering unit is fed to the storage tank. The outflow rate is equal to its set point.

If ($V_{st} \geq V_{st,max}$ **and** $Q_{do} \leq Q_{st,set}$) **then**

$$Q_{st,i} = Q_{do}$$
$$Q_{st,out} = Q_{st,set} \quad (210)$$
$$Q_{st,bypass} = 0$$

When the liquid volume in the storage tank is smaller than the minimal value, the reject water from the dewatering unit is fed into the storage tank and the outflow rate is set to 0:

If ($V_{st} \leq V_{st,min}$) **then**

$$Q_{st,in} = Q_{do}$$
$$Q_{st,out} = 0 \quad (211)$$
$$Q_{st,bypass} = 0$$

The variations of the storage tank volume are given as:

$$\frac{dV_{st}}{dt} = Q_{s,in} - Q_{st,out} \quad (212)$$

6.2.2. Variation of a concentration

The tank is assumed to be non reactive. Given a volume V_{st}, a influent flow rate $Q_{st,i}$, an influent concentration C_{do} and an effluent flow rate $Q_{st,out}$, the corresponding concentration C_{st} is calculated by integration of Equation (213).

$$\frac{dC_{st}}{dt} = \frac{1}{V_{st}}\left(Q_{st,i}C_{do} - Q_{st,i}C_{st}\right) = \frac{Q_{st,i}}{V_{st}}\left(C_{do} - C_{st}\right) \quad (213)$$

6.3. Implementation

The implementation is performed with the following parameters:
$V_{st,total}$= 160 m^3
$V_{stank,max} = 0.9 \cdot V_{st,total}$
$V_{stank,min} = 0.1 \cdot V_{st,total}$
$Q_{st,set} \leq 1500\ m^3.d^{-1}$
For initialization, the tank is half full ($V_{st} = 0.5 \cdot V_{st,total}$) when a sludge tank is considered in the system. It is full when a sludge tank is not considered. In such a case the flow from the dewatering unit overflow is bypassed directly to the primary clarifier.

7. ASM/ADM and ADM/ASM interfaces

The purpose of the ASM/ADM interface is to transform the state variables from the activated sludge section corresponding to the ASM1 formulation into state variables usable in the anaerobic digester corresponding to the ADM1 formulation. The opposite function is assigned to the ADM/ASM interface.

The ASM/ADM interface is applied after mixing the primary sludge from the underflow of the primary clarifier with the thickened secondary sludge wasted from the secondary clarifier. This means that for any ASM1 state variable (Z_{as}) to be transformed into an ADM1 state variable (Z_{ad}):

$$Z_{as} = \frac{Z_{pu} \cdot Q_{pu} + Z_{tu} \cdot Q_{tu}}{Q_{pu} + Q_{tu}} = \frac{Z_{pu} \cdot Q_{pu} + Z_{tu} \cdot Q_{tu}}{Q_{ad}} \quad (214)$$

The ADM/ASM interface is applied to transform any Z_{ad} state variable at the outlet of the digester (except pH and temperature) into a Z_{as} state variable, used in the dewatering unit. pH of ASM/ADM and ADM/ASM interfaces are identical to current (at every time step) pH in the digester. The temperature within the interfaces is equal to the digester temperature, i.e. 35°C. The temperature at the output of interface ADM/ASM at time *t* is equal to the temperature at the inlet of interface ASM/ADM at time *t*.

Both interfaces are built from sets of rules (Nopens *et al.*, 2009). To help the developer, the MATLAB and FORTRAN codes are given in Appendix A2. The following equations guarantee the charge balance (with T_{base} = 298.15 K (25 °C) and $T_{ad} = T_{base}$ + 10 K (35 °C):

$$\alpha_{ac}^{ch} = \alpha_{pro}^{ch} = \alpha_{bu}^{ch} = \alpha_{va}^{ch} = \frac{-1/C_i}{1 + 10^{pK_a - pH_{ad}}} \quad (215)$$

with pK_a=4.76, 4.88, 4.82, 4.86 resp. (T=25°C) with C_i respectively equal to 64, 112, 160, 208

$$\alpha_{IN}^{ch} = \frac{10^{pK_a - pH_{ad}}}{1 + 10^{pK_a - pH_{ad}}} \quad (216)$$

$$\text{with } pK_a = 9.25 - \log_{10}\left\{\exp\left[\frac{51965}{100 \cdot R}\left(\frac{1}{T_{base}} - \frac{1}{T_{ad}}\right)\right]\right\} \quad (217)$$

$$\alpha_{IC}^{ch} = \frac{-1}{1 + 10^{pK_a - pH_{ad}}} \quad (218)$$

$$\text{with } pK_a = 6.35 - \log_{10}\left\{\exp\left[\frac{7646}{100 \cdot R}\left(\frac{1}{T_{base}} - \frac{1}{T_{ad}}\right)\right]\right\} \quad (219)$$

(conversion from AS unit g $N.m^{-3}$ into AD unit kmole $N.m^{-3}$) (220)

(conversion from AS unit g $N.m^{-3}$ into AD unit kmole $N.m^{-3}$) (221)

(conversion from AS unit mole $HCO_3.m^{-3}$ into interface unit kmole $HCO_3.m^{-3}$) (222)

(223)

Note that $R = 0.083145$ bar.m^3. K^{-1}.$kmol^{-1}$

For the ASM/ADM interface:

(224)

It can be noted that based on the definition of the BSM TG ASM/ADM interface, the variables S_{ac}, S_{pro}, S_{bu} and S_{va} will always be zero, but they are added here to provide a general description. Special care must be taken to the S_{NO} concentration of the ASM input. The ASM/ADM interface immediately converts any influent nitrate to nitrogen gas (with an associated COD loss), i.e. a direct denitrification. However, in the charge balancing equation (the S_{IC} equation above) the true influent nitrate concentration from the ASM (prior to interface internal denitrification) should be used.

For the ADM/ASM interface:

(225)

It can be noted that based on the definition of the BSM TG ADM/ASM interface the variable S_{NO} will always be zero, but it is added here to provide a general description.

As the ADM1 fulfils absolutely the charge balance via S_{an} and S_{cat}, it is necessary to calculate S_{an} and S_{cat} to have a closed charge balance. The ASM1 does not calculate all ions and, thus, the charge balance is not absolutely closed, but of course all processes of the ASM1 respect the charge balance relatively.

To calculate S_{an} and S_{cat} of the influent to ADM1, the full charge balance should be used, i.e. it should include also the OH^- and H^+ ions:

(226)

with and leading to:

(227)

If the result is greater than zero, it is assumed:

(228)

and

(229)

If the result is smaller than zero,

(230)

and

(231)

Note that the above S_{an-} and S_{cat+} equations are only relevant in the ASM/ADM interface.

8. INFLUENT DATA

In BSM2 the evaluation of the plant performance is done on a full year (364 days). A dynamic stabilization period is required before evaluation. The file starts on 63 days before Jan 1st. The first 245 days (i.e. 63 + 182) serve for the stabilization under dynamic conditions.

The 609 days (63 + 182 + 364 = 609 days) dynamic file for BSM2 can be loaded from the CD. The structure of the file is as follows: time, S_I, S_S, X_I, X_S, $X_{B,H}$, $X_{B;A}$, X_P, S_O, S_{NO}, S_{NH}, S_{ND}, X_{ND}, S_{ALK}, TSS, Q_i, T and five dummy states for further extension. Some of these variables are plotted in Figure 8. TSS gives the total suspended solids in the influent according to:

$$TSS = 0.75\left(X_S + X_I + X_{B,H} + X_{B,A} + X_P\right) \tag{232}$$

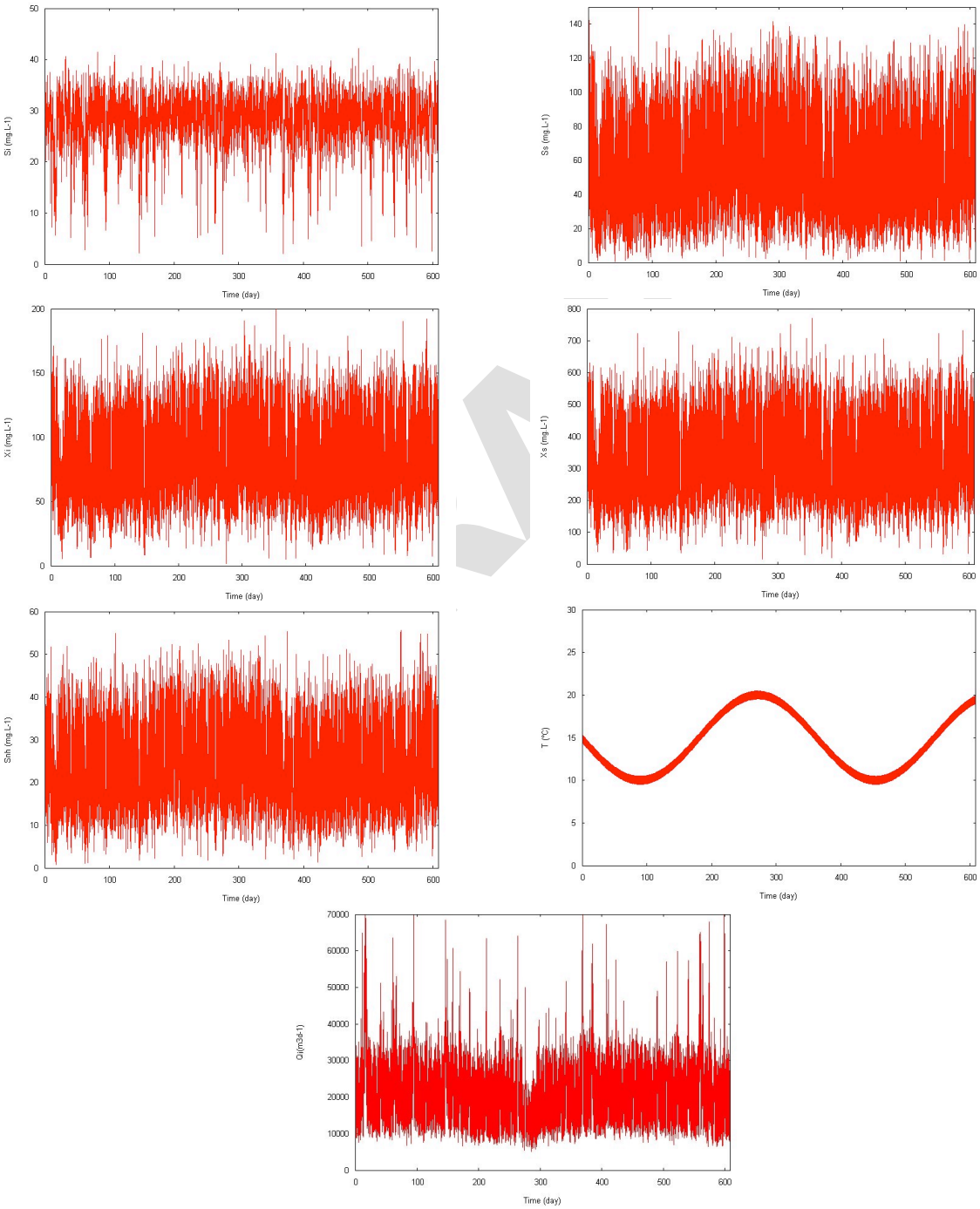


Figure 8: Some of the influent characteristics with respect to time: S_I, S_S, X_I, X_S, S_{NH}, T and Q_i

9. INITIALIZATION

Any initial values can be selected by the user. However, a 1000-day period of stabilization in closed-loop using constant inputs (Table 11) with no noise on the measurements has to be completed before using the influent file (609 days). Noise on measurements should be used with the dynamic files.

Table 11: Influent values for the stabilization period

Variable	Value	Unit
$S_{I,stab}$	27.22619062	g COD.m^{-3}
$S_{S,stab}$	58.17618568	g COD.m^{-3}
$X_{I,stab}$	92.49900106	g COD.m^{-3}
$X_{S,stab}$	363.943473	g COD.m^{-3}
$X_{B,H,stab}$	50.68328815	g COD.m^{-3}
$X_{B,A,stab}$	0	g COD.m^{-3}
$X_{P,stab}$	0	g COD.m^{-3}
$S_{O,stab}$	0	g (-COD)/m^3
$S_{NO,stab}$	0	g N.m^{-3}
$S_{NH,stab}$	23.85946563	g N.m^{-3}
$S_{ND,stab}$	5.651606031	g N.m^{-3}
$X_{ND,stab}$	16.12981606	g N.m^{-3}
$S_{ALK,stab}$	7	mole.m^{-3}
TSS_{stab}	380.3443217	g .m^{-3}
$Q_{i,stab}$	20648.36121	m^3.d^{-1}
T_{stab}	14.85808006	°C

The following operation conditions are applied during the stabilization period:

- Internal recycle flow rate Q_{int} = 61,944 m^3.d^{-1}
- External recylce flowrate Q_r = 20,648 m^3.d^{-1}
- Wastage flowrate Q_w = 300 m^3.d^{-1}
- External carbon flowrate in 1st anoxic reactor Q_{EC1} = 2 m^3.d^{-1}
- External carbon concentration: 400,000 g COD.m^{-3}
- Oxygen transfer coefficients : $K_La_3 = K_La_4$ = 120 d^{-1} and K_La_5 = 60 d^{-1}
- Flow rate from the reject water storage tank $Q_{st,set}$ = 0.

Appendix A3 summarizes the steady-state results obtained under these conditions.

10. EVALUATION

For evaluation of the simulation results over a fixed period of time (t_{obs}= t_f - t_0), average values are to be calculated as follows (The user should be aware that all the integrals for performance assessment are calculated by rectangular integration with a time step of 15 min):

- Flow rate (m^3.d^{-1}): (233)

- Concentration for compound Z_k (mass.m^{-3}) in flow Q must be flow proportional:

(234)

11. SET-UP OF A DEFAULT CONTROLLER

A default controller is proposed so the closed-loop simulation and the implementation of the evaluation criteria can be tested before the user implements his/her own control strategy. The primary control objective for the default strategies is to maintain the dissolved oxygen concentration in the fifth compartment at a predetermined set point value (2 g (-COD).m^{-3}) by manipulation of the oxygen transfer coefficient in the fourth reactor in such a way that: $K_La_3 = K_La_4$; $K_La_5 = K_La_4/2$. Actuators models are used for the three oxygen transfer coefficients. The modeling principles of the sensors are given in Section 13 of this document. Furthermore, external carbon addition rate is 2.0 m^3.d^{-1}. Finally, two different wastage flow rates are imposed dependent on time of the year (Table 12), assuming day 0 is at the start of the 609 days period. For this reason a first-order filter (time constant = 0.0001 day) is added to simulate the response of the wastage pump.

Table 12: Wastage flowrate in function of time

Time (d)	Q_w (m^3.d^{-1})
$0 \le t < 182$	300
$182 \le t < 364$	450
$364 \le t < 546$	300
$546 \le t < 608$	450

Appendices A4 to A6 summarize the results obtained in dynamic conditions in open loop (A4), in closed-loop with ideal sensors and actuators (A5) and in closed-loop with realistic sensors and actuators (A6).

11.1. Controller variables

For the dissolved oxygen control in second aerated compartment, the probe is assumed to be of class A with a measurement range of 0 to 10 g (-COD).m^{-3} and a measurement noise of 0.25 g (-COD).m^{-3}. The manipulated variable is the oxygen transfer coefficient, K_La_4.

Constraints are applied on recirculation flows. In the test case, Q_{int} is maintained constant and is set at $Q_{i,stab}$. The external recycle flow rate Q_r is maintained constant and is set to $Q_r = Q_{i,stab}$. There are also constraints on oxygen transfer in compartment 4: $K_La_4 = 0$ to 10 d^{-1}.

11.2. Controller type

The suggested controller is of the PI type. Its performance is assessed by:
- the Integral of Absolute Error (*IAE*)

$$IAE = \int_{t_0}^{t_f} |e| \cdot dt \tag{235}$$

where e is the error:

$$e = Z^{setpoint} - Z^{meas} \tag{236}$$

- the Integral of Squared Error (*ISE*)

$$ISE = \int_{t_0}^{t_f} e^2 \cdot dt \tag{237}$$

- the maximal deviation from set point:

$$Dev^{max} = \max\{|e|\} \tag{238}$$

- the error variance:

$$Var(e) = \overline{e^2} - (\overline{e})^2 \tag{239}$$

with

$$\overline{e} = \frac{\int_{t_0}^{t_f} e \cdot dt}{t_{obs}} \quad (240)$$

$$\overline{e^2} = \frac{\int_{t_0}^{t_f} e^2 \cdot dt}{t_{obs}} \quad (241)$$

- the variance of manipulated variable (u_i) variations:

$$Var(\Delta u) = \overline{\Delta u^2} - \left(\overline{\Delta u}\right)^2 \quad (242)$$

with

$$\Delta u = \left| u(t + dt) - u(t) \right| \quad (243)$$

$$\overline{\Delta u} = \frac{\int_{t_0}^{t_f} \Delta u \cdot dt}{t_{obs}} \quad (244)$$

and $$\overline{\Delta u^2} = \frac{\int_{t_0}^{t_f} \Delta u^2 \cdot dt}{t_{obs}} \quad (245)$$

These criteria can be generalized for any controller implemented on the benchmark.

12. PERFORMANCE ASSESSMENT

The flow-weighted average values of the effluent concentrations over the evaluation should obey the limits given in Table 13. Total nitrogen (N_{tot}) is calculated as the sum of $S_{NO,e}$ and $S_{NKj,e}$, where S_{NKj} is the Kjeldahl nitrogen concentration.

Table 13: Effluent quality limits

Variable	Value
N_{tot}	<18 g N.m^{-3}
COD_{tot}	<100 g COD.m^{-3}
S_{NH}	<4 g N.m^{-3}
TSS	<30 g SS.m^{-3}
BOD_5	<10 g BOD.m^{-3}

The *percentage of time* the effluent limits are not met must be reported, as well as the *number of violations*. The number of violations is defined as the *number of crossings* of the limit (from below to above the limit).

The performance assessment is made at two levels.

- The **first level** concerns the local control loops, assessed by *IAE* (Integral of the Absolute Error) and *ISE* (Integral of the Squared Error) criteria, by maximal deviation from set points, and by error variance. Basically, this serves as a proof that the proposed control strategy has been applied properly.
- The **second level** provides measures for the effect of the control strategy as such on plant performance and it can be divided into four sub-levels:

  - the **effluent quality**: levies or fines are to be paid due to the discharge of pollution in the receiving water bodies. The Effluent Quality Index (*EQI*) (kg pollution unit.d^{-1}) is averaged over the period of observation t_{obs} (d) (i.e. 364 days = 1 year) based on a weighting of the effluent loads of compounds that have a major influence on the quality of the receiving water and that are usually included in regional legislation. It is defined as:

$$EQI = \frac{1}{t_{obs} \cdot 1000} \int_{t=245\,days}^{t=609\,days} \left(\begin{matrix} B_{TSS} \cdot TSS_e(t) + B_{COD} \cdot COD_e(t) + B_{NKj} \cdot S_{NKj,e}(t) \\ + B_{NO} \cdot S_{NO,e}(t) + B_{BOD5} \cdot BOD_e(t) \end{matrix} \right) Q_e(t) \cdot dt \quad (246)$$

where

$$S_{NKj,e} = S_{NH,e} + S_{ND,e} + X_{ND,e} + i_{XB}\left(X_{B,H,e} + X_{X,A,e}\right) + i_{XP}\left(X_{P,e} + X_{i,e}\right) \quad (247)$$

$$TSS_e = 0.75 \cdot \left(X_{S,e} + X_{I,e} + X_{B,H,e} + X_{B,A,e} + X_{P,e}\right) \quad (248)$$

$$BOD_{5,e} = 0.25 \cdot \left(S_{S,e} + X_{S,e} + \left(1 - f_P\right) \cdot \left(X_{B,H,e} + X_{B,A,e}\right)\right) \quad (249)$$

$$COD_e = S_{S,e} + S_{I,e} + X_{S,e} + X_{I,e} + X_{B,H,e} + X_{B,A,e} + X_{P,e} \quad (250)$$

$$Q_e = Q_{sc,e} + Q_{bypass} \quad (251)$$

and the B_i are weighting factors for the different types of pollution to convert them into pollution units (Table 14). The concentrations are to be expressed in g.m^{-3}. The values for B_i have been deduced from Vanrolleghem *et al.* (1996).

Table 14: B_i values

Factor	B_{TSS}	B_{COD}	B_{NKj}	B_{NO}	B_{BOD5}
Value (g pollution unit.g^{-1})	2	1	30	10	2

The 95% percentiles of the effluent ammonia ($S_{NH,e95}$), effluent total nitrogen ($N_{tot,e95}$) and total suspended solids (TSS_{e95}) have to be shown as well. These percentiles represent the S_{NH}, N_{tot} and TSS effluent concentrations that are exceeded 5% of the time.

- the **cost factors for operation**

- the **sludge production to be disposed** *(SP)* (kg.d^{-1})

The sludge production, *SP*, is calculated from the total solid flow from wastage and the solids accumulated in the system over the period of time considered (the last 364 days of the weather file).

$$SP = \frac{1}{t_{obs}} \left(TSS(609\ days) - TSS(245\ days) + 0.75 \cdot \int_{t=245\,days}^{t=609\,days} \left(X_{S,w} + X_{I,w} + X_{B,H,w} + X_{B,A,w}\right) \cdot Q_w(t) \cdot dt \right) \quad (252)$$

where $TSS\,(t)$ is the amount of solids in the system at time t, i.e.

$$TSS(t) = TSS_{as}(t) + TSS_{sc}(t) \quad (253)$$

TSS_{as} and TSS_{sc} are given respectively by equations 60 and 62.

- the **total sludge production** *(SP_{total})* (kg.d^{-1}) takes into account the sludge to be disposed and the sludge lost at the weir:

$$SP_{total} = SP + \frac{0.75}{t_{obs}} \int_{t=245\,days}^{t=609\,days} \left(X_{S,e} + X_{I,e} + X_{B,H,e} + X_{B,A,e} + X_{P,e}\right) \cdot Q_e(t)\,dt \quad (254)$$

- the **aeration energy** (*AE*) (kWh.d^{-1}) and the **pumping energy** (*PE*) (kWh.d^{-1}) (internal and external flow recycle pumps).

The pumping energy depends on how the various tanks can be arranged on the available space. Considering the state-of-the-art design rules an arrangement with two parallel lines, similar to the one shown in Appendix 1, can be proposed for the activated ludge section of BSM2. In BSM2 the pumping energy is calculated as:

$$PE = \frac{1}{t_{obs}} \int_{t=245\,days}^{t=609\,days} \left(0.004 \cdot Q_{int}(t) + 0.008 \cdot Q_r(t) + 0.05 \cdot Q_w(t) + 0.075 \cdot Q_{pu} + 0.06 \cdot Q_{du} + 0.004 \cdot Q_{du}\right) dt \quad (255)$$

with the flow rates expressed in m^3.d^{-1}.

The aeration energy *AE* should take into account the plant peculiarities (type of diffuser, bubble size, depth of submersion, etc.) and is calculated from the K_La according to the following relation, valid for Degrémont DP230 porous disks at an immersion depth of 4 m:

$$AE = \frac{S_O^{sat}}{t_{obs} \cdot 1.8 \cdot 1000} \int_{t=245\,days}^{t=609\,days} \sum_{k=1}^{5} V_{as,k} \cdot K_L a_k(t)\,dt \quad (256)$$

with K_La given in d^{-1} and k referring to the compartment number.

- the **consumption of external carbon source** (*EC*) (kg COD.d^{-1}) that could be added to improve denitrification (see Section 7 on control and handles)

$$EC = \frac{COD_{EC}}{t_{obs} \cdot 1000} \int_{t=245\,days}^{t=609\,days} \left(\sum_{k=1}^{k=n} Q_{EC,k} \right) \cdot dt \tag{257}$$

where $Q_{EC,k}$ is the flow rate of external carbon added to compartment *k* and COD_{EC} = 400,000 g COD.m^{-3} is the concentration of readily biodegradable substrate in the external carbon source.

- the **mixing energy** (*ME*) (kWh.d^{-1})

The compartments in anoxic state should be mixed to avoid settling. Mixing energy is a function of the compartment volume.

$$ME = \frac{24}{t_{obs}} \int_{t=245\,days}^{t=609\,days} \sum_{k=1}^{k=5} \begin{bmatrix} 0.005 \cdot V_{as,k} \;\; if \;\; K_L a_k(t) < 20 \;\; d^{-1} \\ 0 \;\; otherwise \end{bmatrix} \cdot dt \tag{258}$$

- the **methane production** (MET_{prod}) (kg.d^{-1})

$$MET_{prod} = \frac{16 \cdot P_{atm}}{RT_{ad} t_{obs}} \int_{t=245\,days}^{t=609\,days} \frac{Q_{gas}(t) \cdot p_{gas,ch4}(t)}{P_{gas}(t)} dt \tag{259}$$

- the **heating energy** (*HE*) (kWh.d^{-1})

It is necessary to heat the digester influent to the digester operating temperature (T_{ad}):

$$HE = \frac{1000 \cdot 4.186}{86400 \cdot t_{obs}} \int_{t=245\,days}^{t=609\,days} (T_{ad} - T_{ad,i}) Q_{ad}(t) dt \tag{260}$$

$$\text{with } T_{ad,i} = \frac{T_{pu} \cdot Q_{pu} + T_{tu} \cdot Q_{tu}}{Q_{ad}} + 273.15 \tag{261}$$

assuming T_{pu} and T_{tu} are given in °C and T_{ad} in K.

The methane produced in the digester is used to generate the necessary heat energy. The net heating energy is calculated as:

$$HE^{net} = \max(0., HE - 7. \cdot MET_{prod}) \tag{262}$$

- controller output variations

The maximum values and the variance of the manipulated variables variations should be given. This will provide an indication on peak loads and the wear of the pumps and aeration devices.

Furthermore, in case the user applies other influent data files than the one defined with BSM2 an Influent Quality Index (*IQI*) index is proposed to compare the influent qualities:

$$IQI = \frac{1}{t_{obs} \cdot 1000} \int_{t=245\,days}^{t=609\,days} \begin{pmatrix} B_{TSS} \cdot TSS_i(t) + B_{COD} \cdot COD_i(t) + B_{NKj} \cdot S_{NKj,i}(t) \\ + B_{NO} \cdot S_{NO,i}(t) + B_{BOD5} \cdot BOD_{5,i}(t) \end{pmatrix} Q_i(t) dt \tag{263}$$

with:

$$S_{Nkj,i} = S_{NH,i} + S_{ND,i} + X_{ND,i} + i_{XB}(X_{B,H,i} + X_{X,A,i}) + i_{XP}(X_{P,i} + X_{i,i}) \tag{264}$$

$$TSS_i = 0.75 \cdot (X_{S,i} + X_{I,i} + X_{B,H,i} + X_{B,A,i} + X_{P,i}) \tag{265}$$

$$BOD_{5,i} = 0.65 \cdot (S_{S,i} + X_{S,i} + (1 - f_P) \cdot (X_{B,H,i} + X_{B,A,i})) \tag{266}$$

$$COD_i = S_{S,i} + S_{I,i} + X_{S,i} + X_{I,i} + X_{B,H,i} + X_{B,A,i} + X_{P,i} \tag{267}$$

- Finally an **Overall Cost Index** (*OCI*) is calculated:

$$OCI = AE + PE + 3 \cdot SP + 3 \cdot EC + ME - 6 \cdot MET_{prod} + HE_{net} \tag{268}$$

13. SENSORS AND CONTROL HANDLES

13.1. Introduction

To test your own control strategy on the BSM2 plant, appropriate sensors and actuators must be selected. To avoid unrealistic control behaviour, the dynamic behaviour of sensors and actuators (control handles) as well as additional measurement noise must be considered. To allow for a wide range of different strategies to be tested (within the confinement of the physical plant layout), a significant number of sensors and control handles are available. Their mathematical descriptions focus on simplicity rather than completely accurate reproductions of their true behaviour.

The principle for any good control strategy implies that the number of sensors and control actions should be minimised within the framework of the selected control strategy, due to the investment and maintenance costs, etc (Rieger *et al*., 2003).

For initialisation purposes, first test of control concepts, or evaluation of the theoretical potential of control options, it is of course a valid option to use ideal sensors (no noise, no delay). For internal flows (e.g. return sludge, internal recycle), which are basically control handles, it can be assumed that the flow rates are known or can be measured without errors and delays. For such an ideal sensor, no specific sensor model is required. But the usage of ideal sensors should be reported when discussing a specific control strategy.

13.2. Sensors

The aim of the sensor classification is to describe different sensor types but also to limit the number of sensor classes in order to ease the comparison of the simulation results. The benchmark deals with control strategies, therefore only a few related criterions are used and only one minimal measuring interval of 5 minutes is taken into account. It is not intended to define a user configurable class, since this would make it difficult to compare different benchmark studies. Should it nevertheless be impossible to choose a class, the benchmark model user is requested to describe the specific sensor in detail.

The main parameter to describe the sensor dynamics of the sensor classes is the “Response time”. This parameter is defined in an ISO norm (ISO 2003) and characterises the sensor dynamics based on a step response as presented in Figure 9.

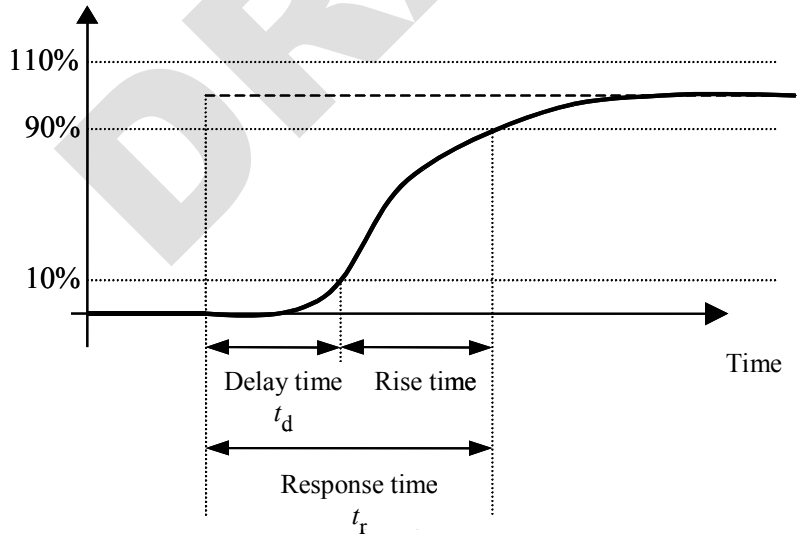


Figure 9: Definition of response time

In the norm the response time is the sum of the delay and the rise (or fall) time. The delay is defined as the time to reach 10% of the final value of a step response (t_d). Thus, the delay time in this context is not exactly the same as a transport delay time or dead-time defined in control engineering. The overall time to reach (and not to leave) a band from 90% - 110% of the final value of the step response is introduced as response time (here t_r). To describe the dynamics of a sensor it is assumed that the two values delay time and response time (as defined by Figure 6) are given.

For the definition of the benchmark sensor classes a response time (t_r) is proposed. The six sensor classes are shown in Table 14 and a list of typical sensors is provided in Table 15.

Table 14: Sensor classes. A measuring interval equal to 0 means continuous measurement

Sensor classes	Response time (t_r) [min]	Measuring interval (t_i) [min]	Examples
Class A	1	0	Ion sensitive, optical without filtration
Class B_0	10	0	Gas sensitive + fast filtration
Class B_1	10	5	Photometric + fast filtration
Class C_0	20	0	Gas-sensitive + slow filtration
Class C_1	20	5	Photometric + slow filtration or sedimentation
Class D	30	30	Photometric or titrimetric for total components

The response time includes the whole system with filtration unit and measuring system. Class A is a more or less ideal sensor; the response time of 1 minute is chosen in order to prevent unrealistic control applications. Class B contains mainly classical on-line analyzers with a fast filtration and short sample loops. In Class C, analyzers with a slow filtration or sedimentation unit are described. Class D includes all batch measurements like respirometer and sensors for total components. To take into account continuously and discontinuously measuring sensors, the classes B and C are divided into two subclasses. Five minutes is selected as the measuring interval, which is a typical minimum value for photometric analyzers. Longer intervals are not useful for control actions and are therefore neglected.

Additional to choosing the sensor class, the user has to define the measuring range for each sensor. Depending on the chosen measurement range, the standard deviation is assumed to be 2.5% of the maximum measurement value (see sensor model description).

Real measurement signals always include measurement noise, which can lead to unwanted control actions or slow down the reaction. Therefore, noise is included in the sensor model. The idea is not to model noise exactly, but to take into account some of its effects. In order to get comparable benchmark simulation results, the noise signal is defined. Choice of a random signal would have required running each benchmark simulation a large number of times in order to eliminate the influence of the random signal. The noise signal is chosen with a standard deviation of 1, which is multiplied with the defined noise level (2.5% of the maximum measurement value). The noise is white zero-mean normally distributed noise. Other types of noise would be too specific and the sensors within one class would not be comparable.

As an illustration, oxygen and nitrate sensors, which can be used in the activated sludge section, can very easily be described as:
- oxygen sensor: Class A, measurement range: 0-10 g (-COD).m^{-3}, measurement noise δ = 0.25 g (-COD).m^{-3}.
- nitrate sensor: Class B_0 with a measurement range 0-20 g N.$m^{-3,}$ measurement noise δ = 0.5 g N.m^{-3}.

13.3. Sensor model description

To ensure identical implementation and behaviour of the sensor models, it is necessary to describe the model in detail. The following description is the result of a Simulink implementation and takes into account a number of performance issues which are similar for most of the simulation systems.

The proposed sensor classes contain a set of continuous (A, B_0, C_0) and time-discrete sensor models (B_1, C_1, D). Continuous models are preferred to time-discrete ones for implementing the continuous sensors for performance reasons. The discontinuous sensors B_1 and C_1 are modelled in a similar way but include an output sample and hold function. Sensor class D is modelled only in discrete form.

13.3.1. Continuously measuring sensors

For the sensor classes A, B_0 and C_0 the approach is shown in Figure 10:

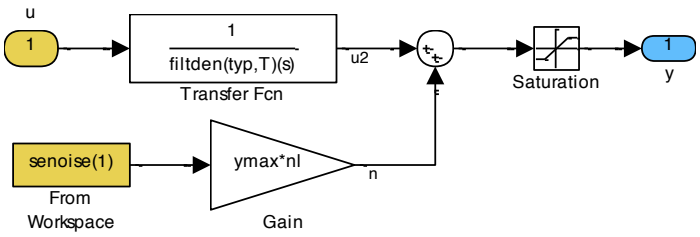


Figure 10: Simulink model of sensor class A, B_0 and C_0

The original sensor signal u is transformed by a linear transfer function (block Transfer Fcn). This transfer function is used to implement the expected time response of the sensor. Real time behaviour of sensors is typically a combination of transport+delay time behaviour (or dead time) caused by sample transport and preparation and a first or higher order dynamics (time constants) caused by different reasons, e.g. a mixing tank.

To have a sensor model with the same response time, a series of equal first order delay transfer functions is assumed. The number of first order transfer functions in series (n) determines the ratio of delay time and response time (as defined in Figure 9). Table 16 shows the parameters for the response-time modelling (see specific sensor model) of the continuously operating sensors.

For the sensor class A a response time (t_r) of 1 min and a system order of n = 2 is suggested. The assumed transfer function is:

(269)

The problem is to find τ such as t_r = 1 min, using either Simulink or the time-domain function:

(270)

With $\tau = 0.257 = t_r / 3.89$, the ratio of the delay time to the rise time ($R_{td/tr}$) is equal to 0.133. Thus the transfer function is only a small fraction of the response time as typical for this sensor class.

For the sensor classes B and C, a system order of n = 8 is assumed to mimic the behaviour of the sensors. For class B a response time of 10 min and for class C of 20 min is selected. The transfer function is

(271)

with $\tau = t_r / 11.7724$.

This will lead to a ratio of the delay time to the response time equal to 0.392. In this case, the delay time is approximately 40% of the response time. This is assumed to consider the significant effect of the transport of the sample for the sensor classes B and C. The step responses for the classes A, B_0 and C_0 are presented in Figure 11.

Table 15: Typical sensor characteristics within the proposed classification scheme

Measured variable	Sensor types	t_d (min)	t_i (min)
MLSS (g.m^{-3})	A	0	0
Turbidity (FNU or g TSS.m^{-3})	A	0	0
S_{NH4} (ion sensitive)	A	0	0
S_{NOx} (ion sensitive)	A	0	0
S_{NOx} (UV)	A	0	0
C_{COD}, S_{COD} (UV/Vis)	A	0	0
Flow rate (m^3.d^{-1})	A	0	0
Water level (m)	A	0	0
Temperature (°C)	A	0	0
pH	A	0	0
S_O (g (-COD).m^{-3})	A	0	0
Sludge blanket heigth (m)	A	0	0
S_{NH4} (gas sensitive + normal filtration)	B_0	10	0
S_{NOx} (UV + normal filtration)	B_0	10	0
S_{NH4} (photometric + normal filtration)	B_1	10	5
S_{NO3} (photometric + normal filtration)	B_1	10	5
S_{NO2} (photometric + normal filtration)	B_1	10	5
S_{PO4} (photometric + normal filtration)	B_1	10	5
S_{NH4} (gassensitive + slow filtration or sedimentation)	C_0	20	0
S_{NOx} (UV + slow filtration or sedimentation)	C_0	20	0
S_{NH4} (photometric + slow filtration or sedimentation)	C_1	20	5
S_{NO3} (photometric + slow filtration or sedimentation)	C_1	20	5
S_{NO2} (photometric + slow filtration or sedimentation)	C_1	20	5
S_{PO4} (photometric + slow filtration or sedimentation)	C_1	20	5
C_{COD} (thermal chemical oxidation + photometric)	D	30	30
TOC (thermal oxidation + IR detector)	D	30	30
C_N (thermal oxidation + IR detector or chemoluminescence detector)	D	30	30
C_P (thermal chemical oxidation + photometric)	D	30	30
Respirometer	D	30	30
Titration biosensor (alkalinity)	D	30	30

Table 16: Parameters for response time modelling

Sensor class	t_r (min)	n	τ (min)	$R_{td/tr}$
A	1	2	0.257	0.133
B_0	10	8	0.849	0.392
C_0	20	8	1.699	0.392

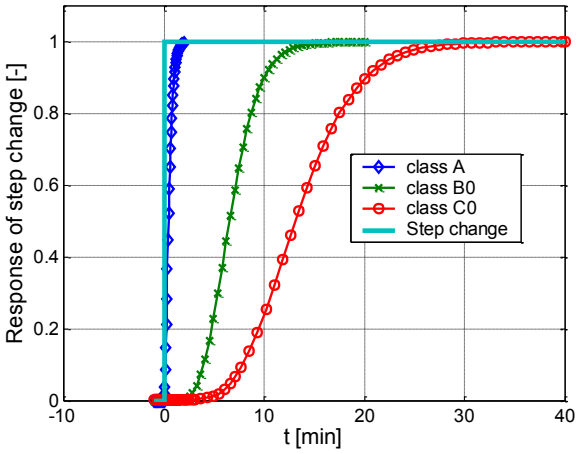


Figure 11: Step response of classes A, B_0, C_0.

The noise is modelled with a constant noise level *nl*. In the Simulink model presented in Figure 10, the noise signal (white noise with a standard deviation δ=1) is multiplied by the noise level *nl* and the maximum value of the measurement interval y_{max}. A normal distributed (standard deviation 1), frequency limited noise signal should be used and created by the user. The signal could be created using a sample time of 1 min and be interpolated using linear interpolation to provide a continuous noise signal. Using the sample time of 1 min together with the linear interpolation will limit the frequency spectrum of the noise (cut-of of high frequencies - pink noise. The noise is added to the delayed measurement signal and limited to the measurement interval (0, y_{max}).

13.3.2. Discontinuously measuring sensors

Sensor classes B_1, C_1 and D are operated discontinuously using a sampling interval t_i. An example of an implementation using a Simulink model is presented in Figures 12 and 13. The implementation is similar to that used in the model for the continuously measuring sensors but includes an additional output sample and hold function.

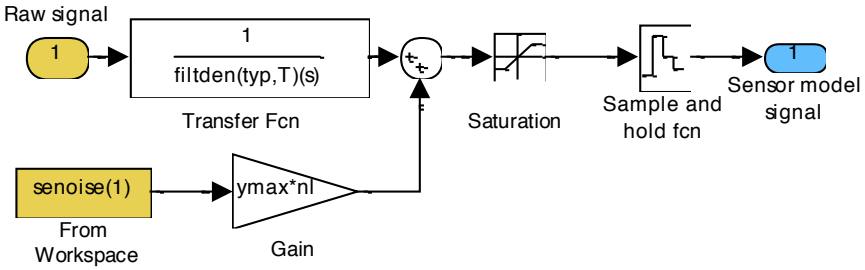


Figure 12: Simulink implementation class B_1, C_1.

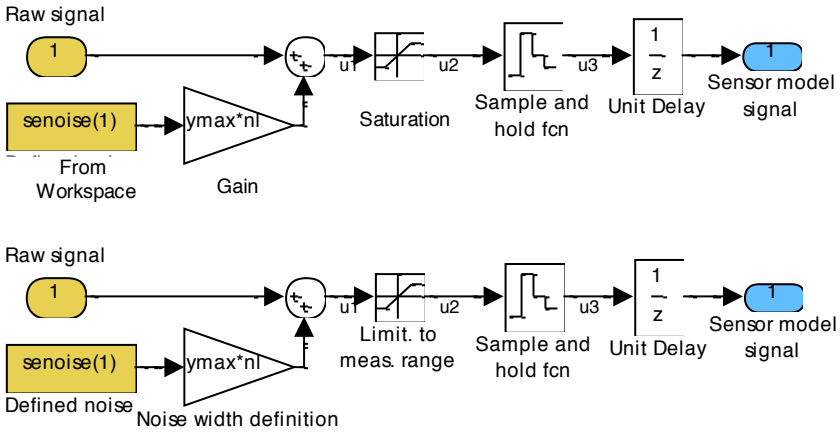


Figure 13: Simulink implementation class D.

Sensor class D represents batch-type reactors, for which any of the continuous delay times are negligible, compared to the batch operation of the measurement. An appropriate Simulink implementation is demonstrated in Figure 13. This model adds noise to the original signal, limits the sum to the measuring range (0, *ymax*) and uses a sample and hold function followed by a unit delay ($y(k) = u_3(k-1)$). Figure 14 shows examples of the output signal for all sensor classes.

13.3.3. Conclusions

Table 17 summarizes the recommended sensor parameter values for BSM2. Except for the plant influent flow rate, all the other flows are not explicitly measured but can be considered as known for simplicity.

13.4. Control handles

For reasons of simplicity, all available control handles are considered to be ideal with regard to their behaviour. In the closed-loop test case, only one control handle is used: the oxygen transfer rate in reactor number 4 (K_La_4). The following control handles are considered to exist for the implementation of new control strategies on the benchmark plant:

- internal flow recirculation rate (Q_{int});
- return sludge flow rate (Q_r);
- wastage flow rate (Q_w);
- anoxic/aerobic volume – all five biological reactors are equipped with both aerators and mechanical mixing devices, i.e. in a discrete fashion the volumes for anoxic and aerobic behaviour can be modified;
- aeration intensity individually for each reactor (K_La_1, K_La_2, K_La_3, K_La_4, K_La_5), taking into account the dynamics of the aeration system;
- external carbon source flow rate (Q_{EC1}, Q_{EC2}, Q_{eEC3}, Q_{EC4}, Q_{EC5}) where the carbon source is considered to consist of readily biodegradable substrate, i.e. COD_{EC};
- influent distribution by use of step feed (fractions of the influent flow to each of the five biological reactors: $f_{Qi1}, f_{Qi2}, f_{Qi3}, f_{Qi4}, f_{Qi5}$);
- distribution of internal flow recirculation (fractions of the internal recirculation flow to each of the five biological reactors: $f_{Qint1}, f_{Qint2}, f_{Qint3}, f_{Qint4}, f_{Qint5}$);
- distribution of return sludge flow (fractions of the return sludge flow to each of the five biological reactors: $f_{Qr1}, f_{Qr2}, f_{Qr3}, f_{Qr4}, f_{Qr5}$);
- reject water flow rate ($Q_{st,set}$)

The above selection gives about 30 individual control handles to manipulate the defined benchmark plant and dramatically increases its flexibility. Such a number of available control handles may not be realistic for a real plant but is defined for the benchmark plant in order to allow for basically any type of general control strategy. The defined limitations for the different control handles are given in Table 18.

The non-ideal aeration system (K_La_1- K_La_5) is defined with significant dynamics. A response time of t_r = 4 min is considered (see Rieger *et al.*, 2005). A second order time delay function gives a reasonable model of this process. The time constant of each of the two identical first order delays is $\tau = t_r/3.89 = 1.03$ min.

Table 17: Recommended BSM2 sensor parameters

Measured variable	Class	Measurement range	Measurement noise (δ)
Flow rate ($m^3.d^{-1}$) high range	A	0-100 000	2500
Water level (m)	A	0-5	0.125
Temperature (°C)	A	5-25	0.5
pH	A	5-9	0.1
S_O (g (-COD).m^{-3})	A	0-10	0.25
Sludge blanket level (m)	A	0-5	0.125
S_{NO} (g N.m^{-3})	B_0	0-20	0.5
S_{NH} (g N.m^{-3}) low range	B_0	0-20	0.5
S_{NH} (g N.m^{-3}) high range	B_0	0-50	1.25
S_{ALK} (mole HCO_3.m^{-3})	B_0	0-20	0.5
Mixed-liquor suspended solids (g.m^{-3})	A	0-10 000	250
Effluent total suspended solids (g.m^{-3})	A	0-200	5
COD_{tot} (g COD.m^{-3})	D	0-1 000	25
OUR (g (-COD).$m^{-3}.d^{-1}$)	D	0-2 000	50

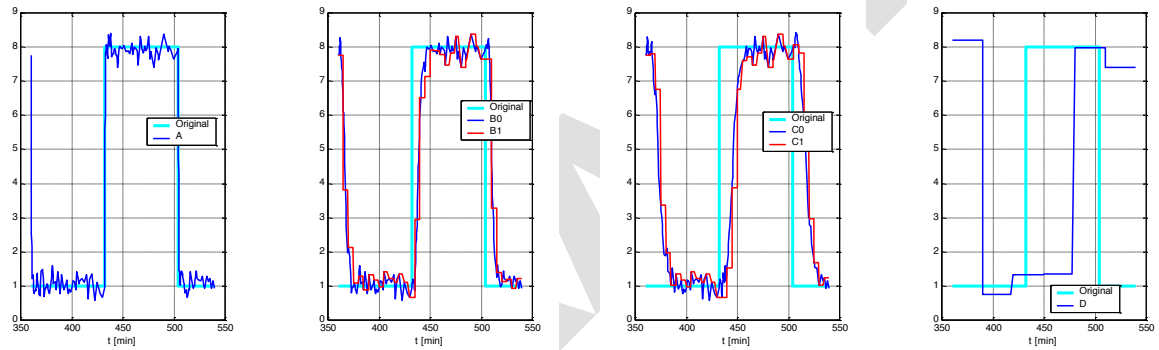


Figure 14: Pulse response of sensor classes.

13.5. Alternative description

To clarify the sensor and actuator models, a presentation in form of differential and difference equations is also presented in this section. The notations are summarized in Table 19.

713.5.1 Model for sensor class A and actuator model

(272)

(273)

(274)

(275)

13.5.2. Model for sensor class B_0 and C_0

(276)

(277)

(278)

(279)

(280)

Table 18: Available control handles and their limitations

Control handle	Minimum value	Maximum value	Comments
Q_{int} ($m^3.d^{-1}$)	0	309,720.	Max = 500% of $Q_{i,stab}$
Q_r ($m^3.d^{-1}$)	0	41,295.	Max = 200% of $Q_{i,stab}$
Q_w ($m^3.d^{-1}$)	0	1844.6	Max = 10% of $Q_{i,stab}$
K_La_4 (d^{-1})	0	240	Reactor 4
Q_{EC1} ($m^3.d^{-1}$)	0	5	Reactor 1 Carbon source conc. 400,000 g COD.m^{-3} available as COD_S (e.g. 25% ethanol solution)

Table 19: Variables used in the sensor models

Variable	Definition
$u(t)$	ideal measurement signal from process
$x_1(t)..x_7(t)$	internal states for dynamic part of sensor model
$u_2(t)$	delayed measurement signal (intermediate variable)
$y_1(t),y_2(t),y_3(k),y_4(k)$	intermediate signals
$y(t)$	real measurement signal from sensor (delayed, noisy, limited)
τ	time constant for one first order time delay
t_i	sampling interval for discontinuous sensor models

13.5.3. Model for sensor class B_1 and C_1

(281)

(282)

(283)

(284)

(285)

$$y_3(k)=y_2(t,\ t = k\cdot t_i) \quad (286)$$

$y(t)=y_3(k,\ k = \text{floor}(t/t_i))$ (287)

13.5.4. Model for sensor D

(288)

(289)

$y_3(k)=y_2(t,\ t = k \cdot t_i)$ (290)

$y_4(k)=y_3(k-1)$ (291)

$y(t)=y_4(k,\ k = \text{floor}(t/t_i))$ (292)

14. CONCLUDING REMARKS

The aim of this Technical Report is to give the details of the models used in BSM2. Further explanations concerning the reasoning which ended up with the choices made can be found in the Corresponding Technical Reports. Furthermore supplementary informations are given in the documents accompanying the software.

15. 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, *ECC'99 (European Control Conference)*, Karlsruhe, Germany.

Alex J., Benedetti L., Copp J., Gernaey K.V., Jeppsson U., Nopens I., Pons M.N., Steyer J.P. and Vanrolleghem P. (2009). Benchmark Simulation Model no. 1 (BSM1). In: Technical report no. ? of the Scientific and Technical Report, IWA Publishing, London.

ASCE (1993). A standard for the measurement of oxygen transfer in clean water. ASCE Transfer Standard Committee, New York, NY, USA.

Batstone D.J., Keller J., Angelidaki I., Kalyuzhnyi S.V., Pavlostathis S.G., Rozzi A., Sanders W.T.M., Siegrist H. and Vavilin V.A. (2002). Anaerobic Digestion Model No. 1. *IWA STR No. 13*, IWA Publishing, London, UK.

Copp J.B. (Ed.) (2002). The COST Simulation Benchmark: Description and Simulator Manual. Office for Official Publications of the European Community, Luxembourg, ISBN 92-894-1658-0, 154 pages.

Henze M., Grady Jr C.P.L., Gujer W., Marais G.v.R. and Matsuo T. (1987). Activated Sludge Model n° 1, *IAWQ Scientific and Technical Report n°1*, IAWQ, London, UK.

ISO 15839 (2003). Water quality – On-line sensors/ Analysing equipment for water – Specifications and performance tests, ISO 15839, ISO, Geneva, Switzerland.

Nopens I., Alex J., Batstone D., Copp J., Dudley J., Pons M.-N., Vanrolleghem P.A, Volcke E.I.P. and Jeppsson U. (2009). ASM/ADM/ASM interfaces for BSM2. In: Technical report no. ? of the Scientific and Technical Report, IWA Publishing, London.

Otterpohl R. and Freund M. (1992). Dynamic Models for clarifiers of activated sludge plants with dry and wet weather flows. *Wat. Sci. Tech*. **26**(5-6) 1391-1400.

Otterpohl R., Raak M. and Rolfs T. (1994). A mathematical model for the efficiency of the primary clarification. *Proceedings of IAWQ 17th Biennial Int. Conference*, Budapest, Hungary.

Pons M.N., Greffe J.L. and Bordet J. (1983). Fast pH calculations in aqueous solution chemistry, *Talanta*, **30**, 205-208.

Pons M.N., Spanjers H. and Jeppsson U. (1999). Towards a benchmark for evaluating control strategies in wastewater treatment plants by simulation, *Escape 9*, Budapest, Hungary.

Rieger L., Alex J., Winkler S., Boehler M., Thomann M. and Siegrist H. (2003). Progress in sensor technology – progress in process control? Part I: Sensor property investigation and classification. *Wat. Sci. Tech*., **47**(2), 103-112.

Rieger L., Alex J., Gujer W. and Siegrist H. (2005). Modelling of aeration systems at wastewater treatment plants. Proc. of the 2nd *IWA Conference on Instrumentation, Control and Automation,* Busan, Korea.

Rosen C. and Jeppsson U. (2009). Aspects on ADM1 implementation within the BSM2 framework. In: Technical report no. ? of the Scientific and Technical Report, IWA Publishing, London.

Rosen C., Vrecko D., Gernaey K.V., Pons M.-N. and Jeppsson U. (2006). Implementing ADM1 for plant-wide benchmark simulations in Matlab/Simulink. *Water Sci. Technol.,* **54**(4), 11-19.

Siegrist H., Vogt D., Garcia-Heras J.L. and Gujer, W. (2002). Mathematical model for meso- and thermophilic anaerobic digestion. *Environ. Sci. Technol*. **36**, 1113-1123.

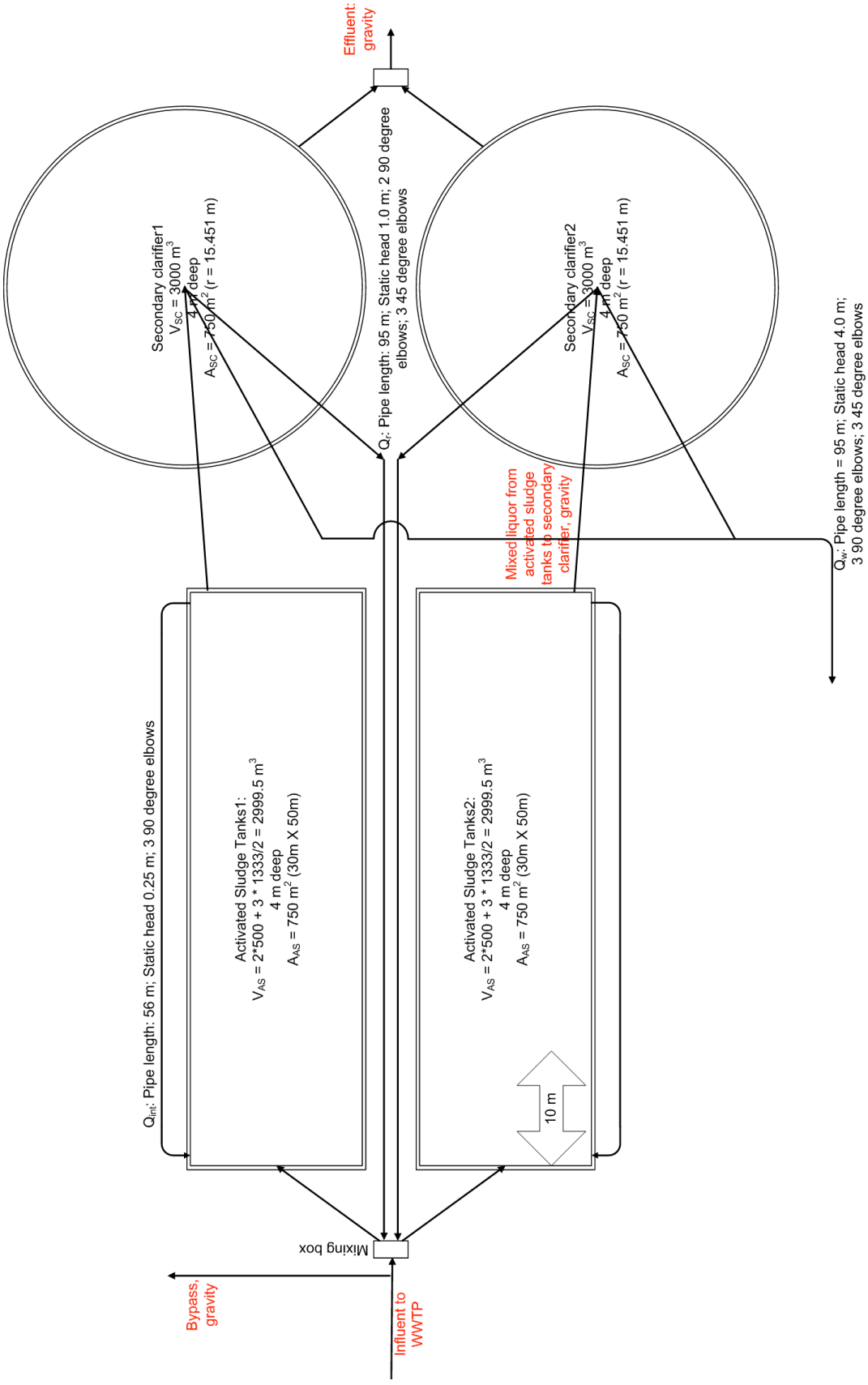
Takács I., Patry G.G. and Nolasco D. (1991). A dynamic model of the clarification thickening process, *Water Research*, **25**(10), 1263-1271.

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

Vanrolleghem P.A., Jeppsson U., Carstensen J., Carlsson B. and Olsson G. (1996). Integration of WWT plant design and operation - A systematic approach using cost functions, *Wat. Sci. Tech.*, **34**(3-4), 159-171.

Volcke E.I.P. (2006). *Modelling, analysis and control of partial nitritation in a SHARON reactor*. PhD thesis, Ghent University, Belgium, pp. 300 (Section 3.4. pH calculation available at http://biomath.ugent.be/publications/download/VolckeEveline_PhD.pdf).

Appendix 1: Practical BSM1 plant layout



Appendix 2: Source codes of ASM/ADM/ASM interfaces for BSM2

A2.1. MATLAB code

```
/*
 * New version (no 3) of the ASM1 to ADM1 interface based on discussions
 * within the IWA TG BSM community during 2002-2006. Now also including charge
 * balancing and temperature dependency for applicable parameters.
 * Model parameters are defined in adm1init_bsm2.m
 * u is the input in ASM1 terminology + extra dummy states, 21 variables
 * plus one extra input = dynamic pH from the ADM1 system (needed for
 * accurate charge balancing - also used the ADM1 to ASM1 interface).
 * If temperature control of AD is used then the operational temperature
 * of the ADM1 should also be an input rather than a defined parameter.
 * Temperature in the ADM1 and the ASM1 to ADM1 and the ADM1 to ASM1
 * interfaces should be identical at every time instant.
 * Input vector:
 * u[0] : Si = soluble inert organic material (g COD/m3)
 * u[1] : Ss = readily biodegradable substrate (g COD/m3)
 * u[2] : Xi = particulate inert organic material (g COD/m3)
 * u[3] : Xs = slowly biodegradable substrate (g COD/m3)
 * u[4] : Xbh = active heterotrophic biomass (g COD/m3)
 * u[5] : Xba = active autotrophic biomass (g COD/m3)
 * u[6] : Xp = particulate product arising from biomass decay (g COD/m3)
 * u[7] : So = oxygen (g -COD/m3)
 * u[8] : Sno = nitrate and nitrite nitrogen (g N/m3)
 * u[9] : Snh = ammonia and ammonium nitrogen (g N/m3)
 * u[10] : Snd = soluble biogradable organic nitrogen (g N/m3)
 * u[11] : Xnd = particulate biogradable organic nitrogen (g N/m3)
 * u[12] : Salk = alkalinity (mole HCO3-/m3)
 * u[13] : TSS = total suspended solids (internal use) (mg SS/l)
 * u[14] : flow rate (m3/d)
 * u[15] : temperature (deg C)
 * u[16:20] : dummy states for future use
 * u[21] : pH in the anaerobic digester
 *
 * y is the output in ADM1 terminology + extra dummy states, 33 variables
 * y[0] : Ssu = monosacharides (kg COD/m3)
 * y[1] : Saa = amino acids (kg COD/m3)
 * y[2] : Sfa = long chain fatty acids (LCFA) (kg COD/m3)
 * y[3] : Sva = total valerate (kg COD/m3)
 * y[4] : Sbu = total butyrate (kg COD/m3)
 * y[5] : Spro = total propionate (kg COD/m3)
 * y[6] : Sac = total acetate (kg COD/m3)
 * y[7] : Sh2 = hydrogen gas (kg COD/m3)
 * y[8] : Sch4 = methane gas (kg COD/m3)
 * y[9] : Sic = inorganic carbon (kmole C/m3)
 * y[10] : Sin = inorganic nitrogen (kmole N/m3)
 * y[11] : Si = soluble inerts (kg COD/m3)
 * y[12] : Xc = composites (kg COD/m3)
 * y[13] : Xch = carbohydrates (kg COD/m3)
 * y[14] : Xpr = proteins (kg COD/m3)
 * y[15] : Xli = lipids (kg COD/m3)
 * y[16] : Xsu = sugar degraders (kg COD/m3)
 * y[17] : Xaa = amino acid degraders (kg COD/m3)
 * y[18] : Xfa = LCFA degraders (kg COD/m3)
 * y[19] : Xc4 = valerate and butyrate degraders (kg COD/m3)
 * y[20] : Xpro = propionate degraders (kg COD/m3)
```

```
 * y[21] : Xac = acetate degraders (kg COD/m3)
 * y[22] : Xh2 = hydrogen degraders (kg COD/m3)
 * y[23] : Xi = particulate inerts (kg COD/m3)
 * y[24] : scat+ = cations (metallic ions, strong base) (kmole/m3)
 * y[25] : san- = anions (metallic ions, strong acid) (kmole/m3)
 * y[26] : flow rate (m3/d)
 * y[27] : temperature (deg C)
 * y[28:32] : dummy states for future use
 *
 * ASM1 --> ADM1 conversion, version 3 for BSM2
 * Copyright: John Copp, Primodal Inc., Canada; Ulf Jeppsson, Lund
 *            University, Sweden; Damien Batstone, Univ of Queensland,
 *            Australia, Ingmar Nopens, Univ of Ghent, Belgium,
 *            Marie-Noelle Pons, Nancy, France, Peter Vanrolleghem,
 *            Univ. Laval, Canada, Jens Alex, IFAK, Germany and
 *            Eveline Volcke, Univ of Ghent, Belgium.
 */

#define S_FUNCTION_NAME asm2adm_v3_bsm2

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

#define PAR     ssGetArg(S,0)

/*
 * mdlInitializeSizes - initialize the sizes array
 */
static void mdlInitializeSizes(SimStruct *S)
{
   ssSetNumContStates(    S, 0);   /* number of continuous states           */
   ssSetNumDiscStates(    S, 0);   /* number of discrete states             */
   ssSetNumInputs(        S, 22);  /* number of inputs                      */
   ssSetNumOutputs(       S, 33);  /* number of outputs                     */
   ssSetDirectFeedThrough(S, 1);   /* direct feedthrough flag               */
   ssSetNumSampleTimes(   S, 1);   /* number of sample times                */
   ssSetNumSFcnParams(    S, 1);   /* number of input arguments             */
   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)
{
}

/*
```

```
 * mdlOutputs - compute the outputs
 */

static void mdlOutputs(double *y, double *x, double *u, SimStruct *S, int tid)
{
        double CODequiv, fnaa, fnxc, fnbac, fxni, fsni, fsni_adm, frlixs, frlibac, frxs_adm, fdegrade_adm, frxs_as, fdegrade_as;
   double R, T_base, T_op, pK_w_base, pK_a_va_base, pK_a_bu_base, pK_a_pro_base, pK_a_ac_base, pK_a_co2_base, pK_a_IN_base;
   double pH_adm, pK_w, pK_a_co2, pK_a_IN, alfa_va, alfa_bu, alfa_pro, alfa_ac, alfa_co2, alfa_IN, alfa_NH, alfa_alk, alfa_NO, factor;
   double CODdemand, remaina, remainb, remainc, remaind, ScatminusSan;
        double sorgn, xorgn, xprtemp, xprtemp2, xlitemp, xlitemp2, xlitemp3, xchtemp, xchtemp2, xchtemp3;
   double biomass, biomass_nobio, biomass_bioN, remainCOD, inertX, xc, noninertX, inertS, utemp[22], utemp2[22];
        int i;

   /* parameters defined in adm1init_bsm2.m, INTERFACEPAR */
   CODequiv = mxGetPr(PAR)[0];
   fnaa = mxGetPr(PAR)[1];
   fnxc = mxGetPr(PAR)[2];
   fnbac = mxGetPr(PAR)[3];
   fxni = mxGetPr(PAR)[4];
   fsni = mxGetPr(PAR)[5];
   fsni_adm = mxGetPr(PAR)[6];
   frlixs = mxGetPr(PAR)[7];
   frlibac = mxGetPr(PAR)[8];
   frxs_adm = mxGetPr(PAR)[9];
   fdegrade_adm = mxGetPr(PAR)[10];
   frxs_as = mxGetPr(PAR)[11];      /* not used in ASM2ADM */
   fdegrade_as = mxGetPr(PAR)[12];   /* not used in ASM2ADM */
   R = mxGetPr(PAR)[13];
   T_base = mxGetPr(PAR)[14];
   T_op = mxGetPr(PAR)[15];          /* should be an input variable if dynamic temperature control is used */
   pK_w_base = mxGetPr(PAR)[16];
   pK_a_va_base = mxGetPr(PAR)[17];
   pK_a_bu_base = mxGetPr(PAR)[18];
   pK_a_pro_base = mxGetPr(PAR)[19];
   pK_a_ac_base = mxGetPr(PAR)[20];
   pK_a_co2_base = mxGetPr(PAR)[21];
   pK_a_IN_base = mxGetPr(PAR)[22];

   pH_adm = u[21];

   factor = (1.0/T_base - 1.0/T_op)/(100.0*R);
   pK_w = pK_w_base - log10(exp(55900.0*factor));
   pK_a_co2 = pK_a_co2_base - log10(exp(7646.0*factor));
   pK_a_IN = pK_a_IN_base - log10(exp(51965.0*factor));
   alfa_va = 1.0/208.0*(-1.0/(1.0 + pow(10, pK_a_va_base - pH_adm)));
   alfa_bu = 1.0/160.0*(-1.0/(1.0 + pow(10, pK_a_bu_base - pH_adm)));
   alfa_pro = 1.0/112.0*(-1.0/(1.0 + pow(10, pK_a_pro_base - pH_adm)));
   alfa_ac = 1.0/64.0*(-1.0/(1.0 + pow(10, pK_a_ac_base - pH_adm)));
   alfa_co2 = -1.0/(1.0 + pow(10, pK_a_co2 - pH_adm));
   alfa_IN = (pow(10, pK_a_IN - pH_adm))/(1.0 + pow(10, pK_a_IN - pH_adm));
   alfa_NH = 1.0/14000.0;  /* convert mgN/l into kmoleN/m3 */
   alfa_alk = -0.001;      /* convert moleHCO3/m3 into kmoleHCO3/m3 */
   alfa_NO = -1.0/14000.0; /* convert mgN/l into kmoleN/m3 */
```

```
        for (i = 0; i < 22; i++) {
        utemp[i] = u[i];
      utemp2[i] = u[i];
   }

        for (i = 0; i < 32; i++)
                y[i] = 0.0;


/*=============================================================================
====================*/
   /* Let CODdemand be the COD demand of available electron
   * acceptors prior to the anaerobic digester, i.e. oxygen and nitrate */
   CODdemand = u[7] + CODequiv*u[8];
   utemp[7] = 0;
   utemp[8] = 0;

   /* if extreme detail was used then some extra NH4 would be transformed
   * into N bound in biomass and some biomass would be formed when
   * removing the CODdemand (based on the yield). But on a total COD balance
        * approach the below is correct (neglecting the N need for biomass growth)
   * The COD is reduced in a hierarchical approach in the order:
   * 1) SS; 2) XS; 3) XBH; 4) XBA. It is no real improvement to remove SS and add
   * biomass. The net result is the same.*/

        if (CODdemand > u[1]) {         /* check if COD demand can be fulfilled by SS*/
                remaina = CODdemand - u[1];
                utemp[1] = 0.0;
                if (remaina > u[3]) {       /* check if COD demand can be fulfilled by XS*/
                remainb = remaina - u[3];
                utemp[3] = 0.0;
                if (remainb > u[4]) {       /* check if COD demand can be fulfilled by XBH */
                        remainc = remainb - u[4];
          utemp[9] = utemp[9] + u[4]*fnbac;
                        utemp[4] = 0.0;
                        if (remainc > u[5]) {       /* check if COD demand can be fulfilled by XBA */
                        remaind = remainc - u[5];
             utemp[9] = utemp[9] + u[5]*fnbac;
                        utemp[5] = 0.0;
             utemp[7] = remaind;
                                        /* if here we are in trouble, carbon shortage: an error printout
should be given */
                /* and execution stopped */
                                }
                        else {              /* reduced all COD demand by use of SS, XS, XBH and XBA */
                        utemp[5] = u[5] - remainc;
             utemp[9] = utemp[9] + remainc*fnbac;
           }
        }
                        else {              /* reduced all COD demand by use of SS, XS and XBH */
                        utemp[4] = u[4] - remainb;
          utemp[9] = utemp[9] + remainb*fnbac;
        }
                }
                else {              /* reduced all COD demand by use of SS and XS */
                utemp[3] = u[3] - remaina;
                }
         }
         else {             /* reduced all COD demand by use of SS */
                utemp[1] = u[1] - CODdemand;
```

```
        }


/*=============================================================================
====================*/
        /* SS becomes part of amino acids when transformed into ADM
   * and any remaining SS is mapped to monosacharides (no N contents)
   * Enough SND must be available for mapping to amino acids */

        sorgn = u[10]/fnaa;     /* Saa COD equivalent to SND */

        if (sorgn >= utemp[1]) {  /* not all SND-N in terms of COD fits into amino acids */
                y[1] = utemp[1];        /* map all SS COD into Saa */
                utemp[10] = utemp[10] - utemp[1]*fnaa; /* excess SND */
                utemp[1] = 0.0;                         /* all SS used */
        }
        else {                  /* all SND-N fits into amino acids */
                y[1] = sorgn;           /* map all SND related COD into Saa */
                utemp[1] = utemp[1] - sorgn;                /* excess SS, which will become sugar in
ADM1 i.e. no nitrogen association */
                utemp[10] = 0.0;                        /* all SND used */
        }


/*=============================================================================
====================*/
        /* XS becomes part of Xpr (proteins) when transformed into ADM
   * and any remaining XS is mapped to Xch and Xli (no N contents)
   * Enough XND must be available for mapping to Xpr */

        xorgn = u[11]/fnaa;     /* Xpr COD equivalent to XND */

        if (xorgn >= utemp[3]) {        /* not all XND-N in terms of COD fits into Xpr */
                xprtemp = utemp[3];         /* map all XS COD into Spr */
                utemp[11] = utemp[11] - utemp[3]*fnaa; /* excess XND */
                utemp[3] = 0.0;             /* all XS used */
      xlitemp = 0.0;
      xchtemp = 0.0;
        }
        else {                  /* all XND-N fits into Xpr */
                xprtemp = xorgn;        /* map all XND related COD into Xpr */
      xlitemp = frlixs*(utemp[3] - xorgn);    /* part of XS COD not associated with N */
      xchtemp = (1.0 - frlixs)*(utemp[3] - xorgn);    /* part of XS COD not associated with N */
      utemp[3] = 0.0;           /* all XS used */
                utemp[11] = 0.0;                        /* all XND used */
        }


/*=============================================================================
====================*/
   /* Biomass becomes part of Xpr and XI when transformed into ADM
        * and any remaining XBH and XBA is mapped to Xch and Xli (no N contents)
        * Remaining XND-N can be used as nitrogen source to form Xpr */

   biomass = utemp[4] + utemp[5];
   biomass_nobio = biomass*(1.0 - frxs_adm);   /* part which is mapped to XI */
   biomass_bioN = (biomass*fnbac - biomass_nobio*fxni);
   if (biomass_bioN < 0.0) {
      /* Problems: if here we should print 'ERROR: not enough biomass N to map the requested inert
part' */
```

```
      /* and execution stopped */
   }
   if ((biomass_bioN/fnaa) <= (biomass - biomass_nobio)) {
      xprtemp2 = biomass_bioN/fnaa;   /* all biomass N used */
      remainCOD = biomass - biomass_nobio - xprtemp2;
      if ((utemp[11]/fnaa) > remainCOD) {  /* use part of remaining XND-N to form proteins */
         xprtemp2 = xprtemp2 + remainCOD;
         utemp[11] = utemp[11] - remainCOD*fnaa;
         remainCOD = 0.0;
         utemp[4] = 0.0;
         utemp[5] = 0.0;
      }
      else {      /* use all remaining XND-N to form proteins */
         xprtemp2 = xprtemp2 + utemp[11]/fnaa;
         remainCOD = remainCOD - utemp[11]/fnaa;
         utemp[11] = 0.0;
      }
      xlitemp2 = frlibac*remainCOD;       /* part of the COD not associated with N */
      xchtemp2 = (1.0 - frlibac)*remainCOD;  /* part of the COD not associated with N */
   }
   else {
      xprtemp2 = biomass - biomass_nobio; /* all biomass COD used */
      utemp[11] = utemp[11] + biomass*fnbac - biomass_nobio*fxni - xprtemp2*fnaa; /* any remaining
N in XND */
   }
   utemp[4] = 0.0;
   utemp[5] = 0.0;


/*=============================================================================
====================*/
   /* direct mapping of XI and XP to ADM1 XI (if fdegrade_ad = 0)
         * assumption: same N content in both ASM1 and ADM1 particulate inerts */

   inertX = (1-fdegrade_adm)*(utemp[2] + utemp[6]);

   /* special case: IF part of XI and XP in the ASM can be degraded in the AD
   * we have no knowledge about the contents so we put it in as composits (Xc)
         * we need to keep track of the associated nitrogen
         * N content which may be different, take first from XI&XP-N, then XND-N, then SND-N,
         * then SNH. A similar principle could be used for other states. */

   xc = 0.0;
   xlitemp3 = 0.0;
   xchtemp3 = 0.0;
   if (fdegrade_adm > 0)  {
      noninertX = fdegrade_adm*(utemp[2] + utemp[6]);
      if (fxni < fnxc) {   /* N in XI&XP(ASM) not enough */
         xc = noninertX*fxni/fnxc;
         noninertX = noninertX - noninertX*fxni/fnxc;
         if (utemp[11] < (noninertX*fnxc)) {   /* N in XND not enough */
            xc = xc + utemp[11]/fnxc;
            noninertX = noninertX - utemp[11]/fnxc;
            utemp[11] = 0.0;
            if (utemp[10] < (noninertX*fnxc)) {   /* N in SND not enough */
               xc = xc + utemp[10]/fnxc;
               noninertX = noninertX - utemp[10]/fnxc;
               utemp[10] = 0.0;
               if (utemp[9] < (noninertX*fnxc)) {   /* N in SNH not enough */
                  xc = xc + utemp[9]/fnxc;
```

```
                noninertX = noninertX - utemp[9]/fnxc;
                utemp[9] = 0.0;
                /* Should be a WARNING printout: Nitrogen shortage when converting biodegradable
XI&XP
                * Putting remaining XI&XP as lipids (50%) and carbohydrates (50%) */
                xlitemp3 = 0.5*noninertX;
                xchtemp3 = 0.5*noninertX;
                noninertX = 0.0;
                }
             else {   /* N in SNH enough for mapping */
                xc = xc + noninertX;
                utemp[9] = utemp[9] - noninertX*fnxc;
                noninertX = 0.0;
                }
             }
          else  {   /* N in SND enough for mapping */
             xc = xc + noninertX;
             utemp[10] = utemp[10] - noninertX*fnxc;
             noninertX = 0.0;
             }
          }
       else  {   /* N in XND enough for mapping */
          xc = xc + noninertX;
          utemp[11] = utemp[11] - noninertX*fnxc;
          noninertX = 0.0;
          }
       }
    else  {   /* N in XI&XP(ASM) enough for mapping */
       xc = xc + noninertX;
       utemp[11] = utemp[11] + noninertX*(fxni-fnxc);   /* put remaining N as XND */
       noninertX = 0;
       }
  }


/*=============================================================================
====================*/
   /* Mapping of ASM SI to ADM1 SI
         * N content may be different, take first from SI-N, then SND-N, then XND-N,
         * then SNH. Similar principle could be used for other states. */

   inertS = 0.0;
   if (fsni < fsni_adm) {   /* N in SI(ASM) not enough */
      inertS = utemp[0]*fsni/fsni_adm;
      utemp[0] = utemp[0] - utemp[0]*fsni/fsni_adm;
      if (utemp[10] < (utemp[0]*fsni_adm)) {    /* N in SND not enough */
         inertS = inertS + utemp[10]/fsni_adm;
         utemp[0] = utemp[0] - utemp[10]/fsni_adm;
         utemp[10] = 0.0;
         if (utemp[11] < (utemp[0]*fsni_adm)) {   /* N in XND not enough */
            inertS = inertS + utemp[11]/fsni_adm;
            utemp[0] = utemp[0] - utemp[11]/fsni_adm;
            utemp[11] = 0.0;
            if (utemp[9] < (utemp[0]*fsni_adm)) {   /* N in SNH not enough */
               inertS = inertS + utemp[9]/fsni_adm;
               utemp[0] = utemp[0] - utemp[9]/fsni_adm;
               utemp[9] = 0.0;
               /* Here there shpuld be a warning printout: Nitrogen shortage when converting SI
               * Putting remaining SI as monosacharides */
               utemp[1] = utemp[1] + utemp[0];
```

```
            utemp[0] = 0.0;
            }
         else  {   /* N in SNH enough for mapping */
            inertS = inertS + utemp[0];
            utemp[9] = utemp[9] - utemp[0]*fsni_adm;
            utemp[0] = 0.0;
            }
         }
      else  {   /* N in XND enough for mapping */
         inertS = inertS + utemp[0];
         utemp[11] = utemp[11] - utemp[0]*fsni_adm;
         utemp[0] = 0.0;
         }
      }
   else  {  /* N in SND enough for mapping */
      inertS = inertS + utemp[0];
      utemp[10] = utemp[10] - utemp[0]*fsni_adm;
      utemp[0] = 0.0;
      }
   }
else  {   /* N in SI(ASM) enough for mapping */
   inertS = inertS + utemp[0];
   utemp[10] = utemp[10] + utemp[0]*(fsni-fsni_adm);   /* put remaining N as SND */
   utemp[0] = 0.0;
   }


/*================================================================================
====================*/
   /* Define the outputs including charge balance */

   y[0] = utemp[1]/1000.0;
   y[1] = y[1]/1000.0;
   y[10] = (utemp[9] + utemp[10] + utemp[11])/14000.0;
   y[11] = inertS/1000.0;
   y[12] = xc/1000.0;
   y[13] = (xchtemp + xchtemp2 + xchtemp3)/1000.0;
   y[14] = (xprtemp + xprtemp2)/1000.0;
   y[15] = (xlitemp + xlitemp2 + xlitemp3)/1000.0;
   y[23] = (biomass_nobio + inertX)/1000.0;
   y[26] = u[14];  /* flow rate */
   y[27] = T_op - 273.15;   /* temperature, degC */
   y[28] = u[16];  /* dummy state */
   y[29] = u[17];  /* dummy state */
   y[30] = u[18];  /* dummy state */
   y[31] = u[19];  /* dummy state */
   y[32] = u[20];  /* dummy state */

   /* charge balance, output S_IC */
   y[9] = ((utemp2[8]*alfa_NO + utemp2[9]*alfa_NH + utemp2[12]*alfa_alk) - (y[3]*alfa_va +
y[4]*alfa_bu + y[5]*alfa_pro + y[6]*alfa_ac + y[10]*alfa_IN))/alfa_co2;

   /* calculate anions and cations based on full charge balance including H+ and OH- */
   ScatminusSan = y[3]*alfa_va + y[4]*alfa_bu + y[5]*alfa_pro + y[6]*alfa_ac + y[10]*alfa_IN +
y[9]*alfa_co2 + pow(10, (-pK_w + pH_adm)) - pow(10, -pH_adm);

   if (ScatminusSan > 0)  {
      y[24] = ScatminusSan;
      y[25] = 0.0;
      }
```

```
    else  {
        y[24] = 0.0;
        y[25] = -1.0*ScatminusSan;
        }

    /* Finally there should be a input-output mass balance check here of COD and N */

}

/*
 * 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)
{
}


/*
 * 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


/*
 * New version (no 3) of the ADM1 to ASM1 interface based on discussions
 * within the IWA TG BSM community during 2002-2006. Now also including charge
 * balancing and temperature dependency for applicable parameters.
 * Model parameters are defined in adm1init_bsm2.m
 * u is the input in ADM1 terminology + extra dummy states, 33 variables
 * plus two extra inputs: 1) dynamic pH from the ADM1 system (needed for
 * accurate charge balancing - also used the ASM1 to ADM1 interface) and
 * 2) wastewater temperature into the ASM2ADM interface, which is used as
 * the output temperature from the ADM2ASM interface (assume heat exchangers etc).
 * If temperature control of AD is used then the operational temperature
 * of the ADM1 should also be an input rather than a defined parameter.
 * Temperature in the ADM1 and the ASM1 to ADM1 and the ADM1 to ASM1
 * interfaces should be identical at every time instant.
 * The interface assumes identical N-content of particulate inerts in both
 * AD and AS. The same holds for biomass. The N-content of soluble inerts may vary.
 *
 * u is the input in ADM1 terminology + extra dummy states, 33 variables
 * u[0] : Ssu = monosacharides (kg COD/m3)
 * u[1] : Saa = amino acids (kg COD/m3)
 * u[2] : Sfa = long chain fatty acids (LCFA) (kg COD/m3)
```

```
* u[3] : Sva = total valerate (kg COD/m3)
* u[4] : Sbu = total butyrate (kg COD/m3)
* u[5] : Spro = total propionate (kg COD/m3)
* u[6] : Sac = total acetate (kg COD/m3)
* u[7] : Sh2 = hydrogen gas (kg COD/m3)
* u[8] : Sch4 = methane gas (kg COD/m3)
* u[9] : Sic = inorganic carbon (kmole C/m3)
* u[10] : Sin = inorganic nitrogen (kmole N/m3)
* u[11] : Si = soluble inerts (kg COD/m3)
* u[12] : Xc = composites (kg COD/m3)
* u[13] : Xch = carbohydrates (kg COD/m3)
* u[14] : Xpr = proteins (kg COD/m3)
* u[15] : Xli = lipids (kg COD/m3)
* u[16] : Xsu = sugar degraders (kg COD/m3)
* u[17] : Xaa = amino acid degraders (kg COD/m3)
* u[18] : Xfa = LCFA degraders (kg COD/m3)
* u[19] : Xc4 = valerate and butyrate degraders (kg COD/m3)
* u[20] : Xpro = propionate degraders (kg COD/m3)
* u[21] : Xac = acetate degraders (kg COD/m3)
* u[22] : Xh2 = hydrogen degraders (kg COD/m3)
* u[23] : Xi = particulate inerts (kg COD/m3)
* u[24] : scat+ = cations (metallic ions, strong base) (kmole/m3)
* u[25] : san- = anions (metallic ions, strong acid) (kmole/m3)
* u[26] : flow rate (m3/d)
* u[27] : temperature (deg C)
* u[28:32] : dummy states for future use
* u[33] : dynamic pH from the ADM1
* u[34] : wastewater temperature into the ASM2ADM interface, deg C
*
* Output vector:
* y[0] : Si = soluble inert organic material (g COD/m3)
* y[1] : Ss = readily biodegradable substrate (g COD/m3)
* y[2] : Xi = particulate inert organic material (g COD/m3)
* y[3] : Xs = slowly biodegradable substrate (g COD/m3)
* y[4] : Xbh = active heterotrophic biomass (g COD/m3)
* y[5] : Xba = active autotrophic biomass (g COD/m3)
* y[6] : Xp = particulate product arising from biomass decay (g COD/m3)
* y[7] : So = oxygen (g -COD/m3)
* y[8] : Sno = nitrate and nitrite nitrogen (g N/m3)
* y[9] : Snh = ammonia and ammonium nitrogen (g N/m3)
* y[10] : Snd = soluble biogradable organic nitrogen (g N/m3)
* y[11] : Xnd = particulate biogradable organic nitrogen (g N/m3)
* y[12] : Salk = alkalinity (mole HCO3-/m3)
* y[13] : TSS = total suspended solids (internal use) (mg SS/l)
* y[14] : flow rate (m3/d)
* y[15] : temperature (deg C)
* y[16:20] : dummy states for future use
*
* ADM1 --> ASM1 conversion, version 3 for BSM2
* Copyright: John Copp, Primodal Inc., Canada; Ulf Jeppsson, Lund
*           University, Sweden; Damien Batstone, Univ of Queensland,
*           Australia, Ingmar Nopens, Univ of Ghent, Belgium,
*           Marie-Noelle Pons, Nancy, France, Peter Vanrolleghem,
*           Univ. Laval, Canada, Jens Alex, IFAK, Germany and
*           Eveline Volcke, Univ of Ghent, Belgium.
*/

#define S_FUNCTION_NAME adm2asm_v3_bsm2

#include "simstruc.h"
```

```
#include <math.h>

#define PAR     ssGetArg(S,0)

/*
 * mdlInitializeSizes - initialize the sizes array
 */
static void mdlInitializeSizes(SimStruct *S)
{
   ssSetNumContStates(    S, 0);   /* number of continuous states           */
   ssSetNumDiscStates(    S, 0);   /* number of discrete states             */
   ssSetNumInputs(        S, 35);  /* number of inputs                      */
   ssSetNumOutputs(       S, 21);  /* number of outputs                     */
   ssSetDirectFeedThrough(S, 1);   /* direct feedthrough flag               */
   ssSetNumSampleTimes(   S, 1);   /* number of sample times                */
   ssSetNumSFcnParams(    S, 1);   /* number of input arguments             */
   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)
{
}

/*
 * mdlOutputs - compute the outputs
 */

static void mdlOutputs(double *y, double *x, double *u, SimStruct *S, int tid)
{
        double CODequiv, fnaa, fnxc, fnbac, fxni, fsni, fsni_adm, frlixs, frlibac, frxs_adm,
fdegrade_adm, frxs_as, fdegrade_as;
   double     R, T_base, T_op, pK_w_base, pK_a_va_base, pK_a_bu_base, pK_a_pro_base,
pK_a_ac_base, pK_a_co2_base, pK_a_IN_base;
   double  pH_adm, pK_w, pK_a_co2, pK_a_IN, alfa_va, alfa_bu, alfa_pro, alfa_ac, alfa_co2, alfa_IN,
alfa_NH, alfa_alk, alfa_NO, factor;
        double  XPtemp, XStemp, XStemp2;
   double  biomass, biomass_nobio, biomass_bioN, remainCOD, inertX, noninertX, inertS, utemp[35];
        int     i;

   /* parameters defined in adm1init_bsm2.m, INTERFACEPAR */
   CODequiv = mxGetPr(PAR)[0];        /* not used in ADM2ASM */
   fnaa = mxGetPr(PAR)[1];
   fnxc = mxGetPr(PAR)[2];
   fnbac = mxGetPr(PAR)[3];
   fxni = mxGetPr(PAR)[4];
```

```
   fsni = mxGetPr(PAR)[5];
   fsni_adm = mxGetPr(PAR)[6];
   frlixs = mxGetPr(PAR)[7];          /* not used in ADM2ASM */
   frlibac = mxGetPr(PAR)[8];         /* not used in ADM2ASM */
   frxs_adm = mxGetPr(PAR)[9];        /* not used in ADM2ASM */
   fdegrade_adm = mxGetPr(PAR)[10];   /* not used in ADM2ASM */
   frxs_as = mxGetPr(PAR)[11];
   fdegrade_as = mxGetPr(PAR)[12];
   R = mxGetPr(PAR)[13];
   T_base = mxGetPr(PAR)[14];
   T_op = mxGetPr(PAR)[15];           /* should be an input variable if dynamic temperature control is
used */
   pK_w_base = mxGetPr(PAR)[16];
   pK_a_va_base = mxGetPr(PAR)[17];
   pK_a_bu_base = mxGetPr(PAR)[18];
   pK_a_pro_base = mxGetPr(PAR)[19];
   pK_a_ac_base = mxGetPr(PAR)[20];
   pK_a_co2_base = mxGetPr(PAR)[21];
   pK_a_IN_base = mxGetPr(PAR)[22];

   pH_adm = u[33];

   factor = (1.0/T_base - 1.0/T_op)/(100.0*R);
   pK_w = pK_w_base - log10(exp(55900.0*factor));
   pK_a_co2 = pK_a_co2_base - log10(exp(7646.0*factor));
   pK_a_IN = pK_a_IN_base - log10(exp(51965.0*factor));
   alfa_va = 1.0/208.0*(-1.0/(1.0 + pow(10, pK_a_va_base - pH_adm)));
   alfa_bu = 1.0/160.0*(-1.0/(1.0 + pow(10, pK_a_bu_base - pH_adm)));
   alfa_pro = 1.0/112.0*(-1.0/(1.0 + pow(10, pK_a_pro_base - pH_adm)));
   alfa_ac = 1.0/64.0*(-1.0/(1.0 + pow(10, pK_a_ac_base - pH_adm)));
   alfa_co2 = -1.0/(1.0 + pow(10, pK_a_co2 - pH_adm));
   alfa_IN = (pow(10, pK_a_IN - pH_adm))/(1.0 + pow(10, pK_a_IN - pH_adm));
   alfa_NH = 1.0/14000.0;  /* convert mgN/l into kmoleN/m3 */
   alfa_alk = -0.001;      /* convert moleHCO3/m3 into kmoleHCO3/m3 */
   alfa_NO = -1.0/14000.0; /* convert mgN/l into kmoleN/m3 */

         for (i = 0; i < 35; i++)
         utemp[i] = u[i];

         for (i = 0; i < 21; i++)
                  y[i] = 0.0;


/*=============================================================================
====================*/
   /* Biomass becomes part of XS and XP when transformed into ASM
         * Assume Npart of formed XS to be fnxc and Npart of XP to be fxni
         * Remaining N goes into the ammonia pool (also used as source if necessary) */

   biomass = 1000.0*(utemp[16] + utemp[17] + utemp[18] + utemp[19] + utemp[20] + utemp[21] +
utemp[22]);
   biomass_nobio = biomass*(1.0 - frxs_as);   /* part which is mapped to XP */
   biomass_bioN = (biomass*fnbac - biomass_nobio*fxni);
   remainCOD = 0.0;
   if (biomass_bioN < 0.0) {
      /* Problems: if here we should print 'WARNING: not enough biomass N to map the requested
inert part of biomass' */
      /* We map as much as we can, and the remains go to XS! */
      XPtemp = biomass*fnbac/fxni;
      biomass_nobio = XPtemp;
```

```
      biomass_bioN = 0.0;
   }
   else  {
      XPtemp = biomass_nobio;
      }
   if ((biomass_bioN/fnxc) <= (biomass - biomass_nobio)) {
      XStemp = biomass_bioN/fnxc;        /* all biomass N used */
      remainCOD = biomass - biomass_nobio - XStemp;
      if ((utemp[10]*14000.0/fnaa) >= remainCOD) {  /* use part of remaining S_IN to form XS */
         XStemp = XStemp + remainCOD;
      }
      else {
         /* Problems: if here we should print 'ERROR: not enough nitrogen to map the requested XS
part of biomass' */
         /* System failure! */
      }
   }
   else {
      XStemp = biomass - biomass_nobio; /* all biomass COD used */
   }

   utemp[10] = utemp[10] + biomass*fnbac/14000.0 - XPtemp*fxni/14000.0 - XStemp*fnxc/14000.0;  /*
any remaining N in S_IN */
   y[3] = (utemp[12] + utemp[13] + utemp[14] + utemp[15])*1000.0 + XStemp;       /* Xs = sum all X
except Xi, + biomass as handled above */
   y[6] = XPtemp;      /* inert part of biomass */



/*=============================================================================
====================*/
   /*  mapping of inert XI in AD into XI and possibly XS in AS
         * assumption: same N content in both ASM1 and ADM1 particulate inerts
         * special case: if part of XI in AD can be degraded in AS
         * we have no knowledge about the contents so we put it in as part substrate (XS)
         * we need to keep track of the associated nitrogen
         * N content may be different, take first from XI-N then S_IN,
         * Similar principle could be used for other states. */
   inertX = (1.0-fdegrade_as)*utemp[23]*1000.0;
   XStemp2 = 0.0;
   noninertX = 0.0;
   if (fdegrade_as > 0.0) {
      noninertX = fdegrade_as*utemp[23]*1000.0;
      if (fxni < fnxc)  {     /* N in XI(AD) not enough */
         XStemp2 = noninertX*fxni/fnxc;
         noninertX = noninertX - noninertX*fxni/fnxc;
         if ((utemp[10]*14000.0) < (noninertX*fnxc))  {  /* N in SNH not enough */
            XStemp2 = XStemp2 + (utemp[10]*14000.0)/fnxc;
            noninertX = noninertX - (utemp[10]*14000.0)/fnxc;
            utemp[10] = 0.0;
            /* Problems: if here we should print 'WARNING: Nitrogen shortage when converting
biodegradable XI' */
            /* Mapping what we can to XS and putting remaining XI back into XI of ASM */
            inertX = inertX + noninertX;
            }
         else  {   /* N in S_IN enough for mapping */
            XStemp2 = XStemp2 + noninertX;
            utemp[10] = utemp[10] - noninertX*fnxc/14000.0;
            noninertX = 0.0;
            }
```

```
        }
      else  {   /* N in XI(AD) enough for mapping */
        XStemp2 = XStemp2 + noninertX;
        utemp[10] = utemp[10] + noninertX*(fxni - fnxc)/14000.0;    /* put remaining N as S_IN */
        noninertX = 0;
        }
      }

   y[2] = inertX;          /* Xi = Xi*fdegrade_AS + possibly nonmappable XS */
   y[3] = y[3] + XStemp2;  /* extra added XS (biodegradable XI) */


/*=========================================================================
====================*/
   /* Mapping of ADM SI to ASM1 SI
   * It is assumed that this mapping will be 100% on COD basis
         * N content may be different, take first from SI-N then from S_IN.
         * Similar principle could be used for other states. */

   inertS = 0.0;
   if (fsni_adm < fsni) {   /* N in SI(AD) not enough */
      inertS = utemp[11]*fsni_adm/fsni;
      utemp[11] = utemp[11] - utemp[11]*fsni_adm/fsni;
      if ((utemp[10]*14.0) < (utemp[11]*fsni))  {  /* N in S_IN not enough */
        inertS = inertS + utemp[10]*14.0/fsni;
        utemp[11] = utemp[11] - utemp[10]*14.0/fsni;
        utemp[10] = 0.0;
        /* Problems: if here we should print 'ERROR: Nitrogen shortage when converting SI' */
        /* System failure: nowhere to put SI */
        }
      else  {  /* N in S_IN enough for mapping */
        inertS = inertS + utemp[11];
        utemp[10] = utemp[10] - utemp[11]*fsni/14.0;
        utemp[11] = 0.0;
        }
      }
   else  {    /* N in SI(AD) enough for mapping */
      inertS = inertS + utemp[11];
      utemp[10] = utemp[10] + utemp[11]*(fsni_adm - fsni)/14.0;  /* put remaining N as S_IN */
      utemp[11] = 0.0;
      }

   y[0] = inertS*1000.0;            /* Si = Si */


/*=========================================================================
====================*/
   /* Define the outputs including charge balance */

   /* nitrogen in biomass, composites, proteins
         * Xnd is the nitrogen part of Xs in ASM1. Therefore Xnd should be based on the
         * same variables as constitutes Xs, ie AD biomass (part not mapped to XP), xc and xpr if we
assume
         * there is no nitrogen in carbohydrates and lipids. The N content of Xi is
   * not included in Xnd in ASM1 and should in my view not be included. */

   y[11] = fnxc*(XStemp + XStemp2) + fnxc*1000.0*utemp[12] + fnaa*1000.0*utemp[14];

   /* Snd is the nitrogen part of Ss in ASM1. Therefore Snd should be based on the
         * same variables as constitutes Ss, and we assume
```

```
            * there is only nitrogen in the amino acids. The N content of Si is
       * not included in Snd in ASM1 and should in my view not be included. */

       y[10] = fnaa*1000.0*utemp[1];

       /* sh2 and sch4 assumed to be stripped upon reentry to ASM side */

       y[1] = (utemp[0] + utemp[1] + utemp[2] + utemp[3] + utemp[4] + utemp[5] + utemp[6])*1000.0; /*    Ss
= sum all S except Sh2, Sch4, Si, Sic, Sin */

       y[9] = utemp[10]*14000.0;                  /* Snh = S_IN including adjustments above */

       y[13] = 0.75*(y[2] + y[3] + y[4] + y[5] + y[6]);
       y[14] = utemp[26];  /* flow rate */
       y[15] = u[34];   /* temperature, degC, should be equal to AS temperature into the AD/AS interface */
       y[16] = utemp[28];  /* dummy state */
       y[17] = utemp[29];  /* dummy state */
       y[18] = utemp[30];  /* dummy state */
       y[19] = utemp[31];  /* dummy state */
       y[20] = utemp[32];  /* dummy state */

       /* charge balance, output S_alk (molHCO3/m3) */
       y[12] = (u[3]*alfa_va + u[4]*alfa_bu + u[5]*alfa_pro + u[6]*alfa_ac + u[9]*alfa_co2 + u[10]*alfa_IN -
y[8]*alfa_NO - y[9]*alfa_NH)/alfa_alk;

       /* Finally there should be a input-output mass balance check here of COD and N */

}

/*
 * 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)
{
}


/*
 * 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
```

A2.1. FORTRAN code

A2.2.1. ASM/ADM

```
      SUBROUTINE asm2admv2(asmm,adm,totCODin,totCODout,totNin,totNout,
     &ph,ancat,warn)

      IMPLICIT NONE
      INTEGER i,warn
      DOUBLE PRECISION fnaa,fnxc,fnxb,fni,fxi,fchxc,flixc
      DOUBLE PRECISION fnbac,fxni,fsni,fsni_adm
      DOUBLE PRECISION nxb,nxc,naa,ni,nbac,xni,sni,sni_adm
      DOUBLE PRECISION frlixs,frxs,frlixb,fdegrade
      DOUBLE PRECISION CODequiv,CODdemand
      DOUBLE PRECISION remaina,remainb,remainc,remaind,remainCOD
      DOUBLE PRECISION demand
      DOUBLE PRECISION sorgn,remorgn,sinertn,remCOD,xinertn,xorgn
      DOUBLE PRECISION xprtemp,xlitemp,xchtemp,biomass_nobio,biomass
      DOUBLE PRECISION biomass_bioN,xprtemp2,xlitemp2,xchtemp2
      DOUBLE PRECISION inertX,xc,xlitemp3,xchtemp3,noninertX,inertS
      DOUBLE PRECISION totTKNin,totCODin,totCODout,totTKNout
      DOUBLE PRECISION totNin,totNout,ph,ancat,pkk,tfac,bigr

c New version of ASM1 ADM1 interface based on discussions
c within the BSM community during 2002-2006.
c
c asmm is the input in ASM1 terminology, 13 state variables
c asmm[1] : Si = soluble inert organic material (g COD/m3)
c asmm[2] : Ss = readily biodegradable substrate (g COD/m3)
c asmm[3] : Xi = particulate inert organic material (g COD/m3)
c asmm[4] : Xs = slowly biodegradable substrate (g COD/m3)
c asmm[5] : Xbh = active heterotrophic biomass (g COD/m3)
c asmm6] : Xba = active autotrophic biomass (g COD/m3)
c asmm[7] : Xp = particulate product arising from biomass decay (g COD/m3)
c asmm[8] : So = oxygen (g -COD/m3)
c asmm[9] : Sno = nitrate and nitrite nitrogen (g N/m3)
c asmm[10] : Snh = ammonia and ammonium nitrogen (g N/m3)
c asmm[11] : Snd = soluble biogradable organic nitrogen (g N/m3)
c asmm[12] : Xnd = particulate biogradable organic nitrogen (g N/m3)
c asmm[13] : Salk = alkalinity (mole HCO3-/m3)
      DOUBLE PRECISION asmm(13),xtemp(13)
c adm is the output in ADM1 terminology, 24 state variables
c adm[1] : Ssu = monosacharides (kg COD/m3)
c adm[2] : Saa = amino acids (kg COD/m3)
c adm[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3)
c adm[4] : Sva = total valerate (kg COD/m3)
c adm[5] : Sbu = total butyrate (kg COD/m3)
c adm[6] : Spro = total propionate (kg COD/m3)
c adm[7] : Sac = total acetate (kg COD/m3)
c adm[8] : Sh2 = hydrogen gas (kg COD/m3)
c adm[9] : Sch4 = methane gas (kg COD/m3)
c adm[10] : Sic = inorganic carbon (kmole C/m3)
c adm[11] : Sin = inorganic nitrogen (kmole N/m3)
c adm[12] : Si = soluble inerts (kg COD/m3)
c adm[13] : Xc = composites (kg COD/m3)
c adm[14] : Xch = carbohydrates (kg COD/m3)
c adm[15] : Xpr = proteins (kg COD/m3)
c adm[16] : Xli = lipids (kg COD/m3)
c adm[17] : Xsu = sugar degraders (kg COD/m3)
```

```
c adm[18] : Xaa = amino acid degraders (kg COD/m3)
c adm[19] : Xfa = LCFA degraders (kg COD/m3)
c adm[20] : Xc4 = valerate and butyrate degraders (kg COD/m3)
c adm[21] : Xpro = propionate degraders (kg COD/m3)
c adm[22] : Xac = acetate degraders (kg COD/m3)
c adm[23] : Xh2 = hydrogen degraders (kg COD/m3)
c adm[24] : Xi = particulate inerts (kg COD/m3)
      DOUBLE PRECISION adm(24)
      DOUBLE PRECISION alfachac, alfachpro,alfachbu,alfachva
      DOUBLE PRECISION alfachin,alfachic,alfachnh
      DOUBLE PRECISION alfachalk,alfachno
      DOUBLE PRECISION chargeasm1,chargeadm1

      tfac=1.d00/298.15d00-1.d00/(273.15d00+35.d00)
      bigr=0.08314d-00
c
c ASM1 --> ADM1 conversion
c Copyright: John Copp, Primodal Inc., Canada Ulf Jeppsson, Lund
c            University, Sweden Damien Batstone, Univ of Queensland,
c            Australia and Ingmar Nopens, Univ of Ghent, Belgium

      do i=1,13
            xtemp(i)=asmm(i)
      enddo
      do i=1,24
            adm(i) = 0.d0
      enddo

      warn=0

c Parameters

c CODequiv is the conversion factor for COD demand of nitrate
c exact value of ASM1 2.86
      CODequiv = 40.d0/14.d0
c fraction of N in amino acids and Xpr as in ADM1 report
      fnaa = 0.098d0
c N content of biomass based on BSM1, same in AS and AD
      fnbac = 0.08d0
c N content of composite material based on BSM2
      fnxc = 0.0376d0
c N content of inerts XI and XP, same in AS and AD
      fxni = 0.06d0
c N content of SI, zero in ASM1 and BSM1
      fsni = 0.d0
c N content of SI in the AD system
      fsni_adm = 0.06d0
c fnbac, fxni and fsni are adjusted to fit the benchmark values of iXB=0.08
and
c iXP=0.06 and iSI=0.
c i.e 8% N content mgCOD/l <-> mgN/l = iXB, in ADM1 8.75%
      nbac = fnbac/14.d0*14000.d0
c i.e. 3.76% N content mgCOD/l <-> mgN/l
      nxc = fnxc/14.d0*14000.d0
c i.e. 9.8% N content mgCOD/l <-> mgN/l
      naa = fnaa/14.d0*14000.d0
c i.e. 6% N content mgCOD/l <-> mgN/l = iXP = iXI
      xni = fxni/14.d0*14000.d0
c i.e. 0% N content mgCOD/l <-> mgN/l = iSI
      sni = fsni/14.d0*14000.d0
      sni_adm = fsni_adm/14.d0*14000.d0
```

```
c lipid fraction of non-nitrogenous XS in BSM2
      frlixs = 0.7d0
c anaerobically degradable fraction of biomass in BSM2
      frxs = 0.68d0
c lipid fraction of non-nitrogenous biomass in BSM2
      frlixb = 0.4d0
c amount of XI and XP degradable in AD, zero in BSM2
      fdegrade = 0.d0

c Let CODdemand be the COD demand of available electron
c acceptors prior to the anaerobic digester, i.e. oxygen and nitrate
      CODdemand = asmm(8) + CODequiv*asmm(9)
      xtemp(8) = 0.d0
      xtemp(9) = 0.d0

c if extreme detail was used then some extra NH4 would be transformed
c into N bound in biomass and some biomass would be formed when
c removing the CODdemand (based on the yield). But on a total COD balance
c approach the below is correct (neglecting the N need for biomass growth)

      if (CODdemand .GT. asmm(2)) then
              remaina = CODdemand - asmm(2)
              xtemp(2) = 0.d0
              if (remaina .GT. asmm(4)) then
                      remainb = remaina - asmm(4)
                      xtemp(4) = 0.d0
                      if (remainb .GT. asmm(5)) then
                              remainc = remainb - asmm(5)
                              xtemp(10) = xtemp(10) + asmm(5)*fnbac
                              xtemp(5) = 0.d0
                              if (remainc .GT. asmm(6)) then
                                remaind = remainc - asmm(6)
                                xtemp(10) = xtemp(10) + asmm(6)*fnbac
                                xtemp(6) = 0.d0
c if here we run into problems, carbon shortage
                                xtemp(8) = remaind

c        disp('ERROR: COD shortage when removing inital oxygen and
nitrate')
                              else
                                xtemp(6) = asmm(6) - remainc
                                xtemp(10) = xtemp(10) + remainc*fnbac
                              endif
                      else
                              xtemp(5) = asmm(5) - remainb
                              xtemp(10) = xtemp(10) + remainb*fnbac
                      endif
              else
                      xtemp(4) = asmm(4) - remaina
              endif
      else
              xtemp(2) = asmm(2) - CODdemand
      endif

c SS becomes part of amino acids when transformed into ADM
c and any remaining SS is mapped to monosacharides (no N contents)
c Enough SND must be available for mapping to amino acids
c Saa COD equivalent to SND
      sorgn = asmm(11)/fnaa

      if (sorgn .GE. xtemp(2)) then
```

```
c not all SND-N fits into amino acids
c map all SS COD into Saa
              adm(2) = xtemp(2)
c the remaining part of SND
              xtemp(11) = xtemp(11) - xtemp(2)*fnaa
c all SS used
              xtemp(2) = 0.d0
      else
c all SND-N fits into amino acids
c map all SND related COD into Saa
              adm(2) = sorgn
c the rest of the SS COD, later mapped into sugar
              xtemp(2) = xtemp(2) - sorgn
c all SND used
              xtemp(11) = 0.d0
      endif

c XS becomes part of Xpr (proteins) when transformed into ADM
c and any remaining XS is mapped to Xch and Xli (no N contents)
c Enough XND must be available for mapping to Xpr

c Xpr COD equivalent to XND
      xorgn = asmm(12)/fnaa

      if (xorgn .GE. xtemp(4)) then
c not all XND-N fits into Xpr
c map all XS COD into Xpr
              xprtemp = xtemp(4)
c the remaining part of XND
              xtemp(12) = xtemp(12) - xtemp(4)*fnaa
c all XS used
              xtemp(4) = 0.d0
              xlitemp = 0.d0
              xchtemp = 0.d0
      else
c all XND-N fits into Xpr
c map all XND related COD into Xpr
              xprtemp = xorgn
c part of XS COD not associated with N
              xlitemp = frlixs*(xtemp(4) - xorgn)
c part of XS COD not associated with N
              xchtemp = (1.d0-frlixs)*(xtemp(4) - xorgn)
c all XS used
              xtemp(4) = 0.d0
c all XND used
              xtemp(12) = 0.d0
      endif

c Biomass becomes part of Xpr and XI when transformed into ADM
c and any remaining XBH and XBA is mapped to Xch and Xli (no N contents)
c Remaining XND-N can be used as nitrogen source to form Xpr
      biomass = xtemp(5) + xtemp(6)
c part which is mapped to XI
      biomass_nobio = biomass*(1.d0-frxs)
      biomass_bioN = (biomass*fnbac - biomass_nobio*fxni)
      if (biomass_bioN .LT. 0.d0) then
c    disp('ERROR: not enough biomass N to map the requested inert part')
            warn=1
      endif
      if ((biomass_bioN/fnaa) .LE. (biomass - biomass_nobio)) then
c all biomass N used
```

```
                xprtemp2 = biomass_bioN/fnaa
                remainCOD = biomass - biomass_nobio - xprtemp2
c use part of remaining XND-N to form proteins
                if ((xtemp(12)/fnaa) .GT. remainCOD)  then
                    xprtemp2 = xprtemp2 + remainCOD
                    xtemp(12) = xtemp(12) - remainCOD*fnaa
                    remainCOD = 0.d0
                    xtemp(5) = 0.d0
                    xtemp(6) = 0
c use all remaining XND-N to form proteins
            else
                    xprtemp2 = xprtemp2 + xtemp(12)/fnaa
                    remainCOD = remainCOD - xtemp(12)/fnaa
                    xtemp(12) = 0.d0
                endif
c part of the COD not associated with N
                xlitemp2 = frlixb*remainCOD
c part of the COD not associated with N
                xchtemp2 = (1.d0-frlixb)*remainCOD
      else
c all biomass COD used
          xprtemp2 = biomass - biomass_nobio
c any remaining N in XND
          xtemp(12) = xtemp(12) + biomass*fnbac - biomass_nobio*fxni
     &- xprtemp2*fnaa
       endif
       xtemp(5) = 0.d0
       xtemp(6) = 0.d0

c direct mapping of XI and XP to ADM1 XI
c assumption: same N content in both ASM1 and ADM1 particulate inerts
       inertX = (1-fdegrade)*(xtemp(3) + xtemp(7))

c special case: if part of XI and XP in ASM can be degraded in AD
c we have no knowledge about the contents so we put it in as composits (xc)
c we need to keep track of the associated nitrogen
c N content may be different, take first from XI&XP-N, then XND-N, then
SND-N,
c then SNH. Similar principle could be used for other states.
       xc = 0.d0
       xlitemp3 = 0.d0
       xchtemp3 = 0.d0
       if (fdegrade .GT. 0.d0) then
             noninertX = fdegrade*(xtemp(3) + xtemp(7))
c N in XI&XP(ASM) not enough
             if ((noninertX*fxni) .LT. (noninertX*fnxc)) then
                       xc = noninertX*fxni/fnxc
                       noninertX = noninertX - noninertX*fxni/fnxc
c N in XND not enough
                       if (xtemp(12) .LT. (noninertX*fnxc))  then
                           xc = xc + xtemp(12)/fnxc
                           noninertX = noninertX - xtemp(12)/fnxc
                           xtemp(12) = 0.d0
c N in SND not enough
                           if (xtemp(11) .LT. (noninertX*fnxc)) then
                              xc = xc + xtemp(11)/fnxc
                              noninertX = noninertX - xtemp(11)/fnxc
                              xtemp(11) = 0.d0
c N in SNH not enough
                              if (xtemp(10) .LT. (noninertX*fnxc))  then
                                  xc = xc + xtemp(10)/fnxc
```

```
                                noninertX = noninertX - xtemp(10)/fnxc
                                xtemp(10) = 0.d0
c  disp('ERROR: Nitrogen shortage when converting biodegradable XI&XP')
c   disp('Putting remaining XI&XP as lipids (50c) and carbohydrates (50%)')
                                xlitemp3 = 0.5d0*noninertX
                                xchtemp3 = 0.5d0*noninertX
                                noninertX = 0.d0
c N in SNH enough for mapping
                            else
                                xc = xc + noninertX
                                xtemp(10) = xtemp(10) - noninertX*fnxc
                                noninertX = 0.d0
                            endif
c N in SND enough for mapping
                        else
                            xc = xc + noninertX
                            xtemp(11) = xtemp(11) - noninertX*fnxc
                            noninertX = 0.d0
                        endif
c N in XND enough for mapping
                    else
                        xc = xc + noninertX
                        xtemp(12) = xtemp(12) - noninertX*fnxc
                        noninertX = 0.d0
                    endif
c N in XI&XP(ASM) enough for mapping
            else
                    xc = xc + noninertX
c put remaining N as XND
                    xtemp(12) = xtemp(12) + noninertX*(fxni-fnxc)
                    noninertX = 0.d0
            endif
      endif

c Mapping of ASM SI to ADM1 SI
c N content may be different, take first from SI-N, then SND-N, then XND-N,
c then SNH. Similar principle could be used for other states.
      inertS = 0.d0
c N in SI(ASM) not enough
      if ((xtemp(1)*fsni) .LT. (xtemp(1)*fsni_adm)) then
                inertS = xtemp(1)*fsni/fsni_adm
                xtemp(1) = xtemp(1) - xtemp(1)*fsni/fsni_adm
c N in SND not enough
                if (xtemp(11) .LT. (xtemp(1)*fsni_adm))  then
                    inertS = inertS + xtemp(11)/fsni_adm
                   xtemp(1) = xtemp(1) - xtemp(11)/fsni_adm
                    xtemp(11) = 0.d0
c N in XND not enough
                  if (xtemp(12) .LT. (xtemp(1)*fsni_adm)) then
                    inertS = inertS + xtemp(12)/fsni_adm
                    xtemp(1) = xtemp(1) - xtemp(12)/fsni_adm
                    xtemp(12) = 0.d0
c N in SNH not enough
                    if (xtemp(10) .LT. (xtemp(1)*fsni_adm))  then
                        inertS = inertS + xtemp(10)/fsni_adm
                        xtemp(1) = xtemp(1) - xtemp(10)/fsni_adm
                        xtemp(10) = 0.d0
c  disp('ERROR: Nitrogen shortage when converting SI')
c   disp('Putting remaining SI as monosacharides')
                        xtemp(2) = xtemp(2) + xtemp(1)
                        xtemp(1) = 0.d0
```

```
c N in SNH enough for mapping
                        else
                             inertS = inertS + xtemp(1)
                             xtemp(10) = xtemp(10) - xtemp(1)*fsni_adm
                             xtemp(1) = 0.d0
                        endif
c N in XND enough for mapping
                      else
                         inertS = inertS + xtemp(1)
                         xtemp(12) = xtemp(12) - xtemp(1)*fsni_adm
                         xtemp(1) = 0.d0
                      endif
c N in SND enough for mapping
                   else
                      inertS = inertS + xtemp(1)
                      xtemp(11) = xtemp(11) - xtemp(1)*fsni_adm
                      xtemp(1) = 0.d0
                   endif
c N in SI(ASM) enough for mapping
      else
                   inertS = inertS + xtemp(1)
c put remaining N as SND
                   xtemp(11) = xtemp(11) + xtemp(1)*(fsni-fsni_adm)
                   xtemp(1) = 0.d0
      endif

      adm(1) = xtemp(2)/1000.d0
      adm(2) = adm(2)/1000.d0
c      adm(10) = xtemp(13)/1000.d0
      adm(11) = (xtemp(10) + xtemp(11) + xtemp(12))/14000
      adm(12) = inertS/1000.d0
      adm(13) = xc/1000.d0
      adm(14) = (xchtemp + xchtemp2 + xchtemp3)/1000.d0
      adm(15) = (xprtemp + xprtemp2)/1000.d0
      adm(16) = (xlitemp + xlitemp2 + xlitemp3)/1000.d0
      adm(24) = (biomass_nobio + inertX)/1000.d0

C Calculation of adm(10) (Sic)
      alfachac=(-1.d00/64.d00)/(1.d00+10.d00**(4.76d00-ph))
      alfachpro=(-1.d00/112.d00)/(1.d00+10.d00**(4.88d00-ph))
      alfachbu=(-1.d00/160.d00)/(1.d00+10.d00**(4.82d00-ph))
      alfachva=(-1.d00/208.d00)/(1.d00+10.d00**(4.86d00-ph))
      pkk=-dlog10((10.d00**(-9.25d0))
     &      *dexp(51965.d00/bigr/100.d00*tfac))
      alfachin=(10.d00**(pkk-ph))/(1.d00+10.d00**(pkk-ph))
      pkk=-dlog10((10.d00**(-6.35d0))
     &      *dexp(7646.d00/bigr/100.d00*tfac))
      alfachic=-1.d00/(1.d00+10.d00**(pkk-ph))
c modifie / PV
      alfachnh=1.d00/14.d00
      alfachno=-1.d00/14.d00
      alfachalk=-1.d00
      chargeasm1=(asmm(13)*alfachalk+asmm(10)*alfachnh+
     &      asmm(9)*alfachno)/1000.
      chargeadm1=adm(7)*alfachac+adm(6)*alfachpro+
     &      adm(5)*alfachbu+adm(4)*alfachva+adm(11)*alfachin
      adm(10)=(chargeasm1-chargeadm1)/alfachic
      pkk=-dlog10((10.d00**(-14))*dexp(55900.d00/bigr/100.d00*tfac))
      ancat=chargeadm1+adm(10)*alfachic-10.d0**(-ph)+10.d0**(-pkk+ph)
c check mass balances
      totCODin = 0.d0
```

```
 do i=1,7
       totCODin=totCODin+asmm(i)
 enddo
 totNin = asmm(9) + asmm(10) + asmm(11) + asmm(12) +
&nbac/1000.d0*(asmm(5) + asmm(6)) + sni/1000.d0*asmm(1) +
&xni/1000.d0*(asmm(3) + asmm(7))

 totCODout=0.d0
 do i=1,9
       totCODout=totCODout+adm(i)*1000.d0
 enddo
 do i=12,24
       totCODout=totCODout+adm(i)*1000.d0
 enddo
 totNout = nbac*(adm(17)+adm(18)+adm(19)+adm(20)+adm(21)
&+adm(22)+adm(23)) + naa*(adm(2) + adm(15)) + adm(11)*14000.d0 +
&sni_adm*adm(12) + nxc*adm(13) + xni*adm(24)


 RETURN
 END
```

A2.2.2. ADM/ASM

```
      SUBROUTINE adm2asmv2(adm,asmm,totCODin,totCODout,totTKNin,
     &totTKNout,ph,warn)

      IMPLICIT NONE
      INTEGER i,warn
c ADM2ASM   Transformation model for conversion of ADM1 variables
c              into ASM1 variables.
c ADM2ASM(x) returns the 13 state variables of the ASM1. The
c input vector x represents the first 24 state variables of ADM1. A number
of
c parameters are required and are currently defined within this file. As
c many of them are also used in ASM1 and ADM1 they should probably only
c be defined in the initialisation files for those models to avoid
c different values if they are changed in one file and not in another.

c Parameters:
c fnaa          fraction of N in amino acids as in ADM1 report (default
0.098)
c fnxc          N content of composites adjusted from ADM1 report
c (default 0.0376)
c fnbac          N content of biomass based on BSM1 (default 0.08)
c fsni_adm           N content of soluble inerts
c fnbac and fsni_adm are adjusted to fit the benchmark values of iXB=0.08
and
c iXP=0.06.
c fxni      N content of inerts XI and XP, same in AS and AD
c fsni = N content of SI, zero in ASM and BSM2
c nbac= fnbac/14*14000 conversion into kmol N/m3 from mg N/l for biomass
c nxb = fnxb/14*14000   conversion into kmol N/m3 from mg N/l for biomass
c nxc = fnxc/14*14000   conversion into kmol N/m3 from mg N/l for
composites
c naa = fnaa/14*14000   conversion into kmol N/m3 from mg N/l for amino
acids
c ni = fni/14*14000        conversion into kmol N/m3 from mg N/l for
inerts
      DOUBLE PRECISION fnaa,fnxc,fnbac,fsni_adm,nxc,naa,ni
      DOUBLE PRECISION iXI,fxni,fsni,nbac,sni_adm,xni,sni
      DOUBLE PRECISION frxs_AS,fdegrade_AS

c adm is the input in ADM1 terminology, 24 variables
c adm[1] : Ssu = monosacharides (kg COD/m3)
c adm[2] : Saa = amino acids (kg COD/m3)
c adm[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3)
c adm[4] : Sva = total valerate (kg COD/m3)
c adm[5] : Sbu = total butyrate (kg COD/m3)
c adm[6] : Spro = total propionate (kg COD/m3)
c adm[7] : Sac = total acetate (kg COD/m3)
c adm[8] : Sh2 = hydrogen gas (kg COD/m3)
c adm[9] : Sch4 = methane gas (kg COD/m3)
c adm[10] : Sic = inorganic carbon (kmole C/m3)
c adm[11] : Sin = inorganic nitrogen (kmole N/m3)
c adm[12] : Sinert = soluble inerts (kg COD/m3)
c adm[13] : Xc = composites (kg COD/m3)
c adm[14] : Xch = carbohydrates (kg COD/m3)
c adm[15] : Xpr = proteins (kg COD/m3)
c adm[16] : Xli = lipids (kg COD/m3)
c adm[17] : Xsu = sugar degraders (kg COD/m3)
c adm[18] : Xaa = amino acid degraders (kg COD/m3)
```

```
c adm[19] : Xfa = LCFA degraders (kg COD/m3)
c adm[20] : Xc4 = valerate and butyrate degraders (kg COD/m3)
c adm[21] : Xpro = propionate degraders (kg COD/m3)
c adm[22] : Xac = acetate degraders (kg COD/m3)
c adm[23] : Xh2 = hydrogen degraders (kg COD/m3)
c adm[24] : xinert = particulate inerts (kg COD/m3)
      DOUBLE PRECISION adm(24)
c
c asmm is the output in ASM1 terminology, 13 variables
c asmm[1] : Si = soluble inert organic material (g COD/m3)
c asmm[2] : Ss = readily biodegradable substrate (g COD/m3)
c asmm[3] : Xi = particulate inert organic material (g COD/m3)
c asmm[4] : Xs = slowly biodegradable substrate (g COD/m3)
c asmm[5] : Xbh = active heterotrophic biomass (g COD/m3)
c asmm[6] : Xba = active autotrophic biomass (g COD/m3)
c asmm[7] : Xp = particulate product arising from biomass decay (g COD/m3)
c asmm[8] : So = oxygen (g -COD/m3)
c asmm[9] : Sno = nitrate and nitrite nitrogen (g N/m3)
c asmm[10] : Snh = ammonia and ammonium nitrogen (g N/m3)
c asmm[11] : Snd = soluble biogradable organic nitrogen (g N/m3)
c asmm[12] : Xnd = particulate biogradable organic nitrogen (g N/m3)
c asmm[13] : Salk = alkalinity (mole HCO3-/m3)
      DOUBLE PRECISION asmm(13)

      DOUBLE PRECISION xtemp(24)

      DOUBLE PRECISION biomass,biomass_nobio,biomass_bioN
      DOUBLE PRECISION remainCOD,xptemp,xstemp
      DOUBLE PRECISION totCODout,totTKNout,totCODin,totTKNin,ph
      DOUBLE PRECISION inertX,xstemp2,noninertX
      DOUBLE PRECISION inertS

      DOUBLE PRECISION alfachac, alfachpro,alfachbu,alfachva
      DOUBLE PRECISION alfachin,alfachic,alfachnh
      DOUBLE PRECISION alfachalk,alfachno
      DOUBLE PRECISION chargeasm1,chargeadm1,pkk,tfac,bigr

      tfac=1.d00/298.15d00-1.d00/(273.15d00+35.d00)
      bigr=0.08314d-00
c ADM1 --> ASM1
c Copyright: John Copp, Hydromantis Inc. and Ulf Jeppsson, Lund University

      do i=1,13
            asmm(i)=0.d0
      enddo

c Set parameter values
      fnaa = 0.098d0
      fnxc = 0.0376d0
      fnbac = 0.08d0
      fsni_adm = 0.06d0
      fxni=0.06d00
      fsni=0.d00
      nbac=fnbac/14.d0*14000.d0
      nxc = fnxc/14.d0*14000
      naa = fnaa/14.d0*14000.d0
      sni_adm = fsni_adm/14.d0*14000.d0
      xni=fxni/14.d0*14000.d0
      sni=fsni/14.d0*14000.d0
      frxs_AS=0.79d0
      fdegrade_AS=0.d00
```

```
      warn=0

      do i=1,24
            xtemp(i)=adm(i)
      enddo
c Biomass becomes part of XS and XP when transformed into ASM
c Assume Npart of formed XS to be fnxc and Npart of XP to be fxni
c Remaining N goes into ammonia pool
      biomass=0.d0
      do i=17,23
            biomass = biomass+xtemp(i)*1000.d0
      enddo
c Part of biomass mapped into XP
      biomass_nobio=biomass*(1.d0-frxs_AS)
      biomass_bioN=biomass*fnbac-biomass_nobio*fxni
      remainCOD=0.d0
      if(biomass_bioN.LT.0.d0) then
            warn=1
            xptemp=biomass*fnbac/fxni
            biomass_bioN=0.d0
      else
            xptemp=biomass_nobio
      end if

      if((biomass_bioN/fnxc).LE.(biomass-biomass_nobio)) then
c all biomass N used
            xstemp=biomass_bioN/fnxc
            remainCOD=biomass-biomass_nobio-xstemp
c use part of remaining S_IN to form XS
            if((xtemp(11)*14000.d0/fnxc).GT.remainCOD) then
                  xstemp=xstemp+remainCOD
            else
                  write(*,*) 'System failure'
                  warn=2
            end if
      else
c all biomass COD used
            xstemp=biomass-biomass_nobio
      end if

c Any remaining N in S_IN
      xtemp(11)=xtemp(11)+biomass*fnbac/14000.d0 -xptemp*fxni/14000.d0
     &-xstemp*fnxc/14000.d0

c Xs = all X from ADM except Xi + biomass
      asmm(4)=xstemp
      do i=13,16
            asmm(4)=asmm(4)+xtemp(i)*1000.
      end do

c inert part of biomass
      asmm(7)=xptemp

c mapping of intert XI in AD into XI and possibly XS in AS
c Assumption: same N content in both ASM1 and ADM1 particulate inerts
c Special case: if part of XI in AD can be degraded in AS
c We have no knowledge about the contents so we put it in as part substrate
(XS)
c We need to keep track of the associated nitrogen
c N content might be different, take first from XI-N then S_IN
```

```
      inertX=(1.d0-fdegrade_AS)*xtemp(24)*1000.d0
      xstemp2=0.d0
      noninertX=0.d0

      if(fdegrade_AS.GT.0.d0) then
            noninertX=fdegrade_AS*xtemp(24)*1000.d0
c N in XI(AD) not enough
            if(fxni.LT.fnxc) then
                  xstemp2=noninertX*fxni/fnxc
                  noninertX=noninertX-noninertX*fxni/fnxc
                  if((xtemp(11)*14000.d0).LT.(noninertX*fnxc)) then
c N in SNH not enough
                        xstemp2=xstemp2+xtemp(11)*14000.d0/fnxc
                        noninertX=noninertX-xtemp(11)*14000.d0/fnxc
                        xtemp(11)=0.d0
                        warn=3
                        inertX=inertX+noninertX
c N in S_IN enough for mapping
                  else
                        xstemp2=xstemp2+noninertX
                        xtemp(11)=xtemp(11)-noninertX*fnxc/14000.d0
                        noninertX=0.d0
                  end if
c N in XI(AD) enough for mapping
            else
                  xstemp2=xstemp2+noninertX
c Put remaining N as S_IN
                  xtemp(11)=xtemp(11)+noninertX*(fxni-fnxc)/14000.d0
                  noninertX=0.d0
            end if
      end if

c Xi = Xi*fdegrade_AS + possibly nonmappable XS
      asmm(3)=inertX
c extra added XS (biodegradable XI)
      asmm(4)=asmm(4)+xstemp2

c mapping of ADM SI into ASM1 SI
c N content may be different, take first from SI_N then S_IN
      inertS=0.d0
c N in SI(AD) not enough
      if(fsni_adm.LT.fsni) then
            inertS=xtemp(12)*fsni_adm/fsni
            if((xtemp(11)*14.d0).LT.(xtemp(12)*fsni)) then
c N in S_IN not enough
                  inertS=inertS+xtemp(11)*14.d0/fsni
                  xtemp(12)=xtemp(12)-xtemp(11)*14.d0/fsni
                  xtemp(11)=0.d0
                  warn=5
c N in S_IN enough for mapping
            else
                  inertS=inertS+xtemp(12)
                  xtemp(11)=xtemp(11)-xtemp(12)*fsni/14.d0
                  xtemp(12)=0.d0
            end if
c N in SI(AD) enough for mapping
      else
            inertS=inertS+xtemp(12)
c Put remaining N as S_IN
            xtemp(11)=xtemp(11)+xtemp(12)*(fsni_adm-fsni)/14.d0
```

```
            xtemp(12)=0.d0
      end if
C Si as SI
      asmm(1)=inertS*1000.

c nitrogen in biomass, composites, proteins
c Xnd is the nitrogen part of XS in ASM1. Should be based on
c the same variables as constitutes XS, xc and xpo (no nitrogen in lipids
and carbohydrates)
      asmm(12)=fnxc*(xstemp+xstemp2)+nxc*xtemp(13)+naa*xtemp(15)


c sh2=x(8) and sch4=x(9) assumed to be stripped upon reentry to ASM side
c Ss = sum of all solubles except Sh2, Sch4, Si, Sic, Sin
      do i=1,7
      asmm(2) = asmm(2)+xtemp(i)*1000.d0
      enddo

c Snd is the nitrogen part of Ss in ASM1. Therefore Snd should be based on
the
c same variables as constitutes Ss, and we assume
c there is only nitrogen in the amino acids. The N content of Si is
c not included in ASM1
      asmm(11) = naa*xtemp(2)

c Snh = Sin including adjustments above
      asmm(10) = xtemp(11)*14000.d0
C Calculation of Salk
      alfachac=(-1.d00/64.d00)/(1.d00+10.d00**(4.76d00-ph))
      alfachpro=(-1.d00/112.d00)/(1.d00+10.d00**(4.88d00-ph))
      alfachbu=(-1.d00/160.d00)/(1.d00+10.d00**(4.82d00-ph))
      alfachva=(-1.d00/208.d00)/(1.d00+10.d00**(4.86d00-ph))
      pkk=-dlog10((10.d00**(-9.25d0))
     &      *dexp(51965.d00/bigr/100.d00*tfac))
      alfachin=(10.d00**(pkk-ph))/(1.d00+10.d00**(pkk-ph))
      pkk=-dlog10((10.d00**(-6.35d0))
     &      *dexp(7646.d00/bigr/100.d00*tfac))
      alfachic=-1.d00/(1.d00+10.d00**(pkk-ph))
c modifie / PV
      alfachnh=1.d00/14.d00
      alfachno=-1.d00/14.d00
      alfachalk=-1.d00
      chargeasm1=asmm(10)*alfachnh+asmm(9)*alfachno
      chargeadm1=(adm(7)*alfachac+adm(6)*alfachpro+adm(11)*alfachin
     &      +adm(5)*alfachbu+adm(4)*alfachva+adm(10)*alfachic)*1000.
      asmm(13)=chargeasm1-chargeadm1
c check mass balances
      totCODin=0.d0
      do i=1,7
            totCODin=totCODin + adm(i)
      enddo
      do i=12,24
            totCODin = totCODin + adm(i)*1000.d0
      enddo
      totTKNin=0.d0
      do i=17,23
            totTKNin=totTKNin+nbac*adm(i)
      enddo
      totTKNin = totTKNin + nxc*adm(13) + naa*adm(15) + naa*adm(2) +
     & adm(11)*14000.d0 + sni_adm*adm(12) + xni*adm(24)
```

```
      totCODout=0.d0
      do i=1,7
            totCODout = totCODout + asmm(i)
      enddo
c!!!Note SI_N not included here
      totTKNout = asmm(10) + asmm(11) + asmm(12) + fsni*asmm(1)+
     &fnbac*(asmm(5) + asmm(6)) + fxni*(asmm(3) + asmm(7))
      return
      end
```

APPENDIX 3: Steady-state results

These results were obtained with FORTRAN (integration using a Runge-Kutta 4 algorithm with a constant integration step = 0.005 h) and MATLAB-Simulink (Solver: ode45, absolute tolerance =10^{-8}, relative tolerance =10^{-5}). For details on operation conditions see text.

CONCENTRATION values	INPUT WASTEWATER	EFFLUENT WATER		
Input variables raw wastewater	Values	**Output variables**	FORTRAN	MATLAB
S_I (g COD.m^{-3})	27.22619062	S_I (g COD.m^{-3})	28.064300	28.0643
S_S (g COD.m^{-3})	58.17618568	S_S (g COD.m^{-3})	0.671945	0.67336
X_I (g COD.m^{-3})	92.49900106	X_I (g COD.m^{-3})	5.918820	5.9191
X_S (g COD.m^{-3})	363.943473	X_S (g COD.m^{-3})	0.123290	0.12329
X_{BH} (g COD.m^{-3})	50.68328815	X_{BH} (g COD.m^{-3})	8.662210	8.6614
X_{BA} (g COD.m^{-3})	0	X_{BA} (g COD.m^{-3})	0.648923	0.6484
X_P (g COD.m^{-3})	0	X_P (g COD.m^{-3})	3.748460	3.7485
S_O (g -COD.m^{-3})	0	S_O (g -COD.m^{-3})	1.373160	1.3748
S_{NO} (g N.m^{-3})	0	S_{NO} (g N.m^{-3})	9.194400	9.1948
S_{NH} (g N.m^{-3})	23.85946563	S_{NH} (g N.m^{-3})	0.158835	0.15845
S_{ND} (g N.m^{-3})	5.651606031	S_{ND} (g N.m^{-3})	0.560279	0.55943
X_{ND} (g N.m^{-3})	16.12981606	X_{ND} (g N.m^{-3})	0.009243	0.0092428
S_{ALK} (mole HCO_3.m^{-3})	7	S_{ALK} (mole HCO_3^-.m^{-3})	4.564060	4.5646
TSS (g.m^{-3})	380.3443217	*TSS* (g.m^{-3})	14.326277	14.3255
Q (m^3.d^{-1})	20648.36121	Q (m^3.d^{-1})	20640.768000	20640.7792
T (°C)	14.85808006	T (°C)	14.8581	14.8581

LOAD values		EFFLUENT WATER		
Input variables raw wastewater	Values	**Output variables**	FORTRAN	MATLAB
S_I (kg COD.d^{-1})	562.1762184	S_I (kg COD.d^{-1})	579.2687054	579.2690197
S_S (kg COD.d^{-1})	1201.242896	S_S (kg COD.d^{-1})	13.86946085	13.89867508
X_I (kg COD.d^{-1})	1909.952785	X_I (kg COD.d^{-1})	122.1689905	122.1748362
X_S (kg COD/d)	7514.836291	X_S (kg COD.d^{-1})	2.544800287	2.544801668
X_{BH} (kg COD.d^{-1})	1046.526841	X_{BH} (kg COD.d^{-1})	178.794667	178.778045
X_{BA} (kg COD.d^{-1})	0	X_{BA} (kg COD.d^{-1})	13.39426909	13.38348123
X_P (kg COD.d^{-1})	0	X_P (kg COD.d^{-1})	77.37109322	77.37196083
S_O (kg -COD.d^{-1})	0	S_O (kg -COD.d^{-1})	28.34307699	28.37694324
S_{NO} (kg N.d^{-1})	0	S_{NO} (kg N.d^{-1})	189.7794773	189.7878366
S_{NH} (kg N.d^{-1})	492.6588647	S_{NH} (kg N.d^{-1})	3.278476385	3.270531464
S_{ND} (kg N.d^{-1})	116.6964028	y_{ND} (kg N.d^{-1})	11.56458885	11.54707111
X_{ND} (kg N.d^{-1})	333.0542683	X_{ND} (kg N.d^{-1})	0.190784063	0.190778594
S_{ALK} (kmole HCO_3^-.d^{-1})	144.5385285	S_{ALK} (kmole HCO_3^-.d^{-1})	94.2057036	94.21690074
TSS (kg.d^{-1})	7853.486938	*TSS* (kg.d^{-1})	295.705365	295.6894824

CONCENTRATION values		SLUDGE FOR DISPOSAL		
Input variables raw wastewater	Values	**Output variables**	FORTRAN	MATLAB
S_I (g COD.m^{-3})	27.22619062	S_I (g COD.m^{-3})	130.863000	130.8682
S_S (g COD.m^{-3})	58.17618568	S_S (g COD.m^{-3})	262.103000	258.5788
X_I (g COD.m^{-3})	92.49900106	X_I (g COD.m^{-3})	314241.000000	314238.8343
X_S (g COD.m^{-3})	363.943473	X_S (g COD.m^{-3})	47664.900000	47667.026
X_{BH} (g COD.m^{-3})	50.68328815	X_{BH} (g COD.m^{-3})	0.000000	0
X_{BA} (g COD.m^{-3})	0	X_{BA} (g COD.m^{-3})	0.000000	0
X_P (g COD.m^{-3})	0	X_P (g COD.m^{-3})	11426.900000	11427.473
S_O (g -COD.m^{-3})	0	S_O (g -COD.m^{-3})	0.000000	0
S_{NO} (g N.m^{-3})	0	S_{NO} (g N.m^{-3})	0.000000	0
S_{NH} (g N.m^{-3})	23.85946563	S_{NH} (g N.m^{-3})	1443.440000	1442.7931
S_{ND} (g N.m^{-3})	5.651606031	S_{ND} (g N.m^{-3})	0.543236	0.54232
X_{ND} (g N.m^{-3})	16.12981606	X_{ND} (g N.m^{-3})	1841.030000	1841.1071
S_{ALK} (mole HCO_3.m^{-3})	7	S_{ALK} (mole HCO_3.m^{-3})	97.823400	97.8462
TSS (g.m^{-3})	380.3443217	*TSS* (g.m^{-3})	279999.600000	280000
Q (m^3.d^{-1})	20648.36121	Q (m^3.d^{-1})	9.582480	9.582
T (°C)	14.85808006	T (°C)	14.858100	14.8581

LOAD values		SLUDGE FOR DISPOSAL	Calculated automatically	
Input variables raw wastewater	Values	**Output variables**	FORTRAN	MATLAB
S_I (kg COD.d^{-1})	562.1762184	S_I (kg COD.d^{-1})	1.25399208	1.253979092
S_S (kg COD.d^{-1})	1201.242896	S_S (kg COD.d^{-1})	2.511596755	2.477702062
X_I (kg COD.d^{-1})	1909.952785	X_I (kg COD.d^{-1})	3011.208098	3011.03651
X_S (kg COD.d^{-1})	7514.836291	X_S (kg COD.d^{-1})	456.747951	456.7454431
X_{BH} (kg COD.d^{-1})	1046.526841	X_{BH} (kg COD.d^{-1})	0	0
X_{BA} (kg COD.d^{-1})	0	X_{BA} (kg COD.d^{-1})	0	0
X_P (kg COD.d^{-1})	0	X_P (kg COD.d^{-1})	109.4980407	109.4980463
S_O (kg -COD.d^{-1})	0	S_O (kg -COD.d^{-1})	0	0
S_{NO} (kg N.d^{-1})	0	S_{NO} (kg N.d^{-1})	0	0
S_{NH} (kg N.d^{-1})	492.6588647	S_{NH} (kg N.d^{-1})	13.83173493	13.82484348
S_{ND} (kg N.d^{-1})	116.6964028	S_{ND} (kg N.d^{-1})	0.005205548	0.00519651
X_{ND} (kg N.d^{-1})	333.0542683	X_{ND} (kg N.d^{-1})	17.64163315	17.64148823
S_{ALK} (kmole HCO_3.d^{-1})	144.5385285	S_{ALK} (kmole HCO_3^-.d^{-1})	0.937390774	0.937562288
TSS (kg.d^{-1})	7853.486938	*TSS* (kg.d^{-1})	2683.090567	2682.96

CONCENTRATION values — PRIMARY EFFLUENT

Output variables	FORTRAN	MATLAB
S_I (g COD.m^{-3})	28.067000	28.067
S_S (g COD.m^{-3})	59.075600	59.0473
X_I (g COD.m^{-3})	49.336300	49.3362
X_S (g COD.m^{-3})	186.585000	186.5845
X_{BH} (g COD.m^{-3})	26.611600	26.6115
X_{BA} (g COD.m^{-3})	0.049527	0.0495
X_P (g COD.m^{-3})	0.341502	0.3415
S_O (g -COD.m^{-3})	0.017526	0.0175
S_{NO} (g N.m^{-3})	0.117352	0.1174
S_{NH} (g N.m^{-3})	34.926900	34.9215
S_{ND} (g N.m^{-3})	5.545710	5.5457
X_{ND} (g N.m^{-3})	8.268320	8.2683
S_{ALK} (mole HCO_3^-.m^{-3})	7.696350	7.6965
TSS (g.m^{-3})	197.192947	197.1925
Q (m^3.d^{-1})	20938.776000	20939
T (°C)	14.858100	14.8581

CONCENTRATION values — PRIMARY UNDERFLOW

Output variables	FORTRAN	MATLAB
S_I (g COD.m^{-3})	28.067000	28.0672
S_S (g COD.m^{-3})	59.075600	59.0473
X_I (g COD.m^{-3})	6480.700000	6480.7
X_S (g COD.m^{-3})	24509.300000	24509
X_{BH} (g COD.m^{-3})	3495.640000	3495.6
X_{BA} (g COD.m^{-3})	6.505740	6.5001
X_P (g COD.m^{-3})	44.858800	44.8571
S_O (g -COD.m^{-3})	0.017526	0.0175
S_{NO} (g N.m^{-3})	0.117352	0.1174
S_{NH} (g N.m^{-3})	34.926900	34.9215
S_{ND} (g N.m^{-3})	5.545710	5.5457
X_{ND} (g N.m^{-3})	1086.110000	1086.1
S_{ALK} (mole HCO_3^-.m^{-3})	7.696350	7.6965
TSS (g.m^{-3})	25902.753405	25903
Q (m^3.d^{-1})	147.604800	147.6047
T (°C)	14.858100	14.8581

CONCENTRATION values

ANOXIC REACTOR NO 2

Output variables	FORTRAN	MATLAB
S_I (g COD.m^{-3})	28.064300	28.0643
S_S (g COD.m^{-3})	1.337900	1.3412
X_I (g COD.m^{-3})	1532.270000	1532.3
X_S (g COD.m^{-3})	58.863100	58.8579
X_{BH} (g COD.m^{-3})	2245.750000	2245.4
X_{BA} (g COD.m^{-3})	166.697000	166.5512
X_P (g COD.m^{-3})	965.720000	965.6773
S_O (g -COD.m^{-3})	0.000109	0.00010907
S_{NO} (g N.m^{-3})	2.219260	2.2207
S_{NH} (g N.m^{-3})	7.203620	7.2028
S_{ND} (g N.m^{-3})	0.687265	0.6862
X_{ND} (g N.m^{-3})	3.742710	3.7424
S_{ALK} (mole HCO_3^-.m^{-3})	5.565480	5.5659
TSS (g.m^{-3})	3726.975075	3726.5
Q (m^3.d^{-1})		103532.78
T (°C)	14.856600	14.8581

CONCENTRATION values

AEROBIC REACTOR NO 2

Output variables	FORTRAN	MATLAB
S_I (g COD.m^{-3})	28.064300	28.0643
S_S (g COD.m^{-3})	0.778875	0.7806
X_I (g COD.m^{-3})	1532.270000	1532.3
X_S (g COD.m^{-3})	37.391100	37.388
X_{BH} (g COD.m^{-3})	2245.990000	2245.6
X_{BA} (g COD.m^{-3})	167.980000	167.8339
X_P (g COD.m^{-3})	968.847000	968.804
S_O (g -COD.m^{-3})	1.427790	1.4284
S_{NO} (g N.m^{-3})	8.405400	8.4066
S_{NH} (g N.m^{-3})	0.693511	0.6922
S_{ND} (g N.m^{-3})	0.610313	0.6094
X_{ND} (g N.m^{-3})	2.681730	2.6815
S_{ALK} (mole HCO_3.m^{-3})	4.658610	4.659
TSS (g.m^{-3})	3714.358575	3713.9
Q (m^3.d^{-1})		103532.78
T (°C)	14.856600	14.8581

CONCENTRATION values	SECONDARY WASTAGE SLUDGE		
	Output variables	FORTRAN	MATLAB
	S_I (g COD.m^{-3})	28.064300	28.0643
	S_S (g COD.m^{-3})	0.671945	0.6734
	X_I (g COD.m^{-3})	3036.240000	3036.2
	X_S (g COD.m^{-3})	63.245400	63.2392
	X_{BH} (g COD.m^{-3})	4443.550000	4442.8
	X_{BA} (g COD.m^{-3})	332.885000	332.5957
	X_P (g COD.m^{-3})	1922.890000	1922.8
	S_O (g -COD.m^{-3})	1.373160	1.3748
	S_{NO} (g N.m^{-3})	9.194400	9.1948
	S_{NH} (g N.m^{-3})	0.158835	0.1585
	S_{ND} (g N.m^{-3})	0.560279	0.5594
	X_{ND} (g N.m^{-3})	4.741520	4.7411
	S_{ALK} (mole HCO_3^-.m^{-3})	4.564060	4.5646
	TSS (g.m^{-3})	7349.107800	7348.3
	Q (m^3.d^{-1})	300.000000	300
	T (°C)	14.856600	14.8581

CONCENTRATION values	THICKENER EFFLUENT		
	Output variables	FORTRAN	MATLAB
	S_I (g COD.m^{-3})	28.064300	28.0643
	S_S (g COD.m^{-3})	0.671945	0.6734
	X_I (g COD.m^{-3})	67.689300	67.6876
	X_S (g COD.m^{-3})	1.409980	1.4098
	X_{BH} (g COD.m^{-3})	99.063400	99.0462
	X_{BA} (g COD.m^{-3})	7.421270	7.4147
	X_P (g COD.m^{-3})	42.868400	42.8659
	S_O (g -COD.m^{-3})	1.373160	1.3748
	S_{NO} (g N.m^{-3})	9.194400	9.1948
	S_{NH} (g N.m^{-3})	0.158835	0.1585
	S_{ND} (g N.m^{-3})	0.560279	0.5594
	X_{ND} (g N.m^{-3})	0.105706	0.1057
	S_{ALK} (mole HCO_3^-.m^{-3})	4.564060	4.5646
	TSS (g.m^{-3})	163.839263	163.8182
	Q (m^3.d^{-1})	269.133600	269.1373
	T (°C)	14.856600	14.8581

CONCENTRATION values	THICKENER UNDERFLOW		
	Output variables	FORTRAN	MATLAB
	S_I (g COD.m^{-3})	28.064300	28.0643
	S_S (g COD.m^{-3})	0.671945	0.6734
	X_I (g COD.m^{-3})	28920.100000	28923
	X_S (g COD.m^{-3})	602.410000	602.4207
	X_{BH} (g COD.m^{-3})	42324.600000	42323
	X_{BA} (g COD.m^{-3})	3170.720000	3168.3
	X_P (g COD.m^{-3})	18315.400000	18317
	S_O (g -COD.m^{-3})	1.373160	1.3748
	S_{NO} (g N.m^{-3})	9.194400	9.1948
	S_{NH} (g N.m^{-3})	0.158835	0.1585
	S_{ND} (g N/m^3)	0.560279	0.5594
	X_{ND} (g N.m^{-3})	45.162800	45.1637
	S_{ALK} (mole HCO_3^-.m^{-3})	4.564060	4.5646
	TSS (g.m^{-3})	69999.922500	70000
	Q (m^3.d^{-1})	30.866160	30.8627
	T (°C)	14.856600	14.8581

CONCENTRATION values

ASM2ADM OUTPUT

Output variables	FORTRAN	MATLAB
S_{su} (kg COD.m^{-3})	0.000000	0
S_{aa} (kg COD.m^{-3})	0.043902	0.0439
S_{fa} (kg COD.m^{-3})	0.000000	0
S_{va} (kg COD.m^{-3})	0.000000	0
S_{bu} (kg COD.m^{-3})	0.000000	0
S_{pro} (kg COD.m^{-3})	0.000000	0
S_{ac} (kg COD.m^{-3})	0.000000	0
S_{h2} (kg COD.m^{-3})	0.000000	0
S_{ch4} (kg COD.m^{-3})	0.000000	0
S_{IC} (kmole C.m^{-3})	0.007918	0.0079
S_{IN} (kmole N.m^{-3})	0.001972	0.002
S_{I} (kg COD.m^{-3})	0.028067	0.0281
X_{c} (kg COD.m^{-3})	0.000000	0
X_{ch} (kg COD.m^{-3})	3.723580	3.7236
X_{pr} (kg COD.m^{-3})	15.924300	15.9235
X_{li} (kg COD.m^{-3})	8.046860	8.047
X_{su} (kg COD.m^{-3})	0.000000	0
X_{aa} (kg COD.m^{-3})	0.000000	0
X_{fa} (kg COD.m^{-3})	0.000000	0
X_{c4} (kg COD.m^{-3})	0.000000	0
X_{pro} (kg COD.m^{-3})	0.000000	0
X_{ac} (kg COD.m^{-3})	0.000000	0
X_{h2} (kg COD.m^{-3})	0.000000	0
X_{I} (kg COD.m^{-3})	17.011000	17.0106
S_{cat+} (kmole.m^{-3})	0.000000	0
S_{an}- (kmole.m^{-3})	0.005210	0.0052
TSS (kg.m^{-3})	Not used in ADM	Not used in ADM
Q (m^{3}.d^{-1})	178.471000	178.4674
T (°C)	35.000000	35

CONCENTRATION values

DIGESTER OUTPUT

Output variables	FORTRAN	MATLAB
S_{su} (kg COD.m^{-3})	0.012395	0.0124
S_{aa} (kg COD.m^{-3})	0.005543	0.0055
S_{fa} (kg COD.m^{-3})	0.107409	0.1074
S_{va} (kg COD.m^{-3})	0.012332	0.0123
S_{bu} (kg COD.m^{-3})	0.014003	0.014
S_{pro} (kg COD.m^{-3})	0.017584	0.0176
S_{ac} (kg COD.m^{-3})	0.092837	0.0893
S_{h2} (kg COD.m^{-3})	0.000000	2.51E-07
S_{ch4} (kg COD.m^{-3})	0.055759	0.0555
S_{IC} (kmole C.m^{-3})	0.094865	0.0951
S_{IN} (kmole N.m^{-3})	0.094515	0.0945
S_{I} (kg COD.m^{-3})	0.130863	0.1309
X_{c} (kg COD.m^{-3})	0.107919	0.1079
X_{ch} (kg COD.m^{-3})	0.020517	0.0205
X_{pr} (kg COD.m^{-3})	0.084226	0.0842
X_{li} (kg COD.m^{-3})	0.043629	0.0436
X_{su} (kg COD.m^{-3})	0.312223	0.3122
X_{aa} (kg COD.m^{-3})	0.931720	0.9317
X_{fa} (kg COD.m^{-3})	0.338388	0.3384
X_{c4} (kg COD.m^{-3})	0.335788	0.3358
X_{pro} (kg COD.m^{-3})	0.101124	0.1011
X_{ac} (kg COD.m^{-3})	0.677136	0.6772
X_{h2} (kg COD.m^{-3})	0.284846	0.2848
X_{I} (kg COD.m^{-3})	17.216600	17.2161
S_{cat+} (kmole.m^{-3})	0.000000	0
S_{an}- (kmole.m^{-3})	0.005210	0.0052
TSS (kg.m^{-3})	not used in ADM	not used in ADM
Q (m^3.d^{-1})	178.471000	178.4674
T (°C)	35.000000	35
pH	7.270000	7.2631
$S_{gas,h2}$ (kg COD.m^{-3})	0.000011	1.10E-05
$S_{gas,ch4}$ (kg COD.m^{-3})	1.662680	1.6535
$S_{gas,co2}$ (kmole C.m^{-3})	0.013276	0.0135
$p_{gas,h2}$ (bar)	0.000018	1.77^{E}-05
$p_{gas,ch4}$ (bar)	0.665582	0.6619
$p_{gas,co2}$ (bar)	0.340125	0.3469
$P_{gas,total}$ (bar)	1.061890	1.0645
Q_{gas} (m^3.d^{-1})	2686.256739	2708.3

P_{atm} (bar)	1.013
$p_{gas,h2o}$ (bar)	T-dependent
	H_2+CH_4+CO_2+H_2O normalized to P_{atm}

CONCENTRATION values

ADM2ASM OUTPUT

Output variables	FORTRAN	MATLAB
S_I (g COD.m^{-3})		130.8674
S_S (g COD.m^{-3})		258.5789
X_I (g COD.m^{-3})		17216
X_S (g COD.m^{-3})		2611.5
X_{BH} (g COD.m^{-3})		0
X_{BA} (g COD.m^{-3})		0
X_P (g COD.m^{-3})		626.0654
S_O (g -COD.m^{-3})		0
S_{NO} (g N.m^{-3})		0
S_{NH} (g N.m^{-3})		1442.8
S_{ND} (g N.m^{-3})		0.5432
X_{ND} (g N.m^{-3})		100.8669
S_{ALK} (mole HCO_3^-.m^{-3})		97.846
TSS (g.m^{-3})		15340
Q (m^3.d^{-1})		178.4674
T (°C)		14.8581

CONCENTRATION values

DEWATERING EFFLUENT

Output variables	FORTRAN	MATLAB
S_I (g COD.m^{-3})	130.863000	130.8674
S_S (g COD.m^{-3})	262.103000	258.5789
X_I (g COD.m^{-3})	363.869000	363.8587
X_S (g COD.m^{-3})	55.192600	55.1931
X_{BH} (g COD.m^{-3})	0.000000	0
X_{BA} (g COD.m^{-3})	0.000000	0
X_P (g COD.m^{-3})	13.231600	13.2317
S_O (g -COD.m^{-3})	0.000000	0
S_{NO} (g N.m^{-3})	0.000000	0
S_{NH} (g N.m^{-3})	1443.440000	1442.8
S_{ND} (g N.m^{-3})	0.543236	0.5432
X_{ND} (g N.m^{-3})	2.131780	2.1318
S_{ALK} (mole HCO_3.m^{-3})	97.823400	97.846
TSS (g.m^{-3})	324.219900	324.2126
Q (m^3.d^{-1})	168.888480	168.8853
T (°C)	14.858100	14.8581

Effluent average concentrations based on load			
Variable	**Unit**	FORTRAN	MATLAB
Effluent average flow rate	$m^3.d^{-1}$	20640.000000	20640.7792
Effluent average S_I concentration	g COD.m^{-3}	28.060000	28.0643
Effluent average S_S concentration	g COD.m^{-3}	0.671900	0.67336
Effluent average X_I concentration	g COD.m^{-3}	5.919000	5.9191
Effluent average X_S concentration	g COD.m^{-3}	0.123300	0.12329
Effluent average X_{BH} concentration	g COD.m^{-3}	8.662000	8.6614
Effluent average X_{BA} concentration	g COD.m^{-3}	0.648900	0.6484
Effluent average X_P concentration	g COD.m^{-3}	3.748000	3.7485
Effluent average S_O concentration	g -COD.m^{-3}	1.373000	1.3748
Effluent average S_{NO} concentration	g N.m^{-3}	9.194000	9.1948
Effluent average S_{NH} concentration (limit = 4 g N.m^{-3})	g N.m^{-3}	0.158800	0.15845
Effluent average S_{ND} concentration	g N.m^{-3}	0.560300	0.55943
Effluent average X_{ND} concentration	g N.m^{-3}	0.009243	0.0092428
Effluent average S_{ALK} concentration	mole HCO_3^-.m^{-3}	4.564000	4.5646
Effluent average *TSS* concentration (limit = 30 g SS.m^{-3})	g.m^{-3}	14.330000	14.3255
Effluent average Temperature	°C	14.856600	14.8581
Effluent average Kjeldahl N concentration	g N.m^{-3}	2.053000	2.052
Effluent average total N concentration (limit = 18 g N.m^{-3})	g N.m^{-3}	11.250000	11.2468
Effluent average total COD concentration (limit = 100 g COD.m^{-3})	g COD.m^{-3}	47.840000	47.8383
Effluent average BOD_5 concentration (limit = 10 g.m^{-3})	g.m^{-3}	2.340000	2.3404

Effluent average load			
Variable	**Unit**	FORTRAN	MATLAB
Effluent average SI load	kg COD.d^{-1}	579.158400	579.269
Effluent average SS load	kg COD.d^{-1}	13.868016	13.8988
Effluent average XI load	kg COD.d^{-1}	122.168160	122.1748
Effluent average XS load	kg COD.d^{-1}	2.544912	2.5447
Effluent average XBH load	kg COD.d^{-1}	178.783680	178.7776
Effluent average XBA load	kg COD.d^{-1}	13.393296	13.3835
Effluent average XP load	kg COD.d^{-1}	77.358720	77.3722
Effluent average SO load	kg -COD.d^{-1}	28.338720	28.376
Effluent average SNO load	kg N.d^{-1}	189.764160	189.7888
Effluent average SNH load	kg N.d^{-1}	3.277632	3.2706
Effluent average SND load	kg N.d^{-1}	11.564592	11.547
Effluent average XND load	kg N.d^{-1}	0.190776	0.19078
Effluent average SALK load	kmole HCO_3^-.d^{-1}	94.200960	94.216
Effluent average TSS load	kg.d^{-1}	295.771200	295.6896
Effluent average Kjeldahl N load	kg N.d^{-1}	42.373920	42.3541
Effluent average total N load	kg N.d^{-1}	232.200000	232.1429
Effluent average total COD load	kg COD.d^{-1}	987.417600	987.4206
Effluent average BOD_5 load	kg.d^{-1}	48.297600	48.3079

Other output quality variables			
Variable	**Unit**	FORTRAN	MATLAB
Influent quality (*IQI*) index	kg poll.units.d^{-1}	74700.000000	74746.1235
Effluent quality (*EQI*) index	kg poll.units.d^{-1}	4840.000000	4843.9256
Average sludge production for disposal per day	kg SS.d^{-1}	2683.000000	2682.9648
Average sludge production released into effluent per day	kg SS.d^{-1}	296.000000	295.6896
Total average sludge production per day	kg SS.d^{-1}	2979.000000	2978.6544

'Energy' related variables			
Variable	**Unit**	FORTRAN	MATLAB
Average aeration energy	kWh.d^{-1}	4000.000000	4000
Average pumping energy	kWh.d^{-1}	440.900000	441.5576
Average carbon source dosage	kg COD.d^{-1}	800.000000	800
Average mixing energy	kWh.d^{-1}	768.000000	768
Average heating energy	kWh.d^{-1}	4180.000000	4179.8063
Average methane gas production (1 kg = 13.8928 kWh)	kg CH_4.d^{-1}	1065.000000	1065.3522
Average hydrogen gas production	kg H_2.d^{-1}		0.0035541
Average carbon dioxide gas production	kg CO_2.d^{-1}		1535.4117
Average total gas flow rate from AD (normalized to P_{atm})	'normal' m^3.d^{-1}	2562.580000	2708.3428

Operational cost index			
Variable (including weight factor)	**Unit**	FORTRAN	MATLAB
Sludge production cost index	–	8049.000000	8048.8944
Aeration energy cost index	–	4000.000000	4000
Pumping energy cost index	–	440.900000	441.5576
Carbon source dosage cost index	–	2400.000000	2400
Mixing energy cost index	–	768.000000	768
Heating energy cost index	–	0.000000	0
Net energy production from methane index (subtracted from rest)	–		6392.1131
Total Operational Cost Index (*OCI*)	–	9267.000000	9266.3389

APPENDIX 4: Open-loop results

These results were obtained with FORTRAN (integration using a Runge-Kutta 4 algorithm with a constant integration step = 0.005 h) and MATLAB-Simulink (Solver: ode45, absolute tolerance =10^{-8}, relative tolerance =10^{-5}). For details on operation conditions see text.

Effluent average concentrations based on load, including bypass			
Variable	**Unit**	FORTRAN	MATLAB
Effluent average flow rate	$m^3.d^{-1}$	20661.600000	20661.2186
Effluent average S_I concentration	g COD.m^{-3}	28.050000	28.0493
Effluent average S_S concentration	g COD.m^{-3}	0.809400	0.81123
Effluent average X_I concentration	g COD.m^{-3}	6.489000	6.4896
Effluent average X_S concentration	g COD.m^{-3}	0.348600	0.34855
Effluent average X_{BH} concentration	g COD.m^{-3}	9.692000	9.6917
Effluent average X_{BA} concentration	g COD.m^{-3}	0.675100	0.67445
Effluent average X_P concentration	g COD.m^{-3}	3.991000	3.9911
Effluent average S_O concentration	g -COD.m^{-3}	1.031000	1.0319
Effluent average S_{NO} concentration	g N.m^{-3}	7.471000	7.4746
Effluent average S_{NH} concentration (limit = 4 g N.m^{-3})	g N.m^{-3}	1.651000	1.6491
Effluent average S_{ND} concentration	g N.m^{-3}	0.602800	0.60193
Effluent average X_{ND} concentration	g N.m^{-3}	0.020180	0.02018
Effluent average S_{ALK} concentration	mole HCO_3^-.m^{-3}	4.796000	4.796
Effluent average *TSS* concentration (limit = 30 g SS.m^{-3})	g.m^{-3}	15.900000	15.8965
Effluent average Temperature	°C		14.8604
Effluent average Kjeldahl N concentration	g N.m^{-3}	3.732000	3.7294
Effluent average total N concentration (limit = 18 g N.m^{-3})	g N.m^{-3}	11.200000	11.204
Effluent average total COD concentration (limit = 100 g COD.m^{-3})	g COD.m^{-3}	50.050000	50.0559
Effluent average BOD_5 concentration (limit = 10 g.m^{-3})	g.m^{-3}	2.674000	2.6742

Effluent average load, including bypass			
Variable	**Unit**	FORTRAN	MATLAB
Effluent average S_I load	kg COD.d^{-1}	579.557880	579.5322
Effluent average S_S load	kg COD.d^{-1}	16.723499	16.761
Effluent average X_I load	kg COD.d^{-1}	134.073122	134.0829
Effluent average X_S load	kg COD.d^{-1}	7.202634	7.2014
Effluent average X_{BH} load	kg COD.d^{-1}	200.252227	200.2421
Effluent average X_{BA} load	kg COD.d^{-1}	13.948646	13.9349
Effluent average X_P load	kg COD.d^{-1}	82.460446	82.4607
Effluent average S_O load	kg -COD.d^{-1}	21.302110	21.3202
Effluent average S_{NO} load	kg N.d^{-1}	154.362814	154.435
Effluent average S_{NH} load	kg N.d^{-1}	34.112302	34.0731
Effluent average S_{ND} load	kg N.d^{-1}	12.454812	12.4366
Effluent average X_{ND} load	kg N.d^{-1}	0.416951	0.41694
Effluent average S_{ALK} load	kmole HCO_3^-.d^{-1}	99.093034	99.0915
Effluent average *TSS* load	kg.d^{-1}	328.519440	328.4415
Effluent average Kjeldahl N load	kg N.d^{-1}	77.109091	77.0534
Effluent average total N load	kg N.d^{-1}	231.409920	231.4884
Effluent average total COD load	kg COD.d^{-1}	1034.113080	1034.2151
Effluent average BOD_5 load	kg.d^{-1}	55.249118	55.2513

Other output quality variables			
Variable	**Unit**	FORTRAN	MATLAB
Influent quality (*IQI*) index	kg poll.units.d^{-1}	74700.000000	74783.3138
Effluent quality (*EQI*) index	kg poll.units.d^{-1}	5660.000000	5657.5529
Average sludge production for disposal per day	kg SS.d^{-1}	2646.000000	2651.8714
Average sludge production released into effluent per day	kg SS.d^{-1}	325.000000	328.4415
Total average sludge production per day	kg SS.d^{-1}	2971.000000	2980.3128

'Energy' related variables			
Variable	**Unit**	FORTRAN	MATLAB
Average aeration energy	kWh.d^{-1}	4000.000000	4000
Average pumping energy	kWh.d^{-1}	440.900000	441.5362
Average carbon source dosage	kg COD.d^{-1}	800.000000	800
Average mixing energy	kWh.d^{-1}	768.000000	768
Average heating energy	kWh.d^{-1}	4177.000000	4177.3089
Average methane gas production (1 kg = 13.8928 kWh)	kg CH_4.d^{-1}	1057.000000	1059.4972
Average hydrogen gas production	kg H_2.d^{-1}	0.003576	0.0036065
Average carbon dioxide gas production	kg CO_2.d^{-1}	1485.000000	1527.5286
Average total gas flow rate from AD (normalized to P_{atm})	'normal' m^3.d^{-1}	2547.000000	2693.6501

Operational cost index			
Variable (including weight factor)	**Unit**	FORTRAN	MATLAB
Sludge production cost index	–	7938.000000	7955.6141
Aeration energy cost index	–	4000.000000	4000
Pumping energy cost index	–	440.900000	441.5362
Carbon source dosage cost index	–	2400.000000	2400
Mixing energy cost index	–	768.000000	768
Heating energy cost index	–	0.000000	0
Net energy production from methane index (subtracted from rest)	–		6356.9833
Total Operational Cost Index (*OCI*)	–	9208.000000	9208.1669

Effluent violations			
Variable	**Unit**	FORTRAN	MATLAB
95% percentile of effluent SNH (Ammonia95)	g $N.m^{-3}$	4.650000	4.6523
95% percentile of effluent total N (TN95)	g $N.m^{-3}$	15.100000	15.1312
95% percentile of effluent TSS (TSS95)	g $COD.m^{-3}$	20.100000	20.1336
Maximum effluent total N limit (18 g $N.m^{-3}$) was violated			
during:	days	0.374920	0.39583
% of total evaluation time:	%	0.103000	0.10875
number of violations:	–	4.000000	5
Maximum effluent total COD limit (100 g $COD.m^{-3}$) was violated			
during:	days	0.218764	0.21875
% of total evaluation time:	%	0.060100	0.060096
number of violations:	–	3.000000	3
Maximum effluent ammonia limit (4 g $N.m^{-3}$) was violated			

during:	days	30.175600	30.1042
% of total evaluation time:	%	8.290000	8.2704
number of violations:	–	172.000000	171
Maximum effluent TSS limit (30 g SS.m^{-3}) was violated			
during:	days	1.426880	1.4375
% of total evaluation time:	%	0.392000	0.39492
number of violations:	–	12.000000	12
Maximum effluent BOD_5 limit (10 g.m^{-3}) was violated			
during:	days	0.436800	0.4375
% of total evaluation time:	%	0.120000	0.12019
number of violations:	–	5.000000	5

APPENDIX 5: Closed-loop results with ideal sensors and actuators

These results were obtained with FORTRAN (integration using a Runge-Kutta 4 algorithm with a constant integration step = 0.005 h) and MATLAB-Simulink (Solver: ode45, absolute tolerance =10^{-8}, relative tolerance =10^{-5}). For details on operation conditions see text. Details on PI controller settings are given in table headings.

Effluent average concentrations based on load, including bypass			
Variable	**Unit**	FORTRAN	MATLAB
Effluent average flow rate	$m^3.d^{-1}$	20661.600000	20661.0229
Effluent average S_I concentration	g COD.m^{-3}	28.060000	28.0618
Effluent average S_S concentration	g COD.m^{-3}	0.724600	0.72625
Effluent average X_I concentration	g COD.m^{-3}	5.913000	5.9139
Effluent average X_S concentration	g COD.m^{-3}	0.331300	0.33129
Effluent average X_{BH} concentration	g COD.m^{-3}	9.892000	9.892
Effluent average X_{BA} concentration	g COD.m^{-3}	0.691500	0.69097
Effluent average X_P concentration	g COD.m^{-3}	3.400000	3.4003
Effluent average S_O concentration	g -COD.m^{-3}	1.577000	1.579
Effluent average S_{NO} concentration	g N.m^{-3}	11.060000	11.0536
Effluent average S_{NH} concentration (limit = 4 g N.m^{-3})	g N.m^{-3}	0.473600	0.47377
Effluent average S_{ND} concentration	g N.m^{-3}	0.585900	0.58498
Effluent average X_{ND} concentration	g N.m^{-3}	0.019080	0.019077
Effluent average S_{ALK} concentration	mole HCO_3^-.m^{-3}	4.455000	4.4553
Effluent average *TSS* concentration (limit = 30 g SS.m^{-3})	g.m^{-3}	15.170000	15.1713
Effluent average Temperature	°C	14.860000	14.8603
Effluent average Kjeldahl N concentration	g N.m^{-3}	2.484000	2.4833
Effluent average total N concentration (limit = 18 g N.m^{-3})	g N.m^{-3}	13.540000	13.5369
Effluent average total COD concentration (limit = 100 g COD.m^{-3})	g COD.m^{-3}	49.020000	49.0165
Effluent average BOD_5 concentration (limit = 10 g.m^{-3})	g.m^{-3}	2.698000	2.6985

Effluent average load, including bypass			
Variable	**Unit**	FORTRAN	MATLAB
Effluent average S_I load	kg COD.d^{-1}	579.764496	579.7858
Effluent average S_S load	kg COD.d^{-1}	14.971395	15.005
Effluent average X_I load	kg COD.d^{-1}	122.172041	122.1863
Effluent average X_S load	kg COD.d^{-1}	6.845188	6.8449
Effluent average X_{BH} load	kg COD.d^{-1}	204.384547	204.3795
Effluent average X_{BA} load	kg COD.d^{-1}	14.287496	14.2761
Effluent average X_P load	kg COD.d^{-1}	70.249440	70.2529
Effluent average S_O load	kg -COD.d^{-1}	32.583343	32.6228
Effluent average S_{NO} load	kg N.d^{-1}	228.517296	228.3783
Effluent average S_{NH} load	kg N.d^{-1}	9.785334	9.7886
Effluent average S_{ND} load	kg N.d^{-1}	12.105631	12.0864
Effluent average X_{ND} load	kg N.d^{-1}	0.394223	0.39415
Effluent average S_{ALK} load	kmol HCO_3^-.d^{-1}	92.047428	92.0513
Effluent average *TSS* load	kg.d^{-1}	313.436472	313.4547
Effluent average Kjeldahl N load	kg N.d^{-1}	51.323414	51.3079
Effluent average total N load	kg N.d^{-1}	279.758064	279.6863
Effluent average total COD load	kg COD.d^{-1}	1012.831632	1012.7305
Effluent average BOD_5 load	kg.d^{-1}	55.744997	55.7533

Other output quality variables			
Variable	**Unit**	FORTRAN	MATLAB
Influent quality (*IQI*) index	kg poll.units.d^{-1}	74700.000000	74783.3138
Effluent quality (*EQI*) index	kg poll.units.d^{-1}	5580.000000	5574.1681
Average sludge production for disposal per day	kg SS.d^{-1}	2702.000000	2707.7709
Average sludge production released into effluent per day	kg SS.d^{-1}	309.000000	313.4547
Total average sludge production per day	kg SS.d^{-1}	3011.000000	3021.2257

'Energy' related variables			
Variable	**Unit**	FORTRAN	MATLAB
Average aeration energy	kWh.d^{-1}	4223.000000	4222.6368
Average pumping energy	kWh.d^{-1}	444.800000	445.4525
Average carbon source dosage	kg COD.d^{-1}	800	800
Average mixing energy	kWh.d^{-1}	768.000000	768
Average heating energy	kWh.d^{-1}	4226.000000	4225.3462
Average methane gas production (1 kg = 13.8928 kWh)	kg CH_4.d^{-1}	1083.000000	1085.3603
Average hydrogen gas production	kg H_2.d^{-1}	0.003681	0.0037088
Average carbon dioxide gas production	kg CO_2.d^{-1}	1521.000000	1562.6873
Average total gas flow rate from AD (normalized to P_{atm})	'normal' m^3.d^{-1}	2605.000000	2757.956

Operational cost index			
Variable (including weight factor)	**Unit**	FORTRAN	MATLAB
Sludge production cost index	–	8106.000000	8123.3128
Aeration energy cost index	–	4223.000000	4222.6368
Pumping energy cost index	–	444.800000	445.4525
Carbon source dosage cost index	–	2400.000000	2400
Mixing energy cost index	–	768.000000	768
Heating energy cost index	–	0.000000	0
Net energy production from methane index (subtracted from rest)	–		6512.1616
Total Operational Cost Index (*OCI*)	–	9443.000000	9447.2406

Effluent violations			
Variable	**Unit**	FORTRAN	MATLAB
95% percentile of effluent S_{NH} (Ammonia95)	g N.m^{-3}	1.530000	1.5451
95% percentile of effluent total N (TN95)	g N.m^{-3}	16.800000	16.756
95% percentile of effluent *TSS* (TSS95)	g COD.m^{-3}	19.700000	19.7367
Maximum effluent total N limit (18 g N.m^{-3}) was violated			
during:	days	4.258800	4.3021
% of total evaluation time:	%	1.170000	1.1819
number of violations:	–	32.000000	33
Maximum effluent total COD limit (100 g COD.m^{-3}) was violated			
during:	days	0.208208	0.20833
% of total evaluation time:	%	0.057200	0.057234
number of violations:	–	3.000000	3
Maximum effluent ammonia limit (4 g N.m^{-3}) was violated			
during:	days	1.488760	1.4896
% of total evaluation time:	%	0.409000	0.40923
number of violations:	–	11.000000	11
Maximum effluent TSS limit (30 g SS.m^{-3}) was violated			
during:	days	1.259440	1.25
% of total evaluation time:	%	0.346000	0.34341
number of violations:	–	11.000000	11
Maximum effluent BOD_5 limit (10 g.m^{-3}) was violated			
during:	days	0.458640	0.45833
% of total evaluation time:	%	0.126000	0.12592
number of violations:	–	6.000000	6

Controller performance

DO controller		**Unit**	FORTRAN	MATLAB
Controller type			discrete PI, bias=120 1/d, K=21.6 m^3.g (–COD)$^{-1}$.d^{-1}, T_i=0.00208 d,T_e=8.3e-4 d	continuous PI with antiwindup, K=25 m^3.g (–COD)$^{-1}$.d^{-1}, T_i=0.002 d, T_t=0.001 d
Setpoint SO4		g (-COD).m^{-3}	2.000000	2
Average of e_{SO4}	mean(e)	g (-COD).m^{-3}	-0.000004	-1.15e-6
Average of $\|e_{SO4}\|$	mean (abs(e))	g (-COD).m^{-3}		0.024942
IAE e_{SO4}	integral of absolute error	g (-COD).m^{-3}.d	10.953500	9.079
ISE e_{SO4}	integral of square error	(g (-COD)/m^3)^2.d	0.578263	0.39804
Max e_{SO4}	max deviation from setpoint	g (-COD).m^{-3}	0.175859	0.14694
Standard deviation of e_{SO4}	std(e)	g (-COD).m^{-3}	0.039860	0.033068
Variance of e_{SO4}	var(e)	(g (-COD).m^{-3})^2	0.001589	0.0010935
Max deviation of K_La_4	max(K_La_4)-min(K_La_4)	d^{-1}	240.000000	191.1517
Max deviation of K_La_4 in 1 sample	max(delta K_La_4)	d^{-1}	1.526198	18.9355
Average value of K_La_4	mean(K_La_4)	d^{-1}	63.614082	126.6785
Standard deviation of K_La_4	std(delta K_La_4)	d^{-1}	0.226210	2.8233
Variance of K_La_4	var(delta K_La_4)	(d^{-1})^2	0.051171	7.9711

APPENDIX 5: Closed-loop results with realistic sensors and actuators

These results were obtained with MATLAB-Simulink (Solver: ode45, absolute tolerance $=10^{-8}$, relative tolerance $=10^{-5}$). For details on operation conditions see text. Details on PI controller settings are given in table headings.

Effluent average concentrations based on load, including bypass		
Variable	**Unit**	MATLAB
Effluent average flow rate	$m^3.d^{-1}$	20661.0241
Effluent average flow rate	g COD.m^{-3}	28.0618
Effluent average S_I concentration	g COD.m^{-3}	0.72634
Effluent average S_S concentration	g COD.m^{-3}	5.9139
Effluent average X_I concentration	g COD.m^{-3}	0.33132
Effluent average X_S concentration	g COD.m^{-3}	9.892
Effluent average X_{BH} concentration	g COD.m^{-3}	0.69096
Effluent average X_{BA} concentration	g COD.m^{-3}	3.4003
Effluent average X_P concentration	g -COD.m^{-3}	1.5798
Effluent average S_O concentration	g N.m^{-3}	11.0455
Effluent average S_{NO} concentration	g N.m^{-3}	0.47429
Effluent average S_{NH} concentration (limit = 4 g N.m^{-3})	g N.m^{-3}	0.58502
Effluent average S_{ND} concentration	g N.m^{-3}	0.019079
Effluent average X_{ND} concentration	mole HCO_3^-.m^{-3}	4.4559
Effluent average S_{ALK} concentration	g.m^{-3}	15.1713
Effluent average Temperature	°C	14.8603
Effluent average Kjeldahl N concentration	g N.m^{-3}	2.4839
Effluent average total N concentration (limit = 18 g N.m^{-3})	g N.m^{-3}	13.5294
Effluent average total COD concentration (limit = 100 g COD.m^{-3})	g COD.m^{-3}	49.0166
Effluent average BOD_5 concentration (limit = 10 g.m^{-3})	g.m^{-3}	2.6985

Effluent average load, including bypass		
Variable	**Unit**	MATLAB
Effluent average S_I load	kg COD.d^{-1}	579.7859
Effluent average S_S load	kg COD.d^{-1}	15.0069
Effluent average X_I load	kg COD.d^{-1}	122.1863
Effluent average X_S load	kg COD.d^{-1}	6.8453
Effluent average X_{BH} load	kg COD.d^{-1}	204.3794
Effluent average X_{BA} load	kg COD.d^{-1}	14.2759
Effluent average X_P load	kg COD.d^{-1}	70.2528
Effluent average S_O load	kg -COD.d^{-1}	32.6408
Effluent average S_{NO} load	kg N.d^{-1}	228.2123
Effluent average S_{NH} load	kg N.d^{-1}	9.7993
Effluent average S_{ND} load	kg N.d^{-1}	12.0872
Effluent average X_{ND} load	kg N.d^{-1}	0.39418
Effluent average S_{ALK} load	kmol HCO_3^-.d^{-1}	92.0639
Effluent average *TSS* load	kg.d^{-1}	313.4547
Effluent average Kjeldahl N load	kg N.d^{-1}	51.3195
Effluent average total N load	kg N.d^{-1}	279.5318
Effluent average total COD load	kg COD.d^{-1}	1012.7325
Effluent average BOD_5 load	kg.d^{-1}	55.7538

Other output quality variables		
Variable	**Unit**	MATLAB
Influent quality (*IQI*) index	kg poll.units.d^{-1}	74783.3138
Effluent quality (*EQI*) index	kg poll.units.d^{-1}	5572.8572
Average sludge production for disposal per day	kg SS.d^{-1}	2707.769
Average sludge production released into effluent per day	kg SS.d^{-1}	313.4547
Total average sludge production per day	kg SS.d^{-1}	3021.2237

'Energy' related variables		
Variable	**Unit**	MATLAB
Average aeration energy	kWh.d^{-1}	4225.4326
Average pumping energy	kWh.d^{-1}	445.4525
Average carbon source dosage	kg COD.d^{-1}	800
Average mixing energy	kWh.d^{-1}	768
Average heating energy	kWh.d^{-1}	4225.3434
Average methane gas production (1 kg = 13.8928 kWh)	kg CH_4.d^{-1}	1085.3599
Average hydrogen gas production	kg H_2.d^{-1}	0.0037088
Average carbon dioxide gas production	kg CO_2.d^{-1}	1562.6859
Average total gas flow rate from AD (normalized to P_{atm})	'normal' m^3.d^{-1}	2757.9546

Operational cost index		
Variable (including weight factor)	**Unit**	MATLAB
Sludge production cost index	–	8123.3069
Aeration energy cost index	–	4225.4326
Pumping energy cost index	–	445.4525
Carbon source dosage cost index	–	2400
Mixing energy cost index	–	768
Heating energy cost index	–	0
Net energy production from methane index (subtracted from rest)	–	6512.1596
Total Operational Cost Index (*OCI*)	–	9450.0324

Effluent violations		
Variable	**Unit**	MATLAB
95% percentile of effluent S_{NH} (Ammonia95)	g N.m^{-3}	1.5427
95% percentile of effluent total N (TN95)	g N.m^{-3}	16.7525
95% percentile of effluent *TSS* (TSS95)	g COD.m^{-3}	19.7367
Maximum effluent total N limit (18 g N.m^{-3}) was violated		
during:	days	4.2812
% of total evaluation time:	%	1.1762
number of violations:	–	32
Maximum effluent total COD limit (100 g COD.m^{-3}) was violated		
during:	days	0.20833
% of total evaluation time:	%	0.057234
number of violations:	–	3
Maximum effluent ammonia limit (4 g N.m^{-3}) was violated		
during:	days	1.4896
% of total evaluation time:	%	0.40923
number of violations:	–	11
Maximum effluent TSS limit (30 g SS.m^{-3}) was violated		
during:	days	1.25
% of total evaluation time:	%	0.34341
number of violations:	–	11
Maximum effluent BOD_5 limit (10 g.m^{-3}) was violated		
during:	days	0.45833
% of total evaluation time:	%	0.12592
number of violations:	–	6

Controller performance			
Qw controller		**Unit**	MATLAB
Controller type			timer based Q_w actuator as first order filter used
Setpoint Qw		m^3.d^{-1}	
Average of e_{Qw}	mean(e)	m^3.d^{-1}	-8.42.10^{-14}
Average of $\|e_{Qw}\|$	mean (abs(e))	m^3.d^{-1}	0.0077684
IAE e_{Qw}	integral of absolute error	m^3	2.8277
ISE e_{Qw}	integral of square error	(m^3)^2.d	383.78
Max e_{Qw}	max deviation from setpoint	m^3.d^{-1}	135.7256
Standard deviation of e_{Qw}	std(e)	m^3.d^{-1}	1.0268
Variance of e_{Qw}	var(e)	(m^3.d^{-1})^2	1.0543
Max deviation of Q_w	max(Q_w)-min(Q_w)	m^3.d^{-1}	150
Max deviation of Q_w in 1 sample	max(deltaQ_w)	m^3.d^{-1}	150
Average value of Q_w	mean(Q_w)	m^3.d^{-1}	375.0021
Standard deviation of deltaQ_w	std(deltaQ_w)	m^3.d^{-1}	1.1348
Variance of deltaQ_w	var(deltaQ_w)	(m^3.d^{-1})^2	1.2877

DO controller		**Unit**	MATLAB
Controller type			continuous PI with antiwindup, K=25 $m^3.g\ (-COD)^{-1}.d^{-1}$, T_i=0.002 d, T_t=0.001 d
Setpoint SO4		g (-COD).m^{-3}	2
Average of e_{SO4}	mean(e)	g (-COD).m^{-3}	-0.00041156
Average of $\|e_{SO4}\|$	mean (abs(e))	g (-COD).m^{-3}	0.087979
IAE e_{SO4}	integral of absolute error	g (-COD).m^{-3}.d	32.0243
ISE e_{SO4}	integral of square error	(g (-COD)/m^3)^2.d	4.5167
Max e_{SO4}	max deviation from setpoint	g (-COD).m^{-3}	0.6321
Standard deviation of e_{SO4}	std(e)	g (-COD).m^{-3}	0.11139
Variance of e_{SO4}	var(e)	(g (-COD).m^{-3})^2	0.012408
Max deviation of K_La_4	max(K_La_4)-min(K_La_4)	d^{-1}	227.6513
Max deviation of K_La_4 in 1 sample	max(delta K_La_4)	d^{-1}	54.9573
Average value of K_La_4	mean K_La_4)	d^{-1}	126.7875
Standard deviation of K_La_4	std(delta K_La_4)	d^{-1}	8.1897
Variance of K_La_4	var(delta K_La_4)	$(d^{-1})^2$	67.0709