Implementing ADM1 for plant-wide benchmark simulations in Matlab/Simulink

C. Rosen*, D. Vrecko*,, K.V. Gernaey*,***, M.N. Pons**** and U. Jeppsson***

*Department of Industrial Electrical Engineering and Automation, Lund University, Box 118, SE-22100 Lund, Sweden (E-mail: *christian.rosen@iea.lth.se*; *ulf.jeppsson@iea.lth.se*)

**Jozef Stefan Institute, Jamova 39, SI-1000 Ljubljana, Slovenia (E-mail: *darko.vrecko@ijs.si*)

***Department of Chemical Engineering, Technical University of Denmark, DK-2800 Lyngby, Denmark (E-mail: *kvg@kt.dtu.dk*)

****Laboratoire des Sciences du Génie Chimique, CNRS-ENSIC-INPL, 1, rue Grandville, F-54001 Nancy cedex, France (E-mail: *Marie-Noëlle.Pons@ensic.inpl-nancy.fr*)

Abstract The IWA Anaerobic Digestion Model No.1 (ADM1) was presented in 2002 and is expected to represent the state-of-the-art model within this field in the future. Due to its complexity the implementation of the model is not a simple task and several computational aspects need to be considered, in particular if the ADM1 is to be included in dynamic simulations of plant-wide or even integrated systems. In this paper, the experiences gained from a Matlab/Simulink implementation of ADM1 into the extended COST/IWA Benchmark Simulation Model (BSM2) are presented. Aspects related to system stiffness, model interfacing with the ASM family, mass balances, acid-base equilibrium and algebraic solvers for pH and other troublesome state variables, numerical solvers and simulation time are discussed. The main conclusion is that if implemented properly, the ADM1 will also produce high-quality results in dynamic plant-wide simulations including noise, discrete sub-systems, etc. without imposing any major restrictions due to extensive computational efforts.
Keywords ADM1; anaerobic digestion; benchmark; BSM2; modelling; simulation

Water Science & Technology **Vol 54 No 4 pp 11–19** © IWA Publishing 2006

Introduction

A wastewater treatment plant should be considered as a unit, where all sub-processes are linked together and are operated and controlled not only on a local level as individual processes but by supervisory systems taking into account all the interactions between the processes. Otherwise, sub-optimisation will be an unavoidable outcome leading to reduced effluent quality and/or higher operational costs. The development of plant-wide modelling in the wastewater treatment field is attractive to many researchers as it provides a holistic view of the process and it allows for a more comprehensive understanding of the interactions between the various unit processes. Further, the impact of dynamic changes in the process can be explored as these changes relate to all unit processes that may be present in a treatment layout. Plant-wide modelling is an important tool for development and testing of new control and monitoring schemes for wastewater treatment. Substantial efforts have been directed towards the development of standardized *simulation benchmarks*, which allow for unbiased evaluation of different control strategies under realistic conditions. The development of the Benchmark Simulation Model no 1 (BSM1) (Copp, 2002) has proven successful and is today widely used in the research community as a benchmark system for the activated sludge process. As a continuation of that work, a plant-wide simulation benchmark (BSM2) is being developed that includes the most common unit processes in a treatment plant (Jeppsson *et al.*, 2006). The BSM2 definition consists of the model, an associated control system, a benchmarking procedure and evaluation criteria. The model includes primary clarifier, a five-tank activated sludge

doi: 10.2166/wst.2006.521

system with a (non-reactive) secondary clarifier, a sludge thickener, an anaerobic digester and a dewatering unit (Figure 1). In contrast to BSM1, whose influent characteristics are provided as a data file, a model including urban drainage and sewer system effects (Gernaey *et al.*, 2005) generates the influent wastewater characteristics.

The availability of faster, more powerful computers allows for more complex models to be simulated but it is clear that a plant-wide model, which includes many unit processes (e.g. BSM2), requires significant computational power. For steady-state simulation (constant inputs), the computational burden is relatively acceptable but for dynamic simulations (changing inputs), simulation time becomes extensive when the evaluation period is long (for BSM2 20 months of dynamic simulations are required for each test). Moreover, dynamic and especially stochastic inputs in combination with the inherent structure of the BSM2 further complicate the simulations due to numerical considerations and restrictions. In this paper, we discuss the implementation of the Anaerobic Digestion Model no 1 (ADM1) (Batstone *et al.*, 2002) in Matlab/Simulink as an integrated part of the BSM2. This includes computational aspects encountered in the implementation of the BSM2 and some solutions in order to improve the simulation speed. The paper also provides additional comments regarding mass balances for nitrogen and carbon as well as other experiences gained. The effort to improve the simulation speed of the ADM1 is due to the need to reduce the simulation time for the BSM2. However, the results presented in this paper are general and other users of the ADM1 should also benefit from these results.

ADM1 for BSM2

The fact that the Matlab/Simulink implementation discussed in this work aims at an integrated part of the BSM2 puts some requirements on the way the ADM1 is implemented. 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.

Dynamic inputs

BSM2 aims at developing and evaluating control strategies for wastewater treatment. The challenge to control a WWTP lies mainly in the changing influent wastewater

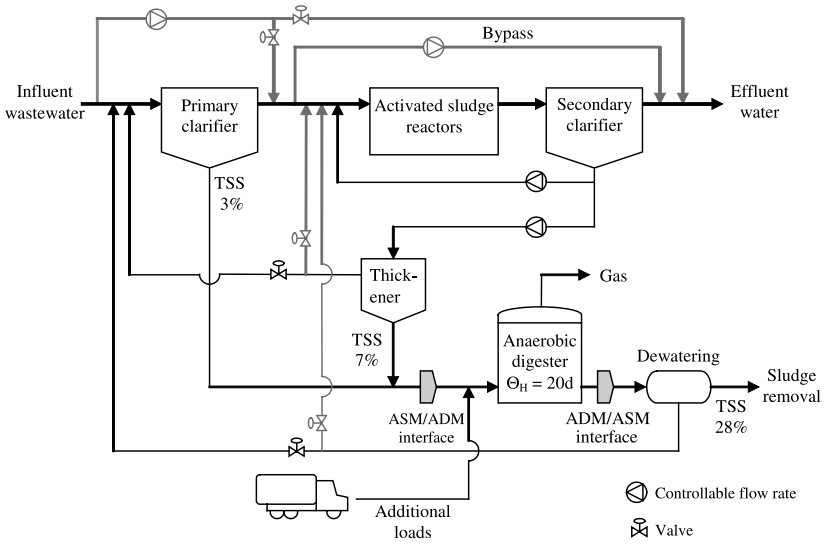

Figure 1 Plant layout for the BSM2

characteristics. The generation of dynamic input is, thus, an integrated part of the BSM2. This means that, except in order to obtain initial conditions, BSM2 is always simulated using dynamic input and, consequently, no plant unit is ever at steady state. According to the BSM2 protocol, using dynamic influent data, the plant is simulated for 63 days to reach a pseudo-steady state. This is followed by 182 days of simulation for initialisation of control and/or monitoring algorithms. The subsequent 364 days of simulation is the actual evaluation period. In total, this encompasses 609 days of dynamic simulations (Gernaey *et al.*, 2005) with new data every 15 minutes.

Control actions

The BSM2 is a control benchmark. It should be possible to test and evaluate various types of controllers (Jeppsson *et al.*, 2006). From a numerical point of view, continuous controllers yield the least computational effort. However, discrete controllers are more common and many of the commercially available sensors are also discrete. Moreover, some control actions can be event driven when applying rule-based control (e.g. if-then-else rules). Introducing discrete controllers and sensors as well as event-driven control in the model results in a so-called hybrid system.

Noise sources

To obtain realistic and useful evaluation results from an investigation of a control strategy, the strategy must be subjected to various types of errors and disturbances encountered in real operation. One of the most important sources of errors is sensor noise and delays. Realistic sensor model behaviour requires the dynamic properties and disturbance sources to be represented. This typically includes modelling of noise and time response and, if not a continuous sensor, the sampling and measuring interval (Rieger *et al.*, 2003). Noise generation must be done individually for each sensor in the system so that it is truly independent.

The stiffness problem

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 react sluggishly. The ADM1 is a very stiff system with time constants ranging from fractions of a second to months. This makes the simulation of such a system challenging and to avoid excessively long simulation times, one needs to be somewhat creative in implementing the model.

Simulating stiff systems in Matlab/Simulink

Some of the solvers in Matlab/Simulink are so-called stiff solvers and, consequently, capable of solving stiff systems. However, a problem common to all stiff solvers is the difficulty in handling dynamic input and especially noise and step changes. The more stochastic or random an input variable behaves, the more problematic is the simulation using a stiff solver. 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 a stiff system it is sometimes possible to reformulate some of the system equations in order to omit the fastest states. The rationale for this is that from the slower states' 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 ordinary differential equations (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. 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.

Implementing ADM1 in Matlab/Simulink

This ADM1 implementation in Matlab/Simulink deviates somewhat from the model description in Batstone *et al.* (2002). There are mainly two reasons for this. Firstly, the ADM1 is implemented so that it is consistent with the BSM2. Secondly, the computational requirements must be taken into account.

ADM1/ASM1 state variable transformations

As the state variables representing the ASM1 and ADM1 models are quite different, model interfaces are necessary when combining the two processes (as in BSM2). A rudimentary interface is proposed in Batstone *et al.* (2002). However, the benchmark developers have created a more elaborate yet simple transformer. The interfaces between ASM1 and ADM1 are enhanced versions of the interfaces described in Copp *et al.* (2003). The ASM1 to ADM1 transformation initially removes any oxygen and nitrate in the wastewater with an associated COD reduction and then divides the remaining COD and nitrogen components into relevant ADM1 state variables (e.g. proteins, lipids, carbohydrates, inerts). Obviously, primary and secondary sludge characteristics differ. However, the enhanced interface handles both types using the same parameter set and will lead to the expected differences in terms of gas production etc. in the digester. No inputs entering the digester are defined as composite material and therefore the disintegration process of ADM1 will only influence the behaviour of the model in terms of internal disintegration of products arising from biomass decay. This approach allows the modeller to clearly separate the disintegration of raw sludge from internal digester sludge. The ADM1 to ASM1 interface amalgamates the large number of ADM1 state variables back into ASM1 state variables. At all times, COD, nitrogen and charge balances are maintained. Work is also on-going to develop a more general protocol for interfacing models based on Petersen matrices (Vanrolleghem *et al.*, 2005). In Zaher *et al.* (2006), the results of this protocol are evaluated and compared to the Copp *et al.* interface for the specific case of ASM1/ADM1 transformations.

Mass balances

To maintain complete mass balances for all model components (COD, N, etc.) is a fundamental issue of any model. Based on the description in Batstone *et al.* (2002), a few comments are required to avoid problems when implementing the model. The original ADM1 includes a process referred to as disintegration, where a composite material (X_C) is transformed into various components (S_I, X_{ch}, X_{pr}, X_{li} and X_I). Assuming one COD mass unit of X_C completely disintegrating will produce:

$$f_{sI,xc}S_I + f_{xI,xc}X_I + f_{ch,xc}X_{ch} + f_{pr,xc}X_{pr} + f_{li,xc}X_{li} = 0.1S_I + 0.25X_I + 0.2X_{ch} + 0.2X_{pr} + 0.25X_{li}$$

A COD balance exists as long as the sum of all $f_{i,xc} = 1$. However, the proposed nitrogen content of X_C (N_{xc}) is 0.002 kmole N/kg COD. If we instead calculate the nitrogen content of the disintegration products (kmole N) using parameter values from Batstone *et al.* (2002), we obtain:

$$N_I \cdot 0.1S_I + N_I \cdot 0.25X_I + N_{ch} \cdot 0.2X_{ch} + N_{aa} \cdot 0.2X_{pr} + N_{li} \cdot 0.25X_{li} = 0.0002 + 0.0005 + 0.0014 = 0.0021$$

($N_I = 0.002$, $N_{ch} = 0$, $N_{li} = 0$ and $N_{aa} = 0.007$ kmole N/kg COD). This means that for every kg of COD that disintegrates 0.1 mole of N is created (a similar problem exist for the carbon mass balance and we recommend that the carbon content of X_C (C_{xc}) is changed accordingly). Obviously, the nitrogen contents and yields from composites are highly variable and may need adjustment for every specific case study but the "default" parameter values should close the mass balances. In this paper, we suggest new values for $f_{xI,xc} = 0.2$ and $f_{li,xc} = 0.3$. For the specific benchmark implementation we have also modified N_I for particulate inerts to $0.06/14 \approx 0.00429$ kmole N/kg COD to be consistent with the ASM1 model. For the same reason N_I for soluble inerts is set to zero. Because of the latter modifications N_{xc} is adjusted to $0.0316/14 \approx 0.00226$ kmole N/kg COD to maintain the nitrogen balance.

The state variables inorganic carbon and inorganic nitrogen may act as source or sink terms to close mass balances. However, the provided stoichiometric matrix is not defined to take this into account. Let us take an example: decay of biomass (processes no 13–19) produces an equal amount of composites on a COD basis. However, the carbon content may certainly vary from biomass to composites resulting from decay. It is suggested in Batstone *et al.* (2002) that the nitrogen content of bacteria (N_{bac}) is 0.00625 kmole N/kg COD, which is three times higher than the suggested value for N_{xc}. In such a case, it is logical to add a stoichiometric term ($N_{bac}-N_{xc}$) into the Petersen matrix, which will keep track of the fate of the excess nitrogen. The same principle holds for carbon during biomass decay, i.e. ($C_{bac}-C_{xc}$). A strong recommendation is to include stoichiometric relationships for all 19 processes regarding inorganic carbon and inorganic nitrogen to guarantee that the mass balances are closed and the conservation law fulfilled at all times for COD, carbon and nitrogen. Moreover, such an approach makes model verification much easier. The complete set of equations is presented in Rosen and Jeppsson (2006).

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) focuses 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) (Table B.4) 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 substances and their 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 in other respects, and due to numerical issues (we are subtracting very small and similar sized numbers) the errors of calculated outputs are much closer to the solution a 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:

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

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

$$K_{A,Bva}\left(S_{va-}\left(K_{a,va} + S_H^+\right) - K_{a,va}S_{va,total}\right)$$

and, consequently, change the stoichiometry since $S_{va,total}$ is not affected when the equilibrium of S_{va-} is changing. If we select the value of $K_{A,Bva}$ large enough (in our case 10^{10}) it is clear that this transformation has provided us with the same equation as described for the DAE implementation (Batstone *et al.*, 2002, Table B.3). The complete set of equations is given in Rosen and Jeppsson (2006). 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.

Algebraic solver for the pH (S_H^+) and S_{h2} states

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. An implicit algebraic equation for the pH calculation is given in Batstone *et al.* (2002) (Table B.3). The differential equation for the S_{h2} state, explicitly given in Rosen and Jeppsson (2006), can be approximated by an algebraic equation in a similar way as was the case for the pH state, simply by setting its differential to zero (assuming fast dynamics). The obtained implicit algebraic equations are non-linear and are solved by an iterative numerical method. The Newton–Raphson method used in Volcke *et al.* (2005) for calculation of the pH and equilibrium concentrations is implemented. By using this method the new value of the unknown state is calculated at each iteration step as:

$$S = S_0 - \frac{E(S_0)}{dE(S)/dS|_{S_0}}$$

where S_0 is the value of the state obtained from the previous iteration step and $E(S_0)$ is the value of the implicit algebraic equation that has to be zero for the equilibrium. The gradient of the algebraic equation $dE(S)/dS$ is also needed for calculation of the new state value. The iteration is repeated as long as $E(S_0)$ remains larger than the predefined tolerance value, which in our case is set to 10^{-12}. Normally only two or three iterations are required to solve the equation at each time step.

Inhibition functions

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. Expressions based on hyperbolic tangents are often used in chemical engineering:

$$I_{pH} = \frac{1}{2}(1 + \tan(a\varphi)), \text{ with } \varphi = \frac{pH - \frac{pH_{LL}+pH_{UL}}{2}}{\frac{pH_{LL}+pH_{UL}}{2}}$$

Values of $a = 11$ for $I_{pH,aa}$ and $a = 22$ for $I_{pH,ac}$ and $I_{pH,h2}$, respectively, are appropriate to fit the function given in Batstone *et al.* (2002). Another option is functions based on Hill functions:

$$I_{pH} = \frac{pH^n}{pH^n + K_{pH}^n}, \text{ with } K_{pH} = \frac{pH_{LL} + pH_{UL}}{2}$$

Siegrist *et al.* (2002) used a Hill inhibition function based on the hydrogen ion concentration instead. For the ADM1, this would give the following expression:

$$I_{pH} = \frac{K_{pH}^n}{S_H^{+n} + K_{pH}^n}, \text{ with } K_{pH} = 10^{-\frac{pH_{LL}+pH_{UL}}{2}}$$

The appropriate values of n in the in the respective Hill functions are quite different. To fit the original function given in Batstone *et al.* (2002), $n = 24$ for $I_{pH,aa}$ when the pH-based Hill function is used and $n = 2$ for the hydrogen ion-based function. For $I_{pH,ac}$ and $I_{pH,h2}$ the respective values of n are 45 and 3. It should be noted that the appropriate values of n are dependent of the values of pH_{LL} and pH_{UL}. For any practical purpose, the choice of function among the three above is arbitrary but for BSM2 the hydrogen ion based function is chosen.

The gas flow equation

In Batstone *et al.* (2002), two different expressions for the gas flow rate are given. For BSM2, the expression based on an over-pressure in the head space is used: $q_{gas} = k_p(P_{gas} - P_{atm})$. The reason for this choice is that the other expression gives a nervous behaviour in the flow rate, slowing down the simulations and propagating noise to other variables. A reasonable choice for the constant k_p is $5{\cdot}10^4$, which will yields an over-pressure at approximately 50 mbar in the head space.

Results and discussion

To evaluate the model implementations proposed in this paper, a number of simulation tests are carried out. These tests include: 1) steady-state simulations, 2) dynamic simulations for two weeks to compare the transient behaviour in detail, and 3) dynamic simulations for 609 days to compare overall simulation times. The model implementations investigated are: ODE – the differential equation implementation (Rosen and Jeppsson, 2006); DAE1 – differential equations with algebraic solution of pH (S_H^+) (Batstone *et al.*, 2002); DAE2 – differential equations with algebraic solution of pH and S_{h2}. All three models are tested as a part of the BSM2. This means that the behaviour reported here refers to the whole BSM2 system (Figure 1). The simulations are carried out using a modern PC running Windows XP and Matlab/Simulink Release 13. The ASM1, the clarifiers and the ADM1 are all implemented as MEX-files based on C source code.

Comparison of model results

The three model implementations are simulated for 200 days to reach steady state. Both relative and absolute errors are investigated using the ODE simulation as a reference. Only minor errors were encountered – the largest relative errors in the range of 10^{-11} and the largest absolute errors, 10^{-12}. The results are not surprising since the differences between the models are in the dynamic description of the equations.

Although the model implementations differ in the description of pH and S_{h2} dynamics, no significant differences are obtained when the transients in the states are investigated. The relative errors are typically in the range of 10^{-5} or smaller, again using the ODE implementation as reference. However, a first comparison between the ODE and the DAE implementations reveals a severe discrepancy between the pH dynamics of the respective implementations. This discrepancy can be traced to the value of the rate coefficient for the base to acid reactions. In Batstone *et al.* (2002), the suggested value for $K_{A,i}$ is 10^8 $M^{-1}d^{-1}$. This value is too small if the dynamic results of an ODE and a DAE implementation are to be compared. The value must be increased to 10^{10} $M^{-1}d^{-1}$ (or higher) for the models to give identical results.

Simulation speed

The simulation speed is tested using stiff and non-stiff solvers available in Matlab/Simulink. The three implementations are tested using models with sensors and controllers. The sensors

and controllers are simulated with and without discontinuities, noise and time delays. The conclusions from this study are that if the model is not affected by noise or discontinuities, the ODE implementation using a stiff solver is superior to the other implementations. However, if the model is affected by noise and discontinuities, the stiff solvers perform poorly. Using DAE2 with a non-stiff solver results in a simulation time for all 609 days of 50 minutes (relative integration tolerance of 10^{-5}). For ODE and DAE1 the corresponding simulations result in simulation times exceeding one day. It is interesting to note that the implementation of only a pH solver does not give any significant improvement in simulation speed and substantial improvement is not obtained unless the fast state associated with S_{h2} is removed. When stiff solvers are used for both ODE and DAE1, the simulation times are reduced to slightly less than one day. However, this is only achieved when the number of noisy and discontinuous sensors/controllers is limited to one or two. When more are implemented, stiff solvers fail completely to solve the systems.

To better understand this dramatic improvement in simulation speed, an analysis of the eigenvalues of the linearised ADM1 implementations, is used. Investigation of the eigenvalues of the system matrix provides an indication on how the distribution of time constants appears in the system at a certain operating point. From the analysis, it is evident that the eigenvalue associated with the pH (S_H^+) is a factor 10–1000 (depends on the chosen value of $K_{A,i}$) larger than the one associated with the S_{h2} state. However, the third largest eigenvalue is only about 1/100 of that of associated with S_{h2}. Clearly, both pH and S_{h2} must be removed to significantly reduce the stiffness of the system. However, it should be noted that the working point investigated is the anticipated working point of the BSM2.

Conclusions

In this paper, several aspects of the Matlab/Simulink implementation of the ADM1 for use in the BSM2 are discussed. A number of practical implementation issues are discussed, e.g. mass-balance discrepancies. It also is shown that optimising the computational efficiency of the BSM2 implies that the stiffness of the ADM1 must be overcome so that fast simulation is achieved for dynamic input data, using a solver that handles stochastic inputs. This means that the stiff solvers provided by Matlab/Simulink cannot be used. Instead, rewriting the system as a DAE system is the only possibility. It is shown that it is not sufficient to describe only pH as an algebraic state. Also the hydrogen state must be approximated by an algebraic equation to obtain satisfying results. The reformulation of the model results in a decrease in simulation time of the BSM2 evaluation period from approximately a day to less than an hour. It should be stated that development of the BSM2 is ongoing and the description given in this paper only describes the state of the work at this point. Minor changes may be imposed in the future.

Acknowledgements

The authors would like to express their gratitude to Eveline Volcke and Usama Zaher (BIOMATH, Ghent University), Dr John Copp (Primodal Inc., Hamilton) and Dr Jens Alex (ifak system GmbH, Magdeburg) for providing us their source codes and helping us understand the intrinsic mysteries of pH solvers. Dr Damien Batstone (AWMC, University of Queensland) has been the perfect partner for discussing all types of issues related to details of the ADM1. The financial support provided for Dr Vrecko and Dr Gernaey through the European Community's Human Potential Programme under contract HPRN-CT-2001-00200 (WWT&SYSENG) is gratefully acknowledged.

References

Batstone, D.J., Keller, J., Angelidaki, R.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*, 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 Communities, Luxembourg.

Copp, J., Jeppsson, U. and Rosen, C. (2003). Towards an ASM1-ADM1 state variable interface for plant-wide wastewater treatment modelling. *76th Annual WEF Conference and Exposition*, Oct. 11–15, Los Angeles, CA, USA.

Gernaey, K.V., Rosen, C. and Jeppsson, U. (2005). Phenomenological modelling of wastewater treatment plant influent disturbance scenarios. *10th Int. Conf. on Urban Drainage*, Aug. 21–26, Copenhagen, Denmark.

Jeppsson, U., Rosen, C., Alex, J., Copp, J., Gernaey, K.V., Pons, M.-N. and Vanrolleghem, P.A. (2006). Towards a benchmark simulation model for plant-wide control strategy performance evaluation of WWTPs. *Wat. Sci. Tech.*, **53**(1), 287–295.

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

Rosen, C. and Jeppsson, U. (2006). Description of the ADM1 for benchmark simulations. Technical Report, Department of Industrial Electrical Engineering and Automation (IEA), Lund University, Lund, Sweden.

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

Vanrolleghem, P.A., Rosen, C., Zaher, U., Copp, J., Benedetti, L., Ayesa, E. and Jeppsson, U. (2005). Continuity-based interfacing of models for wastewater systems described by Petersen matrices. *Wat. Sci. Tech.*, **52**(1–2), 493–500.

Volcke, E.I.P., Van Hulle, S., Deksissa, T., Zaher, U. and Vanrolleghem, P.A. (2005). Calculation of pH and concentration of equilibrium components during dynamic simulation by means of a charge balance. BIOMATH Tech. Report, Ghent University, Ghent, Belgium.

Zaher, U., Grau, P., Benedetti, L., Ayesa, E. and Vanrolleghem, P.A. (2006). Transformers for interfacing anaerobic digestion models to pre- and post-treatment processes. *Environ. Model. Softw.*, (in press).