Investigation of Anaerobic Digestion Alternatives for Henriksdal's WWTP

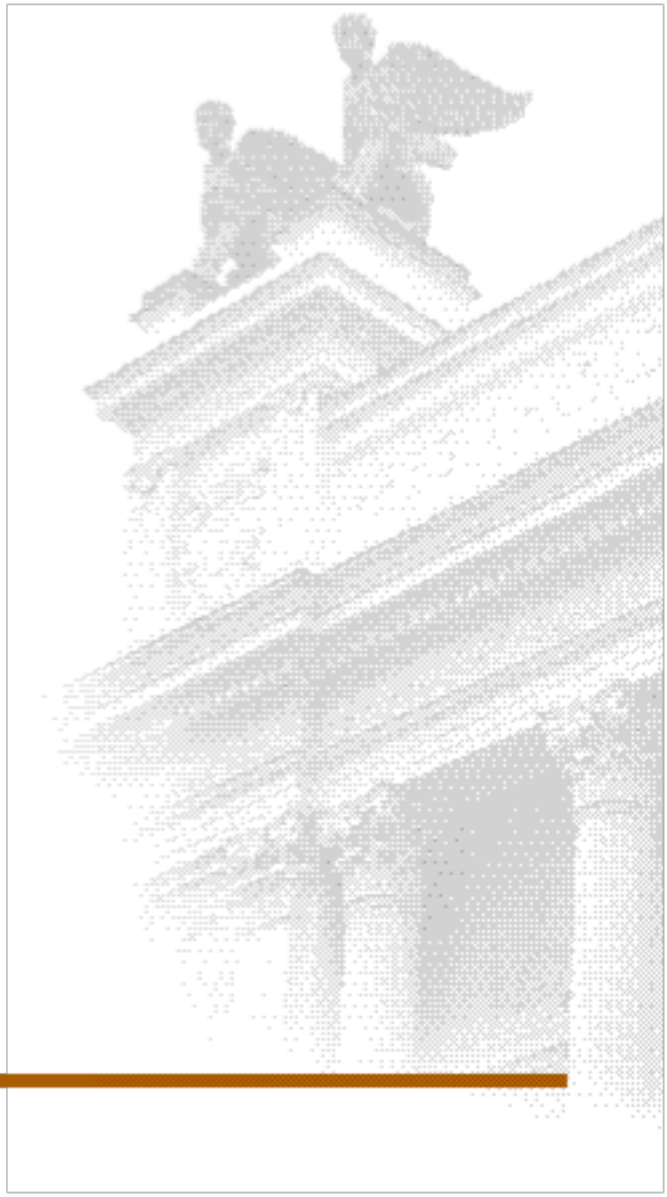
Jeppsson, Ulf

2007

CODEN:LUTEDX/(TEIE-7225)/1-111/(2007)

Industrial Electrical Engineering and Automation

Investigation of Anaerobic Digestion Alternatives for Henriksdal's WWTP



Ulf Jeppsson

Dept. of Industrial Electrical Engineering and Automation
Lund University

Investigation of anaerobic digestion alternatives for Henriksdal's WWTP

by

Dr Ulf Jeppsson

Contact information:

IEA, Lund University
PO Box 118, SE-221 00 Lund
Sweden
phone: +46 (0)46 222 92 87
email: ulf.jeppsson@iea.lth.se

Contents

1. Introduction……3
2. Methodology…… 3
3. Influent characteristics…… 4
4. Results……7
   4.1 Case 1A……10
   4.2 Case 1B……14
   4.3 Case 2A……16
   4.4 Case 2B……22
   4.5 Case 3A……26
   4.6 Case 3B……30
   4.7 Case 4A……32
   4.8 Case 4B……38
5. Conclusions……42

Appendix A. C code of ASM2ADM interface model 43
Appendix B. C code of ADM2ASM interface model 53
Appendix C. Example of initialisation file for simulations 60
Appendix D. C code of ADM1 64
Appendix E. Example of simulation layout (case 1A)..... 74
Appendix F. Detailed report of ADM1 and added modifications..... 75

Investigation of anaerobic digestion alternatives for Henriksdal's WWTP

Dr Ulf Jeppsson
IEA, Lund University
PO Box 118, 221 00 Lund, Sweden
phone: +46 (0)46 222 92 87
email: ulf.jeppsson@iea.lth.se

1. Introduction

On the request of Dr Daniel Hellström (Stockholm Water AB), IEA was asked to perform a preliminary investigation of two alternatives for anaerobic digestion operation at Henriksdal's WWTP in Stockholm, Sweden. The system should be analysed based on parallel or series operation of two existing AD reactors. Moreover, the input load should be based on the current situation and a future scenario (estimated 10 years ahead).

2. Methodology

The system is modelled using the IWA Anaerobic Digestion Model no 1 (ADM1). This represents the state-of-the art model for AD systems. Only COD and nitrogen is considered in the model. The model used is a slightly modified version of the original one (enhancements supported by the IWA ADM1 Task Group) and the updated version has already been distributed to many research groups world-wide. The details of this model can be found in the appendices, where a report describing the model is provided together with the C code of the actual model. Values for all model parameters can also be found here. The system is only investigated for steady-state behaviour in this preliminary analysis. Special interface models for connecting the ADM1 to the more traditional models (e.g. ASM1) have also been developed in Lund and the Henriksdal system will therefore be analysed from two further aspects (see next section). The C code for the interface models are also provided in the appendices. The entire system is simulated using Matlab/Simulink. A layout of the simulated system (case 1A, see below) is shown as an example in the appendices.

For all simulations the temperature of the AD systems is set to 35 °C (mesophilic conditions) and the concentrations of anions and cations in the influent sludge are set so that pH in the digesters is between 7.15 and 7.3, to avoid any significant unwanted effects due to pH inhibition.

The total volume of the AD reactors is 38900 m^3. One reactor (step 1 in series operation) is 23700 m^3 and the second one (step two in series operation) is 15200 m^3. It is assumed that these volumes represent the liquid phase of the reactors and an extra head space volume of about 10% is added to each reactor (a slight over pressure of about 5 mbar is maintained in the head space). During all simulations the liquid and head space volumes are assumed constant. When the system is simulated in parallel operation only one reactor of 38900 m^3 is

used (from a model point of view the results are the same as if using two smaller reactors in parallel).

3. Influent characteristics

The information available regarding the influent sludge is very limited. For the situation today the following data were provided:

Average flow rate: 2040 m^3/d
Primary sludge: 55.3 ton TS/d, 40.1 ton VS/d
Secondary sludge from AS system: 19.9 ton TS/d, 12.3 ton VS/d
External organic material: 7.2 ton TS/d, 6.8 ton VS/d (considered to be 100% lipids)

For the future scenario the following data were provided:

Average flow rate: 2544 m^3/d
Primary sludge: 62.2 ton TS/d, 45.1 ton VS/d
Secondary sludge from AS system: 22.1 ton TS/d, 13.6 ton VS/d
External organic material: 33.6 ton TS/d, 31.9 ton VS/d (considered to be 100% lipids)

As the average total input solids concentration is about 40 kg TS/m^3 in the current case and 46 kg/m^3 in the future case, it would appear that a thickener is available prior to the AD inlet (although no information about this was provided). The average total AD retention time for the current case is 19 days and for the future case about 15 days.

The input can be created based on direct ADM1 state variables or by creating the input based on ASM1 variables and allowing these to be transformed into ADM1 variables via the interface model. Both cases are tested. At all times the external organic material is added to the AD model using the direct ADM1 state variable 'lipids' (and a small part of inert particulate material, i.e. TS-VS using the same COD content as for lipids). As details of the potential thickener are not available and the individual flow rates from the primary and secondary settlers are not provided, several assumptions must be made. Moreover, the ADM1 model is based on COD content. We assume:

- 1.42 kg COD/kg TS (for both total primary and total secondary sludge);
- 1.42 kg COD/kg VS (for both primary and secondary sludge without inerts, leading to a value of about 2 kg COD/kg VSS for the total sludge input);
- contributions from soluble COD are not considered (very low in comparison);
- inert COD does not contribute to VS;
- input oxygen concentration is set to zero;
- in primary sludge only inert particulate material and particulate substrate are considered;
- in secondary sludge only inert particulate material and biomass are considered;
- a reasonable amount of nitrogen is included;
- COD content of lipids is assumed as 1.86 g COD/g lipid (based on $C_{57}H_{104}O_6$);
- VS = TS for lipids.

Obviously the input characterisation can be greatly improved if more information is made available. However, for this preliminary investigation it may suffice. Based on the above assumptions, the influent sludge characteristics for the current case (in ASM1 variables) can be set to:

CURRENT CASE	Primary sludge (concentration)	Primary sludge (kg "unit"/d)	Secondary sludge (concentration)	Secondary sludge (kg "unit"/d)
Inert soluble (S_I, mg COD/l)	0	0	0	0
Soluble substrate (S_S, mg COD/l)	0	0	0	0
Inert particulate (X_I, mg COD/l)	19500	21567	13500	10746
Particulate substrate (X_S, mg COD/l)	51500	56959	0	0
Heterotrophs (X_{BH}, mg COD/l)	0	0	22000	17512
Autotrophs (X_{BA}, mg COD/l)	0	0	0	0
Inert from biomass decay (X_P, mg COD/l)	0	0	0	0
Oxygen (S_O, mg –COD/l	0	0	0	0
Nitrate (S_{NO}, mg N/l)	0	0	5	4
Ammonia (S_{NH}, mg N/l)	30	33,2	5	4
Soluble organic N (S_{ND}, mg N/l)	0	0	0	0
Particulate organic N (X_{ND}, mg N/l)	2060	2278	0	0
Alkalinity (mol HCO_3/m^3)	7		7	
Total COD (mg COD/l)	71000	78526	35500	28258
TS (mg/l)	50000	55300	25000	19900
VS (mg/l)	36257	40100	15452	12300
Flow rate (m^3/d)	1106	1106000	796	796000

In can be clearly seen that several variables have been neglected: oxygen, soluble inerts, readily biodegradable material are all set to zero, we set autotrophs to zero and only consider heterotrophs in the secondary sludge, all inerts are included as X_I rather than a combination of X_I and X_P etc. From the point of the digester these simplification have a limited impact. Note that in ASM1 variables the nitrogen content of biomass and inert material is not stated explicitly but it is added as part of the model interface. However, with more information about the influent sludge a more detailed fractionation should be performed. Flow rates for the primary and secondary sludge have been assumed in order to calculate concentrations to simplify the characterisation but obviously the sludge flows are all combined before entering the digester so it has no practical implications on the overall analysis. The remaining flow of 138 m^3/d is assumed to be associated with the external organic material. For this external source we assume:

Flow rate 138 m^3/d, lipids 91600 mg COD/l, inert particulate material 4100 mg COD/l, which provides us with 6800 kg VS and 7200 kg TS per day. We neglect any small nitrogen part associated with the inerts in this external source (lipids themselves contain no nitrogen). This source will always be fed directly into the digester without use of any interface models.

In the case when the interface models are not used we simply put all COD in primary and secondary sludge into the digester as composite material (which will then divide it into proteins, carbohydrates, lipids, soluble inerts and particulate inerts in accordance with predefined values of the ADM1 model). The total COD load is based on the assumption 1.42 kg COD/kg TSS, i.e. the total COD load on the digester will be the same in both cases, but it will enter the system in very different form. The value predicted by the interface model for total inorganic nitrogen will be used also for the nitrogen input in this latter case, i.e. the inorganic nitrogen loads will in both cases be identical (but the organic nitrogen load will differ). The input concentration of cations will be used to adjust the pH to a reasonable value

The same principles are then used to define the influent sludge characteristics for the future scenario. The values for primary and secondary sludge are shown in the table below. The external sludge is fed directly into the digester as lipids (and a small part of particulate inerts).

FUTURE CASE	Primary sludge (concentration)	Primary sludge (kg "unit"/d)	Secondary sludge (concentration)	Secondary sludge (kg "unit"/d)
Inert soluble (S_I, mg COD/l)	0	0	0	0
Soluble substrate (S_S, mg COD/l)	0	0	0	0
Inert particulate (X_I, mg COD/l)	19500	24258	13500	11934
Particulate substrate (X_S, mg COD/l)	51500	64066	0	0
Heterotrophs (X_{BH}, mg COD/l)	0	0	22000	19448
Autotrophs (X_{BA}, mg COD/l)	0	0	0	0
Inert from biomass decay (X_P, mg COD/l)	0	0	0	0
Oxygen (S_O, mg –COD/l	0	0	0	0
Nitrate (S_{NO}, mg N/l)	0	0	5	4,4
Ammonia (S_{NH}, mg N/l)	30	37,3	5	4,4
Soluble organic N (S_{ND}, mg N/l)	0	0	0	0
Particulate organic N (X_{ND}, mg N/l)	2060	2563	0	0
Alkalinity (mol HCO_3/m^3)	7		7	
Total COD (mg COD/l)	71000	88324	35500	31382
TS (mg/l)	50000	62200	25000	22100
VS (mg/l)	36254	45100	15385	13600
Flow rate (m^3/d)	1244	1244000	884	884000

Flow rates for the primary and secondary sludge have been assumed in order to calculate concentrations to simplify the characterisation but obviously the sludge flows are all combined before entering the digester so it has no practical implications on the overall analysis. The remaining flow of 416 m^3/d is assumed to be associated with the external organic material. For this external source we assume:

Flow rate 416 m^3/d, lipids 142600 mg COD/l, inert particulate material 5800 mg COD/l, which provides us with 31900 kg VS and 33600 kg TS per day. We neglect any small nitrogen part associated with the inerts in this external source (lipids themselves contain no nitrogen). This source will always be fed directly into the digester without use of any interface models.

4. Results

Based on the rough characterisation of the various sludge sources we can run the simulations. The detailed results are found below. Eight cases are presented:

- case 1A: Current case using interface models, operation in parallel (1 reactor);
- case 1B: Current case not using interface models, operation in parallel (1 reactor);
- case 2A: Current case using interface models, operation in series (2 reactors);
- case 2B: Current case not using interface models, operation in series (2 reactors);
- case 3A: Future case using interface models, operation in parallel (1 reactor);
- case 3B: Future case not using interface models, operation in parallel (1 reactor);
- case 4A: Future case using interface models, operation in series (2 reactors);
- case 4B: Future case not using interface models, operation in series (2 reactors);

In all cases with two reactors in series the external organic material is fed into the first reactor together with the primary and secondary sludge. The model parameters of the digester(s) are identical for all cases, apart from the obvious differences in volumes. Also when two reactors are simulated in series the parameter set-up is identical for both reactors. In all cases the digesters are assumed to be completely mixed.

The numbers need to be studied in detail but this will be an effort for Stockholm Water. It is however clear that in terms of gas production the results are similar for cases 1A, 1B, 2A and 2B (the same holds for cases 3 and 4), which means that the effects of the model interfaces do not have a significant impact and also the effects of operation in parallel versus operation in series is limited. Operation in series will lead to a higher gas production but this is mainly due to effect of applying complete mix in two rather than one reactor (the same effects that we see when modelling a large completely mixed AS reactor compared to a plug-flow system of the same volume).

The ADM1 is a very robust model and it can be highly loaded before any collapse of the system will appear. Consequently, we see that the model predicts the majority of the gas production in the first reactor (when using in series operation) and a very small contribution from the second reactor. However, the difference is quite clear when using input via the

interfaces or the direct approach. When the direct approach is used the input enters the system as composite material and the slowest process within the ADM1 is the disintegration process from composite material into lipids, carbohydrates, proteins and inerts. What the interface does is that it considers this process to be immediate (in terms of the input material) since the input is converted directly into lipids, proteins etc. It should, however, be noted that the disintegration process is still used also in this case but only for the internal digester material, i.e. when biomass decay and gradually forms new substrate. This is actually a main reason for the development of the interface, since it allows a separation between disintegration of incoming sludge from the disintegration of internal digester material, as the disintegration rate of decaying biomass and influent sludge (e.g. primary sludge with a high amount of fairly easily available substrate) are believed to be quite different. In the cases shown below we are demonstrating the two extremes, i.e. all influent material as composite material or no influent material as composite material.

The choice of using identical model parameters for the first and second reactor is also a simplification that may not be fully realistic. The first reactor (in series operation) is a highly loaded system whereas the second digester is low loaded. Naturally this can lead to different biomass populations and thereby promote different pathways of the digester process.

Based on the results below the amount of produced methane can be calculated. Note that the overpressure of the head space should be taken into account. For example, for case 1A the produced methane in kg CH_4/d can be calculated as (P_atm = 1.013 bar):

qgas * Sgas,ch4 * P_atm/pgas,total = 70931.2622 kg COD/d (as the gas flow is normalised to atmospheric pressure to simplify comparisons)

which can then be further converted as we know that 1 mol CH_4 is equal to 64 g COD and the weight of CH_4 is 16 g/mole and consequently

70931.2622 kg COD(CH_4)/d = 70931.2622 * 16/64 = 70931.2622 * 0.25 = 17732.8 kg CH_4/d

Or the alternative below can be used (based on the general gas law):

pgas,ch4 * P_atm/pgas,total * V = n*R*T_op

0.63731 * 1.013/1.0175 *1 *16/(0.083145
* (35 + 273.15)) = 0.39623 kg CH_4/m^3

and finally

0.39623 * qgas = 17732.4 kg CH_4/d

The small difference between these two results can be attributed to round-off errors. In a similar way the mass of produced CO_2 and H_2 can be calculated (note 1 mole of H_2 is equal to 16 g COD, and the gas phase concentration of CO_2 is given directly in kmole C/m^3). Knowing that the energy content of 1 kg CH_4 is equal to 50.014 MJ we can also easily determine the energy content of the produced gas. But for your convenience the author has also included these values directly in the results shown below.

Digesters are often characterized in terms of VS. This is difficult in this case since VS is not a variable considered in the ADM1, which is instead based on COD. From a theoretical point of view we can consider the theoretical maximum biogas potential (Nm^3 CH_4 per ton VS) for the standard components of the sludge:

- Fat (based on $C_{57}H_{104}O_6$) = 1014 Nm^3 CH_4 per ton VS;
- Protein (based on $C_5H_7NO_2$) = 496 Nm^3 CH_4 per ton VS;
- Carbohydrate (based on $(C_6H_{10}O_5)_n$ = 415 Nm^3 CH_4 per ton VS.

For the current scenario we have an input of 59.2 tonnes of VS (according to numbers provided by Stockholm Water). The methane gas produced in the simulations for this case (1A) is about 28000 Nm^3 CH_4, which amounts to 473 Nm^3 CH_4 per ton VS. This appears to be a reasonable value since the load of lipids into the digester is very high.

For the future scenario we have an input of 90.6 tonnes of VS (according to numbers provided by Stockholm Water). The methane gas produced in the simulations for this case (3A) is about 47200 Nm^3 CH_4, which amounts to 520 Nm^3 CH_4 per ton VS. This also appears to be a reasonable value since the load of lipids into the digester is even higher in this scenario.

Also the volume fraction of CH_4 in the produced biogas ranges between 61 and 64%, which is also a reasonable number when compared to literature sources.

It is, however, very difficult to make a reasonable estimation of the amount of VS reduced in the digester. We can look at the cases where we have used the ADM2ASM interface to change the ADM1 variables back into ASM1 variables and then we can use the same principles for calculating VS in the sludge output as we used when creating the input variables. The uncertainty associated to this approach is significant as the VS relation to COD and TS is likely to be quite different in the digested sludge than in the input sludge. Moreover, the interface immediately converts all biomass into substrate and inerts which changes the composition of the sludge in terms of VS (but is consistent in terms of COD). Some rough estimates indicate a VS reduction of 80-90%, which would be a high value. Naturally, the ADM1 with its current set up simulates a digester where everything is working optimally, which is seldom the case in reality. As the ASM2ADM interface removes the slow disintegration process for the influent sludge it could also be argued that the retention time of the digester is actually higher than what the simple flow rate and volume calculations indicate as an effect of this. For example, the volume based retention time of about 20 days in case 1A may actually be more like 30 days as a result of the interface.

Due to similar effects also the TS reduction is high in the simulations (50 to 60%). This problem is emphasised by the fact that non-organic particulate material is not modelled in ADM1. It is clear that sludge contains a significant amount of non-organic particulate material, which will contribute to TS and not be affected by any biological treatment. As we have converted all sludge input into organic material (although part of it inert) we will allow the model to degrade more TS than would otherwise be possible.

4.1 CASE 1A

Combined input of primary and secondary sludge prior to ASM2ADM interface:

SI = 0 mg COD/l
SS = 0 mg COD/l
XI = 16988.959 mg COD/l
XS = 29946.898 mg COD/l
XBH = 9207.1504 mg COD/l
XBA = 0 mg COD/l
XP = 0 mg COD/l
SO = 0 mg -COD/l
SNO = 2.0925 mg N/l
SNH = 19.5373 mg N/l
SND = 0 mg N/l
XND = 1197.8759 mg N/l
SALK = 7 mol HCO3/m3
TSS = 39537.3291 mg SS/l
Flow rate = 1902 m3/d
Temperature = 15 degC

SI_load = 0 kg COD/d
SS_load = 0 kg COD/d
XI_load = 32313 kg COD/d
XS_load = 56959 kg COD/d
XBH_load = 17512 kg COD/d
XBA_load = 0 kg COD/d
XP_load = 0 kg COD/d
SO_load = 0 kg -COD/d
SNO_load = 3.98 kg N/d
SNH_load = 37.16 kg N/d
SND_load = 0 kg N/d
XND_load = 2278.36 kg N/d
SALK_load = 13.314 kmol HCO3/d
TSS_load = 75200 kg SS/d

Combined primary and secondary sludge after ASM2ADM interface

adm_in[1] : Ssu = monosacharides (kg COD/m3) = 0
adm_in[2] : Saa = amino acids (kg COD/m3) = 0
adm_in[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0
adm_in[4] : Sva = total valerate (kg COD/m3) = 0
adm_in[5] : Sbu = total butyrate (kg COD/m3) = 0
adm_in[6] : Spro = total propionate (kg COD/m3) = 0
adm_in[7] : Sac = total acetate (kg COD/m3) = 0
adm_in[8] : Sh2 = hydrogen gas (kg COD/m3) = 0
adm_in[9] : Sch4 = methane gas (kg COD/m3) = 0
adm_in[10] : Sic = inorganic carbon (kmole C/m3) = 0.0082697
adm_in[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.0013955

adm_in[12] : Si = soluble inerts (kg COD/m3) = 0
adm_in[13] : Xc = composites (kg COD/m3) = 0
adm_in[14] : Xch = carbohydrates (kg COD/m3) = 5.6445
adm_in[15] : Xpr = proteins (kg COD/m3) = 17.9354
adm_in[16] : Xli = lipids (kg COD/m3) = 12.6219
adm_in[17] : Xsu = sugar degraders (kg COD/m3) = 0
adm_in[18] : Xaa = amino acid degraders (kg COD/m3) = 0
adm_in[19] : Xfa = LCFA degraders (kg COD/m3) = 0
adm_in[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0
adm_in[21] : Xpro = propionate degraders (kg COD/m3) = 0
adm_in[22] : Xac = acetate degraders (kg COD/m3) = 0
adm_in[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0
adm_in[24] : Xi = particulate inerts (kg COD/m3) = 19.9352
adm_in[25] : Scat+ = cations (base) (kmole/m3) = 0
adm_in[26] : San- = anions (acid) (kmole/m3) = 0.0057538
adm_in[27] : Flow rate (m3/d) = 1902
adm_in[28] : Temperature (degC) = 35

Combined primary and secondary sludge after ASM2ADM interface + external sludge input, i.e. total input to digester

adm_in[1] : Ssu = monosacharides (kg COD/m3) = 0
adm_in[2] : Saa = amino acids (kg COD/m3) = 0
adm_in[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0
adm_in[4] : Sva = total valerate (kg COD/m3) = 0
adm_in[5] : Sbu = total butyrate (kg COD/m3) = 0
adm_in[6] : Spro = total propionate (kg COD/m3) = 0
adm_in[7] : Sac = total acetate (kg COD/m3) = 0
adm_in[8] : Sh2 = hydrogen gas (kg COD/m3) = 0
adm_in[9] : Sch4 = methane gas (kg COD/m3) = 0
adm_in[10] : Sic = inorganic carbon (kmole C/m3) = 0.0073475
adm_in[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.0013011
adm_in[12] : Si = soluble inerts (kg COD/m3) = 0
adm_in[13] : Xc = composites (kg COD/m3) = 0
adm_in[14] : Xch = carbohydrates (kg COD/m3) = 5.2627
adm_in[15] : Xpr = proteins (kg COD/m3) = 16.7221
adm_in[16] : Xli = lipids (kg COD/m3) = 17.9645
adm_in[17] : Xsu = sugar degraders (kg COD/m3) = 0
adm_in[18] : Xaa = amino acid degraders (kg COD/m3) = 0
adm_in[19] : Xfa = LCFA degraders (kg COD/m3) = 0
adm_in[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0
adm_in[21] : Xpro = propionate degraders (kg COD/m3) = 0
adm_in[22] : Xac = acetate degraders (kg COD/m3) = 0
adm_in[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0
adm_in[24] : Xi = particulate inerts (kg COD/m3) = 18.864
adm_in[25] : Scat+ = cations (base) (kmole/m3) = 0
adm_in[26] : San- = anions (acid) (kmole/m3) = 0.0053644
adm_in[27] : Flow rate (m3/d) = 2040

adm_in[28] : Temperature (degC) = 35

Digester output (steady state)

adm_out[1] : Ssu = monosacharides (kg COD/m3) = 0.012386
adm_out[2] : Saa = amino acids (kg COD/m3) = 0.0055395
adm_out[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0.10732
adm_out[4] : Sva = total valerate (kg COD/m3) = 0.012146
adm_out[5] : Sbu = total butyrate (kg COD/m3) = 0.01413
adm_out[6] : Spro = total propionate (kg COD/m3) = 0.01757
adm_out[7] : Sac = total acetate (kg COD/m3) = 0.089858
adm_out[8] : Sh2 = hydrogen gas (kg COD/m3) = 2.5038e-07
adm_out[9] : Sch4 = methane gas (kg COD/m3) = 0.056508
adm_out[10] : Sic = inorganic carbon (kmole C/m3) = 0.092091
adm_out[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.092179 (= 1.2905 kg N/m3)
adm_out[12] : Si = soluble inerts (kg COD/m3) = 0.14081
adm_out[13] : Xc = composites (kg COD/m3) = 0.14768
adm_out[14] : Xch = carbohydrates (kg COD/m3) = 0.028924
adm_out[15] : Xpr = proteins (kg COD/m3) = 0.088706
adm_out[16] : Xli = lipids (kg COD/m3) = 0.095922
adm_out[17] : Xsu = sugar degraders (kg COD/m3) = 0.46458
adm_out[18] : Xaa = amino acid degraders (kg COD/m3) = 0.97929
adm_out[19] : Xfa = LCFA degraders (kg COD/m3) = 0.75009
adm_out[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0.36257
adm_out[21] : Xpro = propionate degraders (kg COD/m3) = 0.11958
adm_out[22] : Xac = acetate degraders (kg COD/m3) = 0.96855
adm_out[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0.43465
adm_out[24] : Xi = particulate inerts (kg COD/m3) = 19.1456
adm_out[25] : Scat+ = cations (base) (kmole/m3) = 3.8695e-26
adm_out[26] : San- = anions (acid) (kmole/m3) = 0.0053644
adm_out[27] : Flow rate (m3/d) = 2040
adm_out[28] : Temperature (degC) = 35
adm_out[34] : pH = pH within AD system = 7.2783
adm_out[35] : S_H+ = protons (kmole/m3) = 5.2692e-08
adm_out[36] : Sva- = valerate (kg COD/m3) = 0.0121
adm_out[37] : Sbu- = butyrate (kg COD/m3) = 0.014081
adm_out[38] : Spro- = propionate (kg COD/m3) = 0.0175
adm_out[39] : Sac- = acetate (kg COD/m3) = 0.089586
adm_out[40] : Shco3- = bicarbonate (kmole C/m3) = 0.08321
adm_out[41] : Sco2 = carbon dioxide (kmole C/m3) = 0.0088807
adm_out[42] : Snh3 = ammonia (kmole N/m3) = 0.0019023
adm_out[43] : Snh4+ = ammonium (kmole N/m3) = 0.090277
adm_out[44] : Sgas,h2 = hydrogen concentration in gas phase (kg COD/m3) = 1.0159e-05
adm_out[45] : Sgas,ch4 = methane concentration in gas phase (kg COD/m3) = 1.592
adm_out[46] : Sgas,co2 = carbon dioxide concentration in gas phase (kmole C/m3) = 0.012664
adm_out[47] : pgas,h2 = partial pressure of hydrogen gas (bar, true value i.e. not normalized) = 1.6267e-05

adm_out[48] : pgas,ch4 = partial pressure of methane gas (bar, true value i.e. not normalized) = 0.63731
adm_out[49] : pgas,co2 = partial pressure of carbon dioxide gas (bar, true value, i.e. not normalized) = 0.32447
adm_out[50] : pgas,total = total head space pressure of H2+CO2+CH4+H2O (bar, true value, i.e. not normalized) = 1.0175
adm_out[51] : qgas = gas flow rate normalized to atmospheric pressure (m3/d) = 44752.7366
Extra calculated outputs
Produced hydrogen gas (kg H2/d) = 0.056579
Produced methane gas (kg CH4/d) = 17733.0456
Produced carbon dioxide gas (kg CO2/d) = 24827.6976
Energy content of methane gas (MJ/d) = 886900.5421
Energy content of methane gas (kWh/d) = 246361.2617

Digester sludge output after ADM2ASM interface

SI = 140.8052 mg COD/l
SS = 258.9446 mg COD/l
XI = 19145.6497 mg COD/l
XS = 3583.8855 mg COD/l
XBH = 0 mg COD/l
XBA = 0 mg COD/l
XP = 856.6541 mg COD/l
SO = 0 mg -COD/l
SNO = 0 mg N/l
SNH = 1452.7286 mg N/l
SND = 0.54287 mg N/l
XND = 135.4177 mg N/l
SALK = 98.4016 mol HCO3/m3
TSS = 16609.9925 mg SS/l
Flow rate = 2040 m3/d
Temperature = 15 degC

SI_load = 287.2427 kg COD/d
SS_load = 528.247 kg COD/d
XI_load = 39057.1254 kg COD/d
XS_load = 7311.1264 kg COD/d
XBH_load = 0 kg COD/d
XBA_load = 0 kg COD/d
XP_load = 1747.5743 kg COD/d
SO_load = 0 kg -COD/d
SNO_load = 0 kg N/d
SNH_load = 2963.5663 kg N/d
SND_load = 1.1075 kg N/d
XND_load = 276.2522 kg N/d
SALK_load = 200.7393 kmol HCO3/d
TSS_load = 33884.3846 kg SS/d

4.2 CASE 1B

Total input to digester

adm_in[1] : Ssu = monosacharides (kg COD/m3) = 0
adm_in[2] : Saa = amino acids (kg COD/m3) = 0
adm_in[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0
adm_in[4] : Sva = total valerate (kg COD/m3) = 0
adm_in[5] : Sbu = total butyrate (kg COD/m3) = 0
adm_in[6] : Spro = total propionate (kg COD/m3) = 0
adm_in[7] : Sac = total acetate (kg COD/m3) = 0
adm_in[8] : Sh2 = hydrogen gas (kg COD/m3) = 0
adm_in[9] : Sch4 = methane gas (kg COD/m3) = 0
adm_in[10] : Sic = inorganic carbon (kmole C/m3) = 0
adm_in[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.0013011
adm_in[12] : Si = soluble inerts (kg COD/m3) = 0
adm_in[13] : Xc = composites (kg COD/m3) = 52.3451
adm_in[14] : Xch = carbohydrates (kg COD/m3) = 0
adm_in[15] : Xpr = proteins (kg COD/m3) = 0
adm_in[16] : Xli = lipids (kg COD/m3) = 6.1965
adm_in[17] : Xsu = sugar degraders (kg COD/m3) = 0
adm_in[18] : Xaa = amino acid degraders (kg COD/m3) = 0
adm_in[19] : Xfa = LCFA degraders (kg COD/m3) = 0
adm_in[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0
adm_in[21] : Xpro = propionate degraders (kg COD/m3) = 0
adm_in[22] : Xac = acetate degraders (kg COD/m3) = 0
adm_in[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0
adm_in[24] : Xi = particulate inerts (kg COD/m3) = 0.2774
adm_in[25] : Scat+ = cations (base) (kmole/m3) = 0.04
adm_in[26] : San- = anions (acid) (kmole/m3) = 0.0053645
adm_in[27] : Flow rate (m3/d) = 2040
adm_in[28] : Temperature (degC) = 35

Digester output (steady state)

adm_out[1] : Ssu = monosacharides (kg COD/m3) = 0.012402
adm_out[2] : Saa = amino acids (kg COD/m3) = 0.0055467
adm_out[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0.1075
adm_out[4] : Sva = total valerate (kg COD/m3) = 0.01113
adm_out[5] : Sbu = total butyrate (kg COD/m3) = 0.014828
adm_out[6] : Spro = total propionate (kg COD/m3) = 0.017598
adm_out[7] : Sac = total acetate (kg COD/m3) = 0.054483
adm_out[8] : Sh2 = hydrogen gas (kg COD/m3) = 2.5071e-07
adm_out[9] : Sch4 = methane gas (kg COD/m3) = 0.057314
adm_out[10] : Sic = inorganic carbon (kmole C/m3) = 0.083784
adm_out[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.042586 (= 0.5962 kg N/m3)
adm_out[12] : Si = soluble inerts (kg COD/m3) = 4.8736
adm_out[13] : Xc = composites (kg COD/m3) = 5.1117
adm_out[14] : Xch = carbohydrates (kg COD/m3) = 0.05085

adm_out[15] : Xpr = proteins (kg COD/m3) = 0.05085
adm_out[16] : Xli = lipids (kg COD/m3) = 0.1086
adm_out[17] : Xsu = sugar degraders (kg COD/m3) = 0.776
adm_out[18] : Xaa = amino acid degraders (kg COD/m3) = 0.56123
adm_out[19] : Xfa = LCFA degraders (kg COD/m3) = 0.84984
adm_out[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0.2431
adm_out[21] : Xpro = propionate degraders (kg COD/m3) = 0.1178
adm_out[22] : Xac = acetate degraders (kg COD/m3) = 0.9352
adm_out[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0.45698
adm_out[24] : Xi = particulate inerts (kg COD/m3) = 10.0246
adm_out[25] : Scat+ = cations (base) (kmole/m3) = 0.04
adm_out[26] : San- = anions (acid) (kmole/m3) = 0.0053645
adm_out[27] : Flow rate (m3/d) = 2040
adm_out[28] : Temperature (degC) = 35
adm_out[34] : pH = pH within AD system = 7.2515
adm_out[35] : S_H+ = protons (kmole/m3) = 5.6039e-08
adm_out[36] : Sva- = valerate (kg COD/m3) = 0.011085
adm_out[37] : Sbu- = butyrate (kg COD/m3) = 0.014773
adm_out[38] : Spro- = propionate (kg COD/m3) = 0.017523
adm_out[39] : Sac- = acetate (kg COD/m3) = 0.054308
adm_out[40] : Shco3- = bicarbonate (kmole C/m3) = 0.075243
adm_out[41] : Sco2 = carbon dioxide (kmole C/m3) = 0.0085406
adm_out[42] : Snh3 = ammonia (kmole N/m3) = 0.00082735
adm_out[43] : Snh4+ = ammonium (kmole N/m3) = 0.041759
adm_out[44] : Sgas,h2 = hydrogen concentration in gas phase (kg COD/m3) = 1.0243e-05
adm_out[45] : Sgas,ch4 = methane concentration in gas phase (kg COD/m3) = 1.6225
adm_out[46] : Sgas,co2 = carbon dioxide concentration in gas phase (kmole C/m3) = 0.012182
adm_out[47] : pgas,h2 = partial pressure of hydrogen gas (bar, true value i.e. not normalized) = 1.6402e-05
adm_out[48] : pgas,ch4 = partial pressure of methane gas (bar, true value i.e. not normalized) = 0.64952
adm_out[49] : pgas,co2 = partial pressure of carbon dioxide gas (bar, true value, i.e. not normalized) = 0.31212
adm_out[50] : pgas,total = total head space pressure of H2+CO2+CH4+H2O (bar, true value, i.e. not normalized) = 1.0173
adm_out[51] : qgas = gas flow rate normalized to atmospheric pressure (m3/d) = 43408.9902
Extra calculated outputs
Produced hydrogen gas (kg H2/d) = 0.055344
Produced methane gas (kg CH4/d) = 17532.603
Produced carbon dioxide gas (kg CO2/d) = 23168.6415
Energy content of methane gas (MJ/d) = 876875.6079
Energy content of methane gas (kWh/d) = 243576.5578

4.3 Case 2A

Combined input of primary and secondary sludge prior to ASM2ADM interface (identical to case 1A):

SI = 0 mg COD/l
SS = 0 mg COD/l
XI = 16988.959 mg COD/l
XS = 29946.898 mg COD/l
XBH = 9207.1504 mg COD/l
XBA = 0 mg COD/l
XP = 0 mg COD/l
SO = 0 mg -COD/l
SNO = 2.0925 mg N/l
SNH = 19.5373 mg N/l
SND = 0 mg N/l
XND = 1197.8759 mg N/l
SALK = 7 mol HCO3/m3
TSS = 39537.3291 mg SS/l
Flow rate = 1902 m3/d
Temperature = 15 degC

SI_load = 0 kg COD/d
SS_load = 0 kg COD/d
XI_load = 32313 kg COD/d
XS_load = 56959 kg COD/d
XBH_load = 17512 kg COD/d
XBA_load = 0 kg COD/d
XP_load = 0 kg COD/d
SO_load = 0 kg -COD/d
SNO_load = 3.98 kg N/d
SNH_load = 37.16 kg N/d
SND_load = 0 kg N/d
XND_load = 2278.36 kg N/d
SALK_load = 13.314 kmol HCO3/d
TSS_load = 75200 kg SS/d

Combined primary and secondary sludge after ASM2ADM interface (identical to case 1A)

adm_in[1] : Ssu = monosacharides (kg COD/m3) = 0
adm_in[2] : Saa = amino acids (kg COD/m3) = 0
adm_in[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0
adm_in[4] : Sva = total valerate (kg COD/m3) = 0
adm_in[5] : Sbu = total butyrate (kg COD/m3) = 0
adm_in[6] : Spro = total propionate (kg COD/m3) = 0
adm_in[7] : Sac = total acetate (kg COD/m3) = 0
adm_in[8] : Sh2 = hydrogen gas (kg COD/m3) = 0
adm_in[9] : Sch4 = methane gas (kg COD/m3) = 0
adm_in[10] : Sic = inorganic carbon (kmole C/m3) = 0.0082697

adm_in[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.0013955
adm_in[12] : Si = soluble inerts (kg COD/m3) = 0
adm_in[13] : Xc = composites (kg COD/m3) = 0
adm_in[14] : Xch = carbohydrates (kg COD/m3) = 5.6445
adm_in[15] : Xpr = proteins (kg COD/m3) = 17.9354
adm_in[16] : Xli = lipids (kg COD/m3) = 12.6219
adm_in[17] : Xsu = sugar degraders (kg COD/m3) = 0
adm_in[18] : Xaa = amino acid degraders (kg COD/m3) = 0
adm_in[19] : Xfa = LCFA degraders (kg COD/m3) = 0
adm_in[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0
adm_in[21] : Xpro = propionate degraders (kg COD/m3) = 0
adm_in[22] : Xac = acetate degraders (kg COD/m3) = 0
adm_in[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0
adm_in[24] : Xi = particulate inerts (kg COD/m3) = 19.9352
adm_in[25] : Scat+ = cations (base) (kmole/m3) = 0
adm_in[26] : San- = anions (acid) (kmole/m3) = 0.0057538
adm_in[27] : Flow rate (m3/d) = 1902
adm_in[28] : Temperature (degC) = 35

Combined primary and secondary sludge after ASM2ADM interface + external sludge input, i.e. total input to digester 1 (identical to case 1A)

adm_in[1] : Ssu = monosacharides (kg COD/m3) = 0
adm_in[2] : Saa = amino acids (kg COD/m3) = 0
adm_in[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0
adm_in[4] : Sva = total valerate (kg COD/m3) = 0
adm_in[5] : Sbu = total butyrate (kg COD/m3) = 0
adm_in[6] : Spro = total propionate (kg COD/m3) = 0
adm_in[7] : Sac = total acetate (kg COD/m3) = 0
adm_in[8] : Sh2 = hydrogen gas (kg COD/m3) = 0
adm_in[9] : Sch4 = methane gas (kg COD/m3) = 0
adm_in[10] : Sic = inorganic carbon (kmole C/m3) = 0.0073837
adm_in[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.0013011
adm_in[12] : Si = soluble inerts (kg COD/m3) = 0
adm_in[13] : Xc = composites (kg COD/m3) = 0
adm_in[14] : Xch = carbohydrates (kg COD/m3) = 5.2627
adm_in[15] : Xpr = proteins (kg COD/m3) = 16.7221
adm_in[16] : Xli = lipids (kg COD/m3) = 17.9645
adm_in[17] : Xsu = sugar degraders (kg COD/m3) = 0
adm_in[18] : Xaa = amino acid degraders (kg COD/m3) = 0
adm_in[19] : Xfa = LCFA degraders (kg COD/m3) = 0
adm_in[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0
adm_in[21] : Xpro = propionate degraders (kg COD/m3) = 0
adm_in[22] : Xac = acetate degraders (kg COD/m3) = 0
adm_in[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0
adm_in[24] : Xi = particulate inerts (kg COD/m3) = 18.864
adm_in[25] : Scat+ = cations (base) (kmole/m3) = 0
adm_in[26] : San- = anions (acid) (kmole/m3) = 0.0053644
adm_in[27] : Flow rate (m3/d) = 2040

adm_in[28] : Temperature (degC) = 35

Digester output from first reactor of 23700 m^3 (steady state), also input to second reactor (states 1 to 28)

adm_out[1] : Ssu = monosacharides (kg COD/m3) = 0.018349
adm_out[2] : Saa = amino acids (kg COD/m3) = 0.0081818
adm_out[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0.18564
adm_out[4] : Sva = total valerate (kg COD/m3) = 0.018575
adm_out[5] : Sbu = total butyrate (kg COD/m3) = 0.021642
adm_out[6] : Spro = total propionate (kg COD/m3) = 0.029195
adm_out[7] : Sac = total acetate (kg COD/m3) = 0.16892
adm_out[8] : Sh2 = hydrogen gas (kg COD/m3) = 3.7284e-07
adm_out[9] : Sch4 = methane gas (kg COD/m3) = 0.062079
adm_out[10] : Sic = inorganic carbon (kmole C/m3) = 0.088676
adm_out[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.089998 (= 1.26 kg N/m3)
adm_out[12] : Si = soluble inerts (kg COD/m3) = 0.08926
adm_out[13] : Xc = composites (kg COD/m3) = 0.15366
adm_out[14] : Xch = carbohydrates (kg COD/m3) = 0.046436
adm_out[15] : Xpr = proteins (kg COD/m3) = 0.14423
adm_out[16] : Xli = lipids (kg COD/m3) = 0.1556
adm_out[17] : Xsu = sugar degraders (kg COD/m3) = 0.50961
adm_out[18] : Xaa = amino acid degraders (kg COD/m3) = 1.0872
adm_out[19] : Xfa = LCFA degraders (kg COD/m3) = 0.82706
adm_out[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0.40141
adm_out[21] : Xpro = propionate degraders (kg COD/m3) = 0.13168
adm_out[22] : Xac = acetate degraders (kg COD/m3) = 1.0665
adm_out[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0.47945
adm_out[24] : Xi = particulate inerts (kg COD/m3) = 19.0426
adm_out[25] : Scat+ = cations (base) (kmole/m3) = 1.2862e-27
adm_out[26] : San- = anions (acid) (kmole/m3) = 0.0053644
adm_out[27] : Flow rate (m3/d) = 2040
adm_out[28] : Temperature (degC) = 35
adm_out[34] : pH = pH within AD system = 7.2574
adm_out[35] : S_H+ = protons (kmole/m3) = 5.5278e-08
adm_out[36] : Sva- = valerate (kg COD/m3) = 0.018501
adm_out[37] : Sbu- = butyrate (kg COD/m3) = 0.021564
adm_out[38] : Spro- = propionate (kg COD/m3) = 0.029073
adm_out[39] : Sac- = acetate (kg COD/m3) = 0.16838
adm_out[40] : Shco3- = bicarbonate (kmole C/m3) = 0.079747
adm_out[41] : Sco2 = carbon dioxide (kmole C/m3) = 0.0089289
adm_out[42] : Snh3 = ammonia (kmole N/m3) = 0.0017721
adm_out[43] : Snh4+ = ammonium (kmole N/m3) = 0.088226
adm_out[44] : Sgas,h2 = hydrogen concentration in gas phase (kg COD/m3) = 1.324e-05
adm_out[45] : Sgas,ch4 = methane concentration in gas phase (kg COD/m3) = 1.5914
adm_out[46] : Sgas,co2 = carbon dioxide concentration in gas phase (kmole C/m3) = 0.012669
adm_out[47] : pgas,h2 = partial pressure of hydrogen gas (bar, true value i.e. not normalized) = 2.1201e-05

adm_out[48] : pgas,ch4 = partial pressure of methane gas (bar, true value i.e. not normalized) = 0.63709
adm_out[49] : pgas,co2 = partial pressure of carbon dioxide gas (bar, true value, i.e. not normalized) = 0.3246
adm_out[50] : pgas,total = total head space pressure of H2+CO2+CH4+H2O (bar, true value, i.e. not normalized) = 1.0174
adm_out[51] : qgas = gas flow rate normalized to atmospheric pressure (m3/d) = 43985.7929
Extra calculated outputs
Produced hydrogen gas (kg H2/d) = 0.072482
Produced methane gas (kg CH4/d) = 17424.5625
Produced carbon dioxide gas (kg CO2/d) = 24414.1424
Energy content of methane gas (MJ/d) = 871472.0694
Energy content of methane gas (kWh/d) = 242075.5748

Digester output from second reactor of 15200 m^3 (steady state)

adm_out[1] : Ssu = monosacharides (kg COD/m3) = 0.00096133
adm_out[2] : Saa = amino acids (kg COD/m3) = 0.00022405
adm_out[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0.0060347
adm_out[4] : Sva = total valerate (kg COD/m3) = 0.0006193
adm_out[5] : Sbu = total butyrate (kg COD/m3) = 0.00074541
adm_out[6] : Spro = total propionate (kg COD/m3) = 0.0011143
adm_out[7] : Sac = total acetate (kg COD/m3) = 0.0048483
adm_out[8] : Sh2 = hydrogen gas (kg COD/m3) = 1.6896e-08
adm_out[9] : Sch4 = methane gas (kg COD/m3) = 0.048918
adm_out[10] : Sic = inorganic carbon (kmole C/m3) = 0.093738
adm_out[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.092873 (= 1.3002 kg N/m3)
adm_out[12] : Si = soluble inerts (kg COD/m3) = 0.1489
adm_out[13] : Xc = composites (kg COD/m3) = 0.16007
adm_out[14] : Xch = carbohydrates (kg COD/m3) = 0.0021945
adm_out[15] : Xpr = proteins (kg COD/m3) = 0.0034896
adm_out[16] : Xli = lipids (kg COD/m3) = 0.0044299
adm_out[17] : Xsu = sugar degraders (kg COD/m3) = 0.4607
adm_out[18] : Xaa = amino acid degraders (kg COD/m3) = 0.96489
adm_out[19] : Xfa = LCFA degraders (kg COD/m3) = 0.74555
adm_out[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0.35889
adm_out[21] : Xpro = propionate degraders (kg COD/m3) = 0.11899
adm_out[22] : Xac = acetate degraders (kg COD/m3) = 0.96437
adm_out[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0.4314
adm_out[24] : Xi = particulate inerts (kg COD/m3) = 19.1618
adm_out[25] : Scat+ = cations (base) (kmole/m3) = 5.667e-29
adm_out[26] : San- = anions (acid) (kmole/m3) = 0.0053644
adm_out[27] : Flow rate (m3/d) = 2040
adm_out[28] : Temperature (degC) = 35
adm_out[34] : pH = pH within AD system = 7.3135
adm_out[35] : S_H+ = protons (kmole/m3) = 4.8587e-08
adm_out[36] : Sva- = valerate (kg COD/m3) = 0.00061713
adm_out[37] : Sbu- = butyrate (kg COD/m3) = 0.00074302
adm_out[38] : Spro- = propionate (kg COD/m3) = 0.0011102

adm_out[39] : Sac- = acetate (kg COD/m3) = 0.0048348
adm_out[40] : Shco3- = bicarbonate (kmole C/m3) = 0.08534
adm_out[41] : Sco2 = carbon dioxide (kmole C/m3) = 0.0083985
adm_out[42] : Snh3 = ammonia (kmole N/m3) = 0.0020749
adm_out[43] : Snh4+ = ammonium (kmole N/m3) = 0.090798
adm_out[44] : Sgas,h2 = hydrogen concentration in gas phase (kg COD/m3) = 8.7276e-07
adm_out[45] : Sgas,ch4 = methane concentration in gas phase (kg COD/m3) = 1.6194
adm_out[46] : Sgas,co2 = carbon dioxide concentration in gas phase (kmole C/m3) = 0.012067
adm_out[47] : pgas,h2 = partial pressure of hydrogen gas (bar, true value i.e. not normalized) = 1.3976e-06
adm_out[48] : pgas,ch4 = partial pressure of methane gas (bar, true value i.e. not normalized) = 0.64829
adm_out[49] : pgas,co2 = partial pressure of carbon dioxide gas (bar, true value, i.e. not normalized) = 0.30918
adm_out[50] : pgas,total = total head space pressure of H2+CO2+CH4+H2O (bar, true value, i.e. not normalized) = 1.0131
adm_out[51] : qgas = gas flow rate normalized to atmospheric pressure (m3/d) = 1333.0827
Extra calculated outputs
Produced hydrogen gas (kg H2/d) = 0.00014541
Produced methane gas (kg CH4/d) = 539.62
Produced carbon dioxide gas (kg CO2/d) = 707.7235
Energy content of methane gas (MJ/d) = 26988.555
Energy content of methane gas (kWh/d) = 7496.8208

Second digester sludge output after ADM2ASM interface

SI = 148.8951 mg COD/l
SS = 14.5474 mg COD/l
XI = 19161.8294 mg COD/l
XS = 3365.5777 mg COD/l
XBH = 0 mg COD/l
XBA = 0 mg COD/l
XP = 849.408 mg COD/l
SO = 0 mg -COD/l
SNO = 0 mg N/l
SNH = 1461.6238 mg N/l
SND = 0.021957 mg N/l
XND = 126.5074 mg N/l
SALK = 99.0369 mol HCO3/m3
TSS = 16462.5458 mg SS/l
Flow rate = 2040 m3/d
Temperature = 15 degC

SI_load = 303.746 kg COD/d
SS_load = 29.6766 kg COD/d
XI_load = 39090.1319 kg COD/d
XS_load = 6865.7786 kg COD/d
XBH_load = 0 kg COD/d

XBA_load = 0 kg COD/d
XP_load = 1732.7922 kg COD/d
SO_load = 0 kg -COD/d
SNO_load = 0 kg N/d
SNH_load = 2981.7126 kg N/d
SND_load = 0.044792 kg N/d
XND_load = 258.0751 kg N/d
SALK_load = 202.0353 kmol HCO3/d
TSS_load = 33583.5935 kg SS/d

4.4 Case 2B

Total input to digester 1 (identical to case 1B)

adm_in[1] : Ssu = monosacharides (kg COD/m3) = 0
adm_in[2] : Saa = amino acids (kg COD/m3) = 0
adm_in[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0
adm_in[4] : Sva = total valerate (kg COD/m3) = 0
adm_in[5] : Sbu = total butyrate (kg COD/m3) = 0
adm_in[6] : Spro = total propionate (kg COD/m3) = 0
adm_in[7] : Sac = total acetate (kg COD/m3) = 0
adm_in[8] : Sh2 = hydrogen gas (kg COD/m3) = 0
adm_in[9] : Sch4 = methane gas (kg COD/m3) = 0
adm_in[10] : Sic = inorganic carbon (kmole C/m3) = 0
adm_in[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.0013011
adm_in[12] : Si = soluble inerts (kg COD/m3) = 0
adm_in[13] : Xc = composites (kg COD/m3) = 52.3451
adm_in[14] : Xch = carbohydrates (kg COD/m3) = 0
adm_in[15] : Xpr = proteins (kg COD/m3) = 0
adm_in[16] : Xli = lipids (kg COD/m3) = 6.1965
adm_in[17] : Xsu = sugar degraders (kg COD/m3) = 0
adm_in[18] : Xaa = amino acid degraders (kg COD/m3) = 0
adm_in[19] : Xfa = LCFA degraders (kg COD/m3) = 0
adm_in[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0
adm_in[21] : Xpro = propionate degraders (kg COD/m3) = 0
adm_in[22] : Xac = acetate degraders (kg COD/m3) = 0
adm_in[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0
adm_in[24] : Xi = particulate inerts (kg COD/m3) = 0.2774
adm_in[25] : Scat+ = cations (base) (kmole/m3) = 0.04
adm_in[26] : San- = anions (acid) (kmole/m3) = 0.0053645
adm_in[27] : Flow rate (m3/d) = 2040
adm_in[28] : Temperature (degC) = 35

Digester output from first digester of 23700 m^3 (steady state), also input to digester 2 (states 1 to 28)

adm_out[1] : Ssu = monosacharides (kg COD/m3) = 0.018377
adm_out[2] : Saa = amino acids (kg COD/m3) = 0.0081945
adm_out[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0.18608
adm_out[4] : Sva = total valerate (kg COD/m3) = 0.016989
adm_out[5] : Sbu = total butyrate (kg COD/m3) = 0.02275
adm_out[6] : Spro = total propionate (kg COD/m3) = 0.029258
adm_out[7] : Sac = total acetate (kg COD/m3) = 0.088602
adm_out[8] : Sh2 = hydrogen gas (kg COD/m3) = 3.7343e-07
adm_out[9] : Sch4 = methane gas (kg COD/m3) = 0.06235
adm_out[10] : Sic = inorganic carbon (kmole C/m3) = 0.078747
adm_out[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.038177 (= 0.53448 kg N/m3)
adm_out[12] : Si = soluble inerts (kg COD/m3) = 4.5476
adm_out[13] : Xc = composites (kg COD/m3) = 7.8288

adm_out[14] : Xch = carbohydrates (kg COD/m3) = 0.07762
adm_out[15] : Xpr = proteins (kg COD/m3) = 0.07762
adm_out[16] : Xli = lipids (kg COD/m3) = 0.16931
adm_out[17] : Xsu = sugar degraders (kg COD/m3) = 0.81006
adm_out[18] : Xaa = amino acid degraders (kg COD/m3) = 0.58486
adm_out[19] : Xfa = LCFA degraders (kg COD/m3) = 0.90074
adm_out[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0.25267
adm_out[21] : Xpro = propionate degraders (kg COD/m3) = 0.12237
adm_out[22] : Xac = acetate degraders (kg COD/m3) = 0.98102
adm_out[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0.48027
adm_out[24] : Xi = particulate inerts (kg COD/m3) = 9.3727
adm_out[25] : Scat+ = cations (base) (kmole/m3) = 0.04
adm_out[26] : San- = anions (acid) (kmole/m3) = 0.0053645
adm_out[27] : Flow rate (m3/d) = 2040
adm_out[28] : Temperature (degC) = 35
adm_out[34] : pH = pH within AD system = 7.2241
adm_out[35] : S_H+ = protons (kmole/m3) = 5.9694e-08
adm_out[36] : Sva- = valerate (kg COD/m3) = 0.016916
adm_out[37] : Sbu- = butyrate (kg COD/m3) = 0.02266
adm_out[38] : Spro- = propionate (kg COD/m3) = 0.029126
adm_out[39] : Sac- = acetate (kg COD/m3) = 0.088299
adm_out[40] : Shco3- = bicarbonate (kmole C/m3) = 0.070253
adm_out[41] : Sco2 = carbon dioxide (kmole C/m3) = 0.0084943
adm_out[42] : Snh3 = ammonia (kmole N/m3) = 0.00069712
adm_out[43] : Snh4+ = ammonium (kmole N/m3) = 0.03748
adm_out[44] : Sgas,h2 = hydrogen concentration in gas phase (kg COD/m3) = 1.3618e-05
adm_out[45] : Sgas,ch4 = methane concentration in gas phase (kg COD/m3) = 1.6292
adm_out[46] : Sgas,co2 = carbon dioxide concentration in gas phase (kmole C/m3) = 0.012065
adm_out[47] : pgas,h2 = partial pressure of hydrogen gas (bar, true value i.e. not normalized) = 2.1807e-05
adm_out[48] : pgas,ch4 = partial pressure of methane gas (bar, true value i.e. not normalized) = 0.65222
adm_out[49] : pgas,co2 = partial pressure of carbon dioxide gas (bar, true value, i.e. not normalized) = 0.30912
adm_out[50] : pgas,total = total head space pressure of H2+CO2+CH4+H2O (bar, true value, i.e. not normalized) = 1.017
adm_out[51] : qgas = gas flow rate normalized to atmospheric pressure (m3/d) = 40455.154
Extra calculated outputs
Produced hydrogen gas (kg H2/d) = 0.068593
Produced methane gas (kg CH4/d) = 16412.0277
Produced carbon dioxide gas (kg CO2/d) = 21391.3456
Energy content of methane gas (MJ/d) = 820831.1555
Energy content of methane gas (kWh/d) = 228008.6543

Digester output from second digester of 15200 m^3 (steady state)

adm_out[1] : Ssu = monosacharides (kg COD/m3) = 0.0040892
adm_out[2] : Saa = amino acids (kg COD/m3) = 0.0018734

adm_out[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0.023559
adm_out[4] : Sva = total valerate (kg COD/m3) = 0.0037442
adm_out[5] : Sbu = total butyrate (kg COD/m3) = 0.0049137
adm_out[6] : Spro = total propionate (kg COD/m3) = 0.0053395
adm_out[7] : Sac = total acetate (kg COD/m3) = 0.013943
adm_out[8] : Sh2 = hydrogen gas (kg COD/m3) = 7.5231e-08
adm_out[9] : Sch4 = methane gas (kg COD/m3) = 0.048703
adm_out[10] : Sic = inorganic carbon (kmole C/m3) = 0.08858
adm_out[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.045751 (= 0.64052 kg N/m3)
adm_out[12] : Si = soluble inerts (kg COD/m3) = 5.2145
adm_out[13] : Xc = composites (kg COD/m3) = 1.7899
adm_out[14] : Xch = carbohydrates (kg COD/m3) = 0.01869
adm_out[15] : Xpr = proteins (kg COD/m3) = 0.01869
adm_out[16] : Xli = lipids (kg COD/m3) = 0.028736
adm_out[17] : Xsu = sugar degraders (kg COD/m3) = 0.83676
adm_out[18] : Xaa = amino acid degraders (kg COD/m3) = 0.60641
adm_out[19] : Xfa = LCFA degraders (kg COD/m3) = 0.89862
adm_out[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0.26371
adm_out[21] : Xpro = propionate degraders (kg COD/m3) = 0.12784
adm_out[22] : Xac = acetate degraders (kg COD/m3) = 1.0025
adm_out[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0.48857
adm_out[24] : Xi = particulate inerts (kg COD/m3) = 10.7064
adm_out[25] : Scat+ = cations (base) (kmole/m3) = 0.04
adm_out[26] : San- = anions (acid) (kmole/m3) = 0.0053645
adm_out[27] : Flow rate (m3/d) = 2040
adm_out[28] : Temperature (degC) = 35
adm_out[34] : pH = pH within AD system = 7.234
adm_out[35] : S_H+ = protons (kmole/m3) = 5.8343e-08
adm_out[36] : Sva- = valerate (kg COD/m3) = 0.0037284
adm_out[37] : Sbu- = butyrate (kg COD/m3) = 0.0048948
adm_out[38] : Spro- = propionate (kg COD/m3) = 0.005316
adm_out[39] : Sac- = acetate (kg COD/m3) = 0.013896
adm_out[40] : Shco3- = bicarbonate (kmole C/m3) = 0.079219
adm_out[41] : Sco2 = carbon dioxide (kmole C/m3) = 0.0093616
adm_out[42] : Snh3 = ammonia (kmole N/m3) = 0.0008544
adm_out[43] : Snh4+ = ammonium (kmole N/m3) = 0.044897
adm_out[44] : Sgas,h2 = hydrogen concentration in gas phase (kg COD/m3) = 3.5991e-06
adm_out[45] : Sgas,ch4 = methane concentration in gas phase (kg COD/m3) = 1.5339
adm_out[46] : Sgas,co2 = carbon dioxide concentration in gas phase (kmole C/m3) = 0.013421
adm_out[47] : pgas,h2 = partial pressure of hydrogen gas (bar, true value i.e. not normalized) = 5.7633e-06
adm_out[48] : pgas,ch4 = partial pressure of methane gas (bar, true value i.e. not normalized) = 0.61406
adm_out[49] : pgas,co2 = partial pressure of carbon dioxide gas (bar, true value, i.e. not normalized) = 0.34387
adm_out[50] : pgas,total = total head space pressure of H2+CO2+CH4+H2O (bar, true value, i.e. not normalized) = 1.0136
adm_out[51] : qgas = gas flow rate normalized to atmospheric pressure (m3/d) = 6029.388

Extra calculated outputs
Produced hydrogen gas (kg H2/d) = 0.0027109
Produced methane gas (kg CH4/d) = 2310.7144
Produced carbon dioxide gas (kg CO2/d) = 3558.4801
Energy content of methane gas (MJ/d) = 115568.0721
Energy content of methane gas (kWh/d) = 32102.2423

4.5 Case 3A

Combined input of primary and secondary sludge prior to ASM2ADM interface:

SI = 0 mg COD/l
SS = 0 mg COD/l
XI = 17007.5188 mg COD/l
XS = 30106.203 mg COD/l
XBH = 9139.0977 mg COD/l
XBA = 0 mg COD/l
XP = 0 mg COD/l
SO = 0 mg -COD/l
SNO = 2.0771 mg N/l
SNH = 19.6147 mg N/l
SND = 0 mg N/l
XND = 1204.2481 mg N/l
SALK = 7 mol HCO3/m3
TSS = 39614.6617 mg SS/l
Flow rate = 2128 m3/d
Temperature = 15 degC

SI_load = 0 kg COD/d
SS_load = 0 kg COD/d
XI_load = 36192 kg COD/d
XS_load = 64066 kg COD/d
XBH_load = 19448 kg COD/d
XBA_load = 0 kg COD/d
XP_load = 0 kg COD/d
SO_load = 0 kg -COD/d
SNO_load = 4.42 kg N/d
SNH_load = 41.74 kg N/d
SND_load = 0 kg N/d
XND_load = 2562.64 kg N/d
SALK_load = 14.896 kmol HCO3/d
TSS_load = 84300 kg SS/d

Combined primary and secondary sludge after ASM2ADM interface

adm_in[1] : Ssu = monosacharides (kg COD/m3) = 0
adm_in[2] : Saa = amino acids (kg COD/m3) = 0
adm_in[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0
adm_in[4] : Sva = total valerate (kg COD/m3) = 0
adm_in[5] : Sbu = total butyrate (kg COD/m3) = 0
adm_in[6] : Spro = total propionate (kg COD/m3) = 0
adm_in[7] : Sac = total acetate (kg COD/m3) = 0
adm_in[8] : Sh2 = hydrogen gas (kg COD/m3) = 0
adm_in[9] : Sch4 = methane gas (kg COD/m3) = 0
adm_in[10] : Sic = inorganic carbon (kmole C/m3) = 0.0079996
adm_in[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.001401

adm_in[12] : Si = soluble inerts (kg COD/m3) = 0
adm_in[13] : Xc = composites (kg COD/m3) = 0
adm_in[14] : Xch = carbohydrates (kg COD/m3) = 5.6704
adm_in[15] : Xpr = proteins (kg COD/m3) = 17.9582
adm_in[16] : Xli = lipids (kg COD/m3) = 12.6863
adm_in[17] : Xsu = sugar degraders (kg COD/m3) = 0
adm_in[18] : Xaa = amino acid degraders (kg COD/m3) = 0
adm_in[19] : Xfa = LCFA degraders (kg COD/m3) = 0
adm_in[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0
adm_in[21] : Xpro = propionate degraders (kg COD/m3) = 0
adm_in[22] : Xac = acetate degraders (kg COD/m3) = 0
adm_in[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0
adm_in[24] : Xi = particulate inerts (kg COD/m3) = 19.932
adm_in[25] : Scat+ = cations (base) (kmole/m3) = 0
adm_in[26] : San- = anions (acid) (kmole/m3) = 0.005747
adm_in[27] : Flow rate (m3/d) = 2128
adm_in[28] : Temperature (degC) = 35

Combined primary and secondary sludge after ASM2ADM interface + external sludge input, i.e. total input to digester

adm_in[1] : Ssu = monosacharides (kg COD/m3) = 0
adm_in[2] : Saa = amino acids (kg COD/m3) = 0
adm_in[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0
adm_in[4] : Sva = total valerate (kg COD/m3) = 0
adm_in[5] : Sbu = total butyrate (kg COD/m3) = 0
adm_in[6] : Spro = total propionate (kg COD/m3) = 0
adm_in[7] : Sac = total acetate (kg COD/m3) = 0
adm_in[8] : Sh2 = hydrogen gas (kg COD/m3) = 0
adm_in[9] : Sch4 = methane gas (kg COD/m3) = 0
adm_in[10] : Sic = inorganic carbon (kmole C/m3) = 0.0067381
adm_in[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.0011719
adm_in[12] : Si = soluble inerts (kg COD/m3) = 0
adm_in[13] : Xc = composites (kg COD/m3) = 0
adm_in[14] : Xch = carbohydrates (kg COD/m3) = 4.7431
adm_in[15] : Xpr = proteins (kg COD/m3) = 15.0217
adm_in[16] : Xli = lipids (kg COD/m3) = 33.93
adm_in[17] : Xsu = sugar degraders (kg COD/m3) = 0
adm_in[18] : Xaa = amino acid degraders (kg COD/m3) = 0
adm_in[19] : Xfa = LCFA degraders (kg COD/m3) = 0
adm_in[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0
adm_in[21] : Xpro = propionate degraders (kg COD/m3) = 0
adm_in[22] : Xac = acetate degraders (kg COD/m3) = 0
adm_in[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0
adm_in[24] : Xi = particulate inerts (kg COD/m3) = 17.6211
adm_in[25] : Scat+ = cations (base) (kmole/m3) = 0
adm_in[26] : San- = anions (acid) (kmole/m3) = 0.0048073
adm_in[27] : Flow rate (m3/d) = 2544
adm_in[28] : Temperature (degC) = 35

Digester output (steady state)

adm_out[1] : Ssu = monosacharides (kg COD/m3) = 0.014671
adm_out[2] : Saa = amino acids (kg COD/m3) = 0.0065539
adm_out[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0.13452
adm_out[4] : Sva = total valerate (kg COD/m3) = 0.014443
adm_out[5] : Sbu = total butyrate (kg COD/m3) = 0.017037
adm_out[6] : Spro = total propionate (kg COD/m3) = 0.021717
adm_out[7] : Sac = total acetate (kg COD/m3) = 0.085172
adm_out[8] : Sh2 = hydrogen gas (kg COD/m3) = 2.9716e-07
adm_out[9] : Sch4 = methane gas (kg COD/m3) = 0.064281
adm_out[10] : Sic = inorganic carbon (kmole C/m3) = 0.073004
adm_out[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.072295 (= 1.0121 kg N/m3)
adm_out[12] : Si = soluble inerts (kg COD/m3) = 0.14621
adm_out[13] : Xc = composites (kg COD/m3) = 0.19124
adm_out[14] : Xch = carbohydrates (kg COD/m3) = 0.032718
adm_out[15] : Xpr = proteins (kg COD/m3) = 0.099501
adm_out[16] : Xli = lipids (kg COD/m3) = 0.22331
adm_out[17] : Xsu = sugar degraders (kg COD/m3) = 0.51274
adm_out[18] : Xaa = amino acid degraders (kg COD/m3) = 0.93171
adm_out[19] : Xfa = LCFA degraders (kg COD/m3) = 1.4843
adm_out[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0.34956
adm_out[21] : Xpro = propionate degraders (kg COD/m3) = 0.12041
adm_out[22] : Xac = acetate degraders (kg COD/m3) = 1.3644
adm_out[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0.6431
adm_out[24] : Xi = particulate inerts (kg COD/m3) = 17.9135
adm_out[25] : Scat+ = cations (base) (kmole/m3) = -2.903e-29
adm_out[26] : San- = anions (acid) (kmole/m3) = 0.0048073
adm_out[27] : Flow rate (m3/d) = 2544
adm_out[28] : Temperature (degC) = 35
adm_out[34] : pH = pH within AD system = 7.1905
adm_out[35] : S_H+ = protons (kmole/m3) = 6.4489e-08
adm_out[36] : Sva- = valerate (kg COD/m3) = 0.014376
adm_out[37] : Sbu- = butyrate (kg COD/m3) = 0.016965
adm_out[38] : Spro- = propionate (kg COD/m3) = 0.021611
adm_out[39] : Sac- = acetate (kg COD/m3) = 0.084857
adm_out[40] : Shco3- = bicarbonate (kmole C/m3) = 0.06457
adm_out[41] : Sco2 = carbon dioxide (kmole C/m3) = 0.0084342
adm_out[42] : Snh3 = ammonia (kmole N/m3) = 0.0012236
adm_out[43] : Snh4+ = ammonium (kmole N/m3) = 0.071071
adm_out[44] : Sgas,h2 = hydrogen concentration in gas phase (kg COD/m3) = 1.0515e-05
adm_out[45] : Sgas,ch4 = methane concentration in gas phase (kg COD/m3) = 1.6437
adm_out[46] : Sgas,co2 = carbon dioxide concentration in gas phase (kmole C/m3) = 0.011966
adm_out[47] : pgas,h2 = partial pressure of hydrogen gas (bar, true value i.e. not normalized) = 1.6838e-05
adm_out[48] : pgas,ch4 = partial pressure of methane gas (bar, true value i.e. not normalized) = 0.65801

adm_out[49] : pgas,co2 = partial pressure of carbon dioxide gas (bar, true value, i.e. not normalized) = 0.30657
adm_out[50] : pgas,total = total head space pressure of H2+CO2+CH4+H2O (bar, true value, i.e. not normalized) = 1.0203
adm_out[51] : qgas = gas flow rate normalized to atmospheric pressure (m3/d) = 73179.9997
Extra calculated outputs
Produced hydrogen gas (kg H2/d) = 0.095504
Produced methane gas (kg CH4/d) = 29856.693
Produced carbon dioxide gas (kg CO2/d) = 38253.9585
Energy content of methane gas (MJ/d) = 1493252.6434
Energy content of methane gas (kWh/d) = 414792.401

Digester sludge output after ADM2ASM interface

SI = 146.2088 mg COD/l
SS = 294.1161 mg COD/l
XI = 17913.5497 mg COD/l
XS = 4817.6942 mg COD/l
XBH = 0 mg COD/l
XBA = 0 mg COD/l
XP = 1135.3114 mg COD/l
SO = 0 mg -COD/l
SNO = 0 mg N/l
SNH = 1224.6945 mg N/l
SND = 0.64229 mg N/l
XND = 177.5287 mg N/l
SALK = 82.6706 mol HCO3/m3
TSS = 16807.4333 mg SS/l
Flow rate = 2544 m3/d
Temperature = 15 degC

SI_load = 371.9553 kg COD/d
SS_load = 748.2313 kg COD/d
XI_load = 45572.0705 kg COD/d
XS_load = 12256.214 kg COD/d
XBH_load = 0 kg COD/d
XBA_load = 0 kg COD/d
XP_load = 2888.2323 kg COD/d
SO_load = 0 kg -COD/d
SNO_load = 0 kg N/d
SNH_load = 3115.6229 kg N/d
SND_load = 1.634 kg N/d
XND_load = 451.633 kg N/d
SALK_load = 210.3141 kmol HCO3/d
TSS_load = 42758.1104 kg SS/d

4.6 Case 3B

Total input to digester

adm_in[1] : Ssu = monosacharides (kg COD/m3) = 0
adm_in[2] : Saa = amino acids (kg COD/m3) = 0
adm_in[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0
adm_in[4] : Sva = total valerate (kg COD/m3) = 0
adm_in[5] : Sbu = total butyrate (kg COD/m3) = 0
adm_in[6] : Spro = total propionate (kg COD/m3) = 0
adm_in[7] : Sac = total acetate (kg COD/m3) = 0
adm_in[8] : Sh2 = hydrogen gas (kg COD/m3) = 0
adm_in[9] : Sch4 = methane gas (kg COD/m3) = 0
adm_in[10] : Sic = inorganic carbon (kmole C/m3) = 0
adm_in[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.0011719
adm_in[12] : Si = soluble inerts (kg COD/m3) = 0
adm_in[13] : Xc = composites (kg COD/m3) = 47.0542
adm_in[14] : Xch = carbohydrates (kg COD/m3) = 0
adm_in[15] : Xpr = proteins (kg COD/m3) = 0
adm_in[16] : Xli = lipids (kg COD/m3) = 23.3182
adm_in[17] : Xsu = sugar degraders (kg COD/m3) = 0
adm_in[18] : Xaa = amino acid degraders (kg COD/m3) = 0
adm_in[19] : Xfa = LCFA degraders (kg COD/m3) = 0
adm_in[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0
adm_in[21] : Xpro = propionate degraders (kg COD/m3) = 0
adm_in[22] : Xac = acetate degraders (kg COD/m3) = 0
adm_in[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0
adm_in[24] : Xi = particulate inerts (kg COD/m3) = 0.9484
adm_in[25] : Scat+ = cations (base) (kmole/m3) = 0.04
adm_in[26] : San- = anions (acid) (kmole/m3) = 0.0048073
adm_in[27] : Flow rate (m3/d) = 2544
adm_in[28] : Temperature (degC) = 35

Digester output (steady state)

adm_out[1] : Ssu = monosacharides (kg COD/m3) = 0.014706
adm_out[2] : Saa = amino acids (kg COD/m3) = 0.0065695
adm_out[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0.13496
adm_out[4] : Sva = total valerate (kg COD/m3) = 0.013191
adm_out[5] : Sbu = total butyrate (kg COD/m3) = 0.017905
adm_out[6] : Spro = total propionate (kg COD/m3) = 0.021783
adm_out[7] : Sac = total acetate (kg COD/m3) = 0.055445
adm_out[8] : Sh2 = hydrogen gas (kg COD/m3) = 2.9788e-07
adm_out[9] : Sch4 = methane gas (kg COD/m3) = 0.064858
adm_out[10] : Sic = inorganic carbon (kmole C/m3) = 0.0686
adm_out[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.026994 (= 0.37792 kg N/m3)
adm_out[12] : Si = soluble inerts (kg COD/m3) = 4.3017
adm_out[13] : Xc = composites (kg COD/m3) = 5.6265
adm_out[14] : Xch = carbohydrates (kg COD/m3) = 0.055899

adm_out[15] : Xpr = proteins (kg COD/m3) = 0.055899
adm_out[16] : Xli = lipids (kg COD/m3) = 0.23536
adm_out[17] : Xsu = sugar degraders (kg COD/m3) = 0.79124
adm_out[18] : Xaa = amino acid degraders (kg COD/m3) = 0.52325
adm_out[19] : Xfa = LCFA degraders (kg COD/m3) = 1.5647
adm_out[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0.23103
adm_out[21] : Xpro = propionate degraders (kg COD/m3) = 0.11617
adm_out[22] : Xac = acetate degraders (kg COD/m3) = 1.3148
adm_out[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0.65557
adm_out[24] : Xi = particulate inerts (kg COD/m3) = 9.5518
adm_out[25] : Scat+ = cations (base) (kmole/m3) = 0.04
adm_out[26] : San- = anions (acid) (kmole/m3) = 0.0048073
adm_out[27] : Flow rate (m3/d) = 2544
adm_out[28] : Temperature (degC) = 35
adm_out[34] : pH = pH within AD system = 7.1804
adm_out[35] : S_H+ = protons (kmole/m3) = 6.6014e-08
adm_out[36] : Sva- = valerate (kg COD/m3) = 0.013128
adm_out[37] : Sbu- = butyrate (kg COD/m3) = 0.017827
adm_out[38] : Spro- = propionate (kg COD/m3) = 0.021674
adm_out[39] : Sac- = acetate (kg COD/m3) = 0.055235
adm_out[40] : Shco3- = bicarbonate (kmole C/m3) = 0.060509
adm_out[41] : Sco2 = carbon dioxide (kmole C/m3) = 0.0080907
adm_out[42] : Snh3 = ammonia (kmole N/m3) = 0.00044651
adm_out[43] : Snh4+ = ammonium (kmole N/m3) = 0.026548
adm_out[44] : Sgas,h2 = hydrogen concentration in gas phase (kg COD/m3) = 1.0676e-05
adm_out[45] : Sgas,ch4 = methane concentration in gas phase (kg COD/m3) = 1.6738
adm_out[46] : Sgas,co2 = carbon dioxide concentration in gas phase (kmole C/m3) = 0.011484
adm_out[47] : pgas,h2 = partial pressure of hydrogen gas (bar, true value i.e. not normalized) = 1.7096e-05
adm_out[48] : pgas,ch4 = partial pressure of methane gas (bar, true value i.e. not normalized) = 0.67006
adm_out[49] : pgas,co2 = partial pressure of carbon dioxide gas (bar, true value, i.e. not normalized) = 0.29424
adm_out[50] : pgas,total = total head space pressure of H2+CO2+CH4+H2O (bar, true value, i.e. not normalized) = 1.02
adm_out[51] : qgas = gas flow rate normalized to atmospheric pressure (m3/d) = 70348.0586
Extra calculated outputs
Produced hydrogen gas (kg H2/d) = 0.09324
Produced methane gas (kg CH4/d) = 29235.1387
Produced carbon dioxide gas (kg CO2/d) = 35303.6517
Energy content of methane gas (MJ/d) = 1462166.2279
Energy content of methane gas (kWh/d) = 406157.2855

4.7 Case 4A

Combined input of primary and secondary sludge prior to ASM2ADM interface (identical to case 3A):

SI = 0 mg COD/l
SS = 0 mg COD/l
XI = 17007.5188 mg COD/l
XS = 30106.203 mg COD/l
XBH = 9139.0977 mg COD/l
XBA = 0 mg COD/l
XP = 0 mg COD/l
SO = 0 mg -COD/l
SNO = 2.0771 mg N/l
SNH = 19.6147 mg N/l
SND = 0 mg N/l
XND = 1204.2481 mg N/l
SALK = 7 mol HCO3/m3
TSS = 39614.6617 mg SS/l
Flow rate = 2128 m3/d
Temperature = 15 degC

SI_load = 0 kg COD/d
SS_load = 0 kg COD/d
XI_load = 36192 kg COD/d
XS_load = 64066 kg COD/d
XBH_load = 19448 kg COD/d
XBA_load = 0 kg COD/d
XP_load = 0 kg COD/d
SO_load = 0 kg -COD/d
SNO_load = 4.42 kg N/d
SNH_load = 41.74 kg N/d
SND_load = 0 kg N/d
XND_load = 2562.64 kg N/d
SALK_load = 14.896 kmol HCO3/d
TSS_load = 84300 kg SS/d

Combined primary and secondary sludge after ASM2ADM interface (identical to case 3A)

adm_in[1] : Ssu = monosacharides (kg COD/m3) = 0
adm_in[2] : Saa = amino acids (kg COD/m3) = 0
adm_in[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0
adm_in[4] : Sva = total valerate (kg COD/m3) = 0
adm_in[5] : Sbu = total butyrate (kg COD/m3) = 0
adm_in[6] : Spro = total propionate (kg COD/m3) = 0
adm_in[7] : Sac = total acetate (kg COD/m3) = 0
adm_in[8] : Sh2 = hydrogen gas (kg COD/m3) = 0
adm_in[9] : Sch4 = methane gas (kg COD/m3) = 0
adm_in[10] : Sic = inorganic carbon (kmole C/m3) = 0.0079996

adm_in[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.001401
adm_in[12] : Si = soluble inerts (kg COD/m3) = 0
adm_in[13] : Xc = composites (kg COD/m3) = 0
adm_in[14] : Xch = carbohydrates (kg COD/m3) = 5.6704
adm_in[15] : Xpr = proteins (kg COD/m3) = 17.9582
adm_in[16] : Xli = lipids (kg COD/m3) = 12.6863
adm_in[17] : Xsu = sugar degraders (kg COD/m3) = 0
adm_in[18] : Xaa = amino acid degraders (kg COD/m3) = 0
adm_in[19] : Xfa = LCFA degraders (kg COD/m3) = 0
adm_in[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0
adm_in[21] : Xpro = propionate degraders (kg COD/m3) = 0
adm_in[22] : Xac = acetate degraders (kg COD/m3) = 0
adm_in[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0
adm_in[24] : Xi = particulate inerts (kg COD/m3) = 19.932
adm_in[25] : Scat+ = cations (base) (kmole/m3) = 0
adm_in[26] : San- = anions (acid) (kmole/m3) = 0.005747
adm_in[27] : Flow rate (m3/d) = 2128
adm_in[28] : Temperature (degC) = 35

Combined primary and secondary sludge after ASM2ADM interface + external sludge input, i.e. total input to first digester (identical to case 3A)

adm_in[1] : Ssu = monosacharides (kg COD/m3) = 0
adm_in[2] : Saa = amino acids (kg COD/m3) = 0
adm_in[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0
adm_in[4] : Sva = total valerate (kg COD/m3) = 0
adm_in[5] : Sbu = total butyrate (kg COD/m3) = 0
adm_in[6] : Spro = total propionate (kg COD/m3) = 0
adm_in[7] : Sac = total acetate (kg COD/m3) = 0
adm_in[8] : Sh2 = hydrogen gas (kg COD/m3) = 0
adm_in[9] : Sch4 = methane gas (kg COD/m3) = 0
adm_in[10] : Sic = inorganic carbon (kmole C/m3) = 0.006793
adm_in[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.0011719
adm_in[12] : Si = soluble inerts (kg COD/m3) = 0
adm_in[13] : Xc = composites (kg COD/m3) = 0
adm_in[14] : Xch = carbohydrates (kg COD/m3) = 4.7431
adm_in[15] : Xpr = proteins (kg COD/m3) = 15.0217
adm_in[16] : Xli = lipids (kg COD/m3) = 33.93
adm_in[17] : Xsu = sugar degraders (kg COD/m3) = 0
adm_in[18] : Xaa = amino acid degraders (kg COD/m3) = 0
adm_in[19] : Xfa = LCFA degraders (kg COD/m3) = 0
adm_in[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0
adm_in[21] : Xpro = propionate degraders (kg COD/m3) = 0
adm_in[22] : Xac = acetate degraders (kg COD/m3) = 0
adm_in[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0
adm_in[24] : Xi = particulate inerts (kg COD/m3) = 17.6211
adm_in[25] : Scat+ = cations (base) (kmole/m3) = 0
adm_in[26] : San- = anions (acid) (kmole/m3) = 0.0048073
adm_in[27] : Flow rate (m3/d) = 2544

adm_in[28] : Temperature (degC) = 35

Digester output from first reactor of 23700 m^3 (steady state), also input to second reactor (states 1 to 28)

adm_out[1] : Ssu = monosacharides (kg COD/m3) = 0.022198
adm_out[2] : Saa = amino acids (kg COD/m3) = 0.0098794
adm_out[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0.25179
adm_out[4] : Sva = total valerate (kg COD/m3) = 0.022763
adm_out[5] : Sbu = total butyrate (kg COD/m3) = 0.026914
adm_out[6] : Spro = total propionate (kg COD/m3) = 0.038304
adm_out[7] : Sac = total acetate (kg COD/m3) = 0.16288
adm_out[8] : Sh2 = hydrogen gas (kg COD/m3) = 4.5257e-07
adm_out[9] : Sch4 = methane gas (kg COD/m3) = 0.073716
adm_out[10] : Sic = inorganic carbon (kmole C/m3) = 0.06939
adm_out[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.069958 (= 0.97941 kg N/m3)
adm_out[12] : Si = soluble inerts (kg COD/m3) = 0.089903
adm_out[13] : Xc = composites (kg COD/m3) = 0.19301
adm_out[14] : Xch = carbohydrates (kg COD/m3) = 0.052283
adm_out[15] : Xpr = proteins (kg COD/m3) = 0.16144
adm_out[16] : Xli = lipids (kg COD/m3) = 0.36321
adm_out[17] : Xsu = sugar degraders (kg COD/m3) = 0.55131
adm_out[18] : Xaa = amino acid degraders (kg COD/m3) = 1.0136
adm_out[19] : Xfa = LCFA degraders (kg COD/m3) = 1.613
adm_out[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0.37888
adm_out[21] : Xpro = propionate degraders (kg COD/m3) = 0.12965
adm_out[22] : Xac = acetate degraders (kg COD/m3) = 1.4773
adm_out[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0.69738
adm_out[24] : Xi = particulate inerts (kg COD/m3) = 17.8009
adm_out[25] : Scat+ = cations (base) (kmole/m3) = 4.3611e-24
adm_out[26] : San- = anions (acid) (kmole/m3) = 0.0048073
adm_out[27] : Flow rate (m3/d) = 2544
adm_out[28] : Temperature (degC) = 35
adm_out[34] : pH = pH within AD system = 7.1616
adm_out[35] : S_H+ = protons (kmole/m3) = 6.8923e-08
adm_out[36] : Sva- = valerate (kg COD/m3) = 0.02265
adm_out[37] : Sbu- = butyrate (kg COD/m3) = 0.026792
adm_out[38] : Spro- = propionate (kg COD/m3) = 0.038104
adm_out[39] : Sac- = acetate (kg COD/m3) = 0.16224
adm_out[40] : Shco3- = bicarbonate (kmole C/m3) = 0.06089
adm_out[41] : Sco2 = carbon dioxide (kmole C/m3) = 0.0085005
adm_out[42] : Snh3 = ammonia (kmole N/m3) = 0.0011091
adm_out[43] : Snh4+ = ammonium (kmole N/m3) = 0.068849
adm_out[44] : Sgas,h2 = hydrogen concentration in gas phase (kg COD/m3) = 1.331e-05
adm_out[45] : Sgas,ch4 = methane concentration in gas phase (kg COD/m3) = 1.6436
adm_out[46] : Sgas,co2 = carbon dioxide concentration in gas phase (kmole C/m3) = 0.011962
adm_out[47] : pgas,h2 = partial pressure of hydrogen gas (bar, true value i.e. not normalized) = 2.1314e-05

adm_out[48] : pgas,ch4 = partial pressure of methane gas (bar, true value i.e. not normalized) = 0.65797
adm_out[49] : pgas,co2 = partial pressure of carbon dioxide gas (bar, true value, i.e. not normalized) = 0.30649
adm_out[50] : pgas,total = total head space pressure of H2+CO2+CH4+H2O (bar, true value, i.e. not normalized) = 1.0201
adm_out[51] : qgas = gas flow rate normalized to atmospheric pressure (m3/d) = 71991.9918
Extra calculated outputs
Produced hydrogen gas (kg H2/d) = 0.11894
Produced methane gas (kg CH4/d) = 29373.8351
Produced carbon dioxide gas (kg CO2/d) = 37626.555
Energy content of methane gas (MJ/d) = 1469102.9911
Energy content of methane gas (kWh/d) = 408084.1642

Digester output from second reactor of 15200 m^3 (steady state)

adm_out[1] : Ssu = monosacharides (kg COD/m3) = 0.001219
adm_out[2] : Saa = amino acids (kg COD/m3) = 0.00031723
adm_out[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0.0058475
adm_out[4] : Sva = total valerate (kg COD/m3) = 0.00089143
adm_out[5] : Sbu = total butyrate (kg COD/m3) = 0.0010761
adm_out[6] : Spro = total propionate (kg COD/m3) = 0.0015969
adm_out[7] : Sac = total acetate (kg COD/m3) = 0.0043302
adm_out[8] : Sh2 = hydrogen gas (kg COD/m3) = 1.9055e-08
adm_out[9] : Sch4 = methane gas (kg COD/m3) = 0.050246
adm_out[10] : Sic = inorganic carbon (kmole C/m3) = 0.074751
adm_out[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.072906 (= 1.0207 kg N/m3)
adm_out[12] : Si = soluble inerts (kg COD/m3) = 0.1527
adm_out[13] : Xc = composites (kg COD/m3) = 0.21022
adm_out[14] : Xch = carbohydrates (kg COD/m3) = 0.0029282
adm_out[15] : Xpr = proteins (kg COD/m3) = 0.0047251
adm_out[16] : Xli = lipids (kg COD/m3) = 0.0090803
adm_out[17] : Xsu = sugar degraders (kg COD/m3) = 0.51239
adm_out[18] : Xaa = amino acid degraders (kg COD/m3) = 0.92623
adm_out[19] : Xfa = LCFA degraders (kg COD/m3) = 1.4817
adm_out[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0.34944
adm_out[21] : Xpro = propionate degraders (kg COD/m3) = 0.12106
adm_out[22] : Xac = acetate degraders (kg COD/m3) = 1.3663
adm_out[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0.64247
adm_out[24] : Xi = particulate inerts (kg COD/m3) = 17.9265
adm_out[25] : Scat+ = cations (base) (kmole/m3) = 2.1412e-24
adm_out[26] : San- = anions (acid) (kmole/m3) = 0.0048073
adm_out[27] : Flow rate (m3/d) = 2544
adm_out[28] : Temperature (degC) = 35
adm_out[34] : pH = pH within AD system = 7.2234
adm_out[35] : S_H+ = protons (kmole/m3) = 5.9786e-08
adm_out[36] : Sva- = valerate (kg COD/m3) = 0.00088758
adm_out[37] : Sbu- = butyrate (kg COD/m3) = 0.0010718
adm_out[38] : Spro- = propionate (kg COD/m3) = 0.0015897

adm_out[39] : Sac- = acetate (kg COD/m3) = 0.0043153
adm_out[40] : Shco3- = bicarbonate (kmole C/m3) = 0.066676
adm_out[41] : Sco2 = carbon dioxide (kmole C/m3) = 0.0080743
adm_out[42] : Snh3 = ammonia (kmole N/m3) = 0.0013292
adm_out[43] : Snh4+ = ammonium (kmole N/m3) = 0.071577
adm_out[44] : Sgas,h2 = hydrogen concentration in gas phase (kg COD/m3) = 9.7171e-07
adm_out[45] : Sgas,ch4 = methane concentration in gas phase (kg COD/m3) = 1.6496
adm_out[46] : Sgas,co2 = carbon dioxide concentration in gas phase (kmole C/m3) = 0.011597
adm_out[47] : pgas,h2 = partial pressure of hydrogen gas (bar, true value i.e. not normalized) = 1.556e-06
adm_out[48] : pgas,ch4 = partial pressure of methane gas (bar, true value i.e. not normalized) = 0.6604
adm_out[49] : pgas,co2 = partial pressure of carbon dioxide gas (bar, true value, i.e. not normalized) = 0.29714
adm_out[50] : pgas,total = total head space pressure of H2+CO2+CH4+H2O (bar, true value, i.e. not normalized) = 1.0132
adm_out[51] : qgas = gas flow rate normalized to atmospheric pressure (m3/d) = 2096.3081
Extra calculated outputs
Produced hydrogen gas (kg H2/d) = 0.00025457
Produced methane gas (kg CH4/d) = 864.3631
Produced carbon dioxide gas (kg CO2/d) = 1069.4889
Energy content of methane gas (MJ/d) = 43230.2564
Energy content of methane gas (kWh/d) = 12008.4046

Second digester sludge output after ADM2ASM interface

SI = 152.7046 mg COD/l
SS = 15.2783 mg COD/l
XI = 17926.5413 mg COD/l
XS = 4492.5682 mg COD/l
XBH = 0 mg COD/l
XBA = 0 mg COD/l
XP = 1133.8978 mg COD/l
SO = 0 mg -COD/l
SNO = 0 mg N/l
SNH = 1233.3853 mg N/l
SND = 0.031088 mg N/l
XND = 168.7544 mg N/l
SALK = 83.2914 mol HCO3/m3
TSS = 16586.6248 mg SS/l
Flow rate = 2544 m3/d
Temperature = 15 degC

SI_load = 388.4805 kg COD/d
SS_load = 38.8679 kg COD/d
XI_load = 45605.121 kg COD/d
XS_load = 11429.0934 kg COD/d
XBH_load = 0 kg COD/d

XBA_load = 0 kg COD/d
XP_load = 2884.6359 kg COD/d
SO_load = 0 kg -COD/d
SNO_load = 0 kg N/d
SNH_load = 3137.7322 kg N/d
SND_load = 0.079089 kg N/d
XND_load = 429.3113 kg N/d
SALK_load = 211.8932 kmol HCO3/d
TSS_load = 42196.3735 kg SS/d

4.8 Case 4B

Total input to digester 1 (identical to case 3B)

adm_in[1] : Ssu = monosacharides (kg COD/m3) = 0
adm_in[2] : Saa = amino acids (kg COD/m3) = 0
adm_in[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0
adm_in[4] : Sva = total valerate (kg COD/m3) = 0
adm_in[5] : Sbu = total butyrate (kg COD/m3) = 0
adm_in[6] : Spro = total propionate (kg COD/m3) = 0
adm_in[7] : Sac = total acetate (kg COD/m3) = 0
adm_in[8] : Sh2 = hydrogen gas (kg COD/m3) = 0
adm_in[9] : Sch4 = methane gas (kg COD/m3) = 0
adm_in[10] : Sic = inorganic carbon (kmole C/m3) = 0
adm_in[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.0011719
adm_in[12] : Si = soluble inerts (kg COD/m3) = 0
adm_in[13] : Xc = composites (kg COD/m3) = 47.0542
adm_in[14] : Xch = carbohydrates (kg COD/m3) = 0
adm_in[15] : Xpr = proteins (kg COD/m3) = 0
adm_in[16] : Xli = lipids (kg COD/m3) = 23.3182
adm_in[17] : Xsu = sugar degraders (kg COD/m3) = 0
adm_in[18] : Xaa = amino acid degraders (kg COD/m3) = 0
adm_in[19] : Xfa = LCFA degraders (kg COD/m3) = 0
adm_in[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0
adm_in[21] : Xpro = propionate degraders (kg COD/m3) = 0
adm_in[22] : Xac = acetate degraders (kg COD/m3) = 0
adm_in[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0
adm_in[24] : Xi = particulate inerts (kg COD/m3) = 0.9484
adm_in[25] : Scat+ = cations (base) (kmole/m3) = 0.04
adm_in[26] : San- = anions (acid) (kmole/m3) = 0.0048073
adm_in[27] : Flow rate (m3/d) = 2544
adm_in[28] : Temperature (degC) = 35

Digester output from first digester of 23700 m^3 (steady state), also input to digester 2 (states 1 to 28)

adm_out[1] : Ssu = monosacharides (kg COD/m3) = 0.022268
adm_out[2] : Saa = amino acids (kg COD/m3) = 0.0099104
adm_out[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0.25315
adm_out[4] : Sva = total valerate (kg COD/m3) = 0.020728
adm_out[5] : Sbu = total butyrate (kg COD/m3) = 0.028374
adm_out[6] : Spro = total propionate (kg COD/m3) = 0.038484
adm_out[7] : Sac = total acetate (kg COD/m3) = 0.093731
adm_out[8] : Sh2 = hydrogen gas (kg COD/m3) = 4.5403e-07
adm_out[9] : Sch4 = methane gas (kg COD/m3) = 0.073416
adm_out[10] : Sic = inorganic carbon (kmole C/m3) = 0.063201
adm_out[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.022372 (= 0.3132 kg N/m3)
adm_out[12] : Si = soluble inerts (kg COD/m3) = 3.9566
adm_out[13] : Xc = composites (kg COD/m3) = 8.4941

adm_out[14] : Xch = carbohydrates (kg COD/m3) = 0.084039
adm_out[15] : Xpr = proteins (kg COD/m3) = 0.084039
adm_out[16] : Xli = lipids (kg COD/m3) = 0.3737
adm_out[17] : Xsu = sugar degraders (kg COD/m3) = 0.8048
adm_out[18] : Xaa = amino acid degraders (kg COD/m3) = 0.52729
adm_out[19] : Xfa = LCFA degraders (kg COD/m3) = 1.6599
adm_out[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0.23229
adm_out[21] : Xpro = propionate degraders (kg COD/m3) = 0.11702
adm_out[22] : Xac = acetate degraders (kg COD/m3) = 1.3699
adm_out[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0.68473
adm_out[24] : Xi = particulate inerts (kg COD/m3) = 8.8615
adm_out[25] : Scat+ = cations (base) (kmole/m3) = 0.04
adm_out[26] : San- = anions (acid) (kmole/m3) = 0.0048073
adm_out[27] : Flow rate (m3/d) = 2544
adm_out[28] : Temperature (degC) = 35
adm_out[34] : pH = pH within AD system = 7.1422
adm_out[35] : S_H+ = protons (kmole/m3) = 7.2082e-08
adm_out[36] : Sva- = valerate (kg COD/m3) = 0.02062
adm_out[37] : Sbu- = butyrate (kg COD/m3) = 0.028239
adm_out[38] : Spro- = propionate (kg COD/m3) = 0.038275
adm_out[39] : Sac- = acetate (kg COD/m3) = 0.093344
adm_out[40] : Shco3- = bicarbonate (kmole C/m3) = 0.055149
adm_out[41] : Sco2 = carbon dioxide (kmole C/m3) = 0.0080519
adm_out[42] : Snh3 = ammonia (kmole N/m3) = 0.00033937
adm_out[43] : Snh4+ = ammonium (kmole N/m3) = 0.022032
adm_out[44] : Sgas,h2 = hydrogen concentration in gas phase (kg COD/m3) = 1.3836e-05
adm_out[45] : Sgas,ch4 = methane concentration in gas phase (kg COD/m3) = 1.6814
adm_out[46] : Sgas,co2 = carbon dioxide concentration in gas phase (kmole C/m3) = 0.01135
adm_out[47] : pgas,h2 = partial pressure of hydrogen gas (bar, true value i.e. not normalized) = 2.2155e-05
adm_out[48] : pgas,ch4 = partial pressure of methane gas (bar, true value i.e. not normalized) = 0.6731
adm_out[49] : pgas,co2 = partial pressure of carbon dioxide gas (bar, true value, i.e. not normalized) = 0.2908
adm_out[50] : pgas,total = total head space pressure of H2+CO2+CH4+H2O (bar, true value, i.e. not normalized) = 1.0196
adm_out[51] : qgas = gas flow rate normalized to atmospheric pressure (m3/d) = 66292.4234
Extra calculated outputs
Produced hydrogen gas (kg H2/d) = 0.11391
Produced methane gas (kg CH4/d) = 27685.41
Produced carbon dioxide gas (kg CO2/d) = 32892.1262
Energy content of methane gas (MJ/d) = 1384658.096
Energy content of methane gas (kWh/d) = 384627.2489

Digester output from second digester of 15200 m^3 (steady state)

adm_out[1] : Ssu = monosacharides (kg COD/m3) = 0.0051624
adm_out[2] : Saa = amino acids (kg COD/m3) = 0.002548
adm_out[3] : Sfa = long chain fatty acids (LCFA) (kg COD/m3) = 0.01841

adm_out[4] : Sva = total valerate (kg COD/m3) = 0.0050586
adm_out[5] : Sbu = total butyrate (kg COD/m3) = 0.0066524
adm_out[6] : Spro = total propionate (kg COD/m3) = 0.007104
adm_out[7] : Sac = total acetate (kg COD/m3) = 0.011479
adm_out[8] : Sh2 = hydrogen gas (kg COD/m3) = 7.2129e-08
adm_out[9] : Sch4 = methane gas (kg COD/m3) = 0.050226
adm_out[10] : Sic = inorganic carbon (kmole C/m3) = 0.073652
adm_out[11] : Sin = inorganic nitrogen (kmole N/m3) = 0.030021 (= 0.4203 kg N/m3)
adm_out[12] : Si = soluble inerts (kg COD/m3) = 4.6423
adm_out[13] : Xc = composites (kg COD/m3) = 2.2954
adm_out[14] : Xch = carbohydrates (kg COD/m3) = 0.023959
adm_out[15] : Xpr = proteins (kg COD/m3) = 0.023959
adm_out[16] : Xli = lipids (kg COD/m3) = 0.040015
adm_out[17] : Xsu = sugar degraders (kg COD/m3) = 0.85897
adm_out[18] : Xaa = amino acid degraders (kg COD/m3) = 0.57383
adm_out[19] : Xfa = LCFA degraders (kg COD/m3) = 1.6171
adm_out[20] : Xc4 = valerate and butyrate degraders (kg COD/m3) = 0.2541
adm_out[21] : Xpro = propionate degraders (kg COD/m3) = 0.12744
adm_out[22] : Xac = acetate degraders (kg COD/m3) = 1.3888
adm_out[23] : Xh2 = hydrogen degraders (kg COD/m3) = 0.69029
adm_out[24] : Xi = particulate inerts (kg COD/m3) = 10.2329
adm_out[25] : Scat+ = cations (base) (kmole/m3) = 0.04
adm_out[26] : San- = anions (acid) (kmole/m3) = 0.0048073
adm_out[27] : Flow rate (m3/d) = 2544
adm_out[28] : Temperature (degC) = 35
adm_out[34] : pH = pH within AD system = 7.1514
adm_out[35] : S_H+ = protons (kmole/m3) = 7.0568e-08
adm_out[36] : Sva- = valerate (kg COD/m3) = 0.0050329
adm_out[37] : Sbu- = butyrate (kg COD/m3) = 0.0066216
adm_out[38] : Spro- = propionate (kg COD/m3) = 0.0070662
adm_out[39] : Sac- = acetate (kg COD/m3) = 0.011433
adm_out[40] : Shco3- = bicarbonate (kmole C/m3) = 0.064441
adm_out[41] : Sco2 = carbon dioxide (kmole C/m3) = 0.0092109
adm_out[42] : Snh3 = ammonia (kmole N/m3) = 0.00046503
adm_out[43] : Snh4+ = ammonium (kmole N/m3) = 0.029556
adm_out[44] : Sgas,h2 = hydrogen concentration in gas phase (kg COD/m3) = 3.3432e-06
adm_out[45] : Sgas,ch4 = methane concentration in gas phase (kg COD/m3) = 1.549
adm_out[46] : Sgas,co2 = carbon dioxide concentration in gas phase (kmole C/m3) = 0.013193
adm_out[47] : pgas,h2 = partial pressure of hydrogen gas (bar, true value i.e. not normalized) = 5.3535e-06
adm_out[48] : pgas,ch4 = partial pressure of methane gas (bar, true value i.e. not normalized) = 0.62012
adm_out[49] : pgas,co2 = partial pressure of carbon dioxide gas (bar, true value, i.e. not normalized) = 0.33801
adm_out[50] : pgas,total = total head space pressure of H2+CO2+CH4+H2O (bar, true value, i.e. not normalized) = 1.0138
adm_out[51] : qgas = gas flow rate normalized to atmospheric pressure (m3/d) = 8077.2669
Extra calculated outputs

Produced hydrogen gas (kg H2/d) = 0.0033728
Produced methane gas (kg CH4/d) = 3125.4997
Produced carbon dioxide gas (kg CO2/d) = 4684.9092
Energy content of methane gas (MJ/d) = 156318.7426
Energy content of methane gas (kWh/d) = 43421.8729

5. Conclusions

As may be expected no major differences between the results from the two system layouts are visible during normal operating conditions, i.e. what we see in steady state. The potential advantages of a two-stage digestion system are more to be found during dynamic conditions and problematic operation (i.e. start-up, toxic and inhibitory events etc) as a result of higher flexibility. A somewhat higher gas production can be achieved by series operation.

It is quite remarkable to see how similar the results in terms of gas production are independently if the interface models are used or not (compare for example cases 1A and 1B). However, the content of the digested sludge is significantly influenced by the interfaces, which is an important factor when simulation entire WWTPs and where the sludge is later dewatered and the water recycled back to the beginning of the plant (especially in terms of ammonia, alkalinity and soluble inerts). It is mainly for this purpose that the interfaces were developed.

The results presented here are based on simulations without any knowledge of the true system apart from the highly limited input data discussed above. Moreover, no model calibration of any kind has been performed. Consequently, it would be interesting to know if the outputs in terms of gas production, gas content etc have any similarities with Stockholm Water data that are known from the actual system during the current input situation. If that would be the case it would certainly enhance the reliability and increase the future trust that can be put in the modified ADM1 and also the interface models developed by the IWA Task Group on Benchmarking.

The simulations predict that the gas flow rate from the existing system at Henriksdal's WWTP with the current load is in the range of 43000 to 45000 Nm^3/d, with a content of 63 to 64% methane and 30 to 32% carbon dioxide (the rest mostly water vapour) when the system is operating without any problems. Whether these values are close to reality remains to be determined and will provide valuable information concerning the accuracy of the models.

If the investigation is continued the following approach may be applied:

- more detailed characterisation of the current digester input;
- calibration of the ADM1 using data from existing operation;
- literature review on series operation of digesters and how dynamics, conversion rates, biomass fractionation is affected;
- including any type of existing control of the digester (unknown at this time);
- analysing dynamics by means of simulations in terms of start-up procedures and other aspects;
- investigate potential model modifications for VS and TS calculations.

Appendix A. C code of ASM2ADM interface model

```
/*
 * 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
```

Appendix B. C code of ADM2ASM interface model

```
/*
 * 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 used as output
T from the ADM2ASM 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] = 1.0/1.42*(y[2] + y[3] + y[4] + y[5] + y[6]); %MODIFIED FOR
HENRIKSDAL
    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
```

Appendix C. Example of initialisation file for simulations

```
% This file initiates parameter values and sets initial conditions for any
of the three
% model implementations adm1_ODE, adm1_DAE1 and adm1_DAE2. Note that some
of the
% parameter values deviates from the values given in the ADM1-STR
(Batstone, et al.).
%
% Copyright 2006:
% Dr Christian Rosen, Dr Darko Vrecko and Dr Ulf Jeppsson
% Dept. Industrial Electrical Engineering and Automation
% Lund University, Sweden
% http://www.iea.lth.se

S_su =  0.024309;
S_aa =  0.010808;
S_fa =  0.29533;
S_va =  0.02329;
S_bu = 0.031123;
S_pro = 0.043974;
S_ac =  0.50765;
S_h2 =  4.9652e-007;
S_ch4 = 0.055598;
S_IC = 0.10258;
S_IN = 0.10373;
S_I = 3.2327;
X_xc = 7.5567;
X_ch = 0.074679;
X_pr = 0.074679;
X_li = 0.11202;
X_su = 0.57565;
X_aa = 0.43307;
X_fa = 0.44433;
X_c4 = 0.18404;
X_pro = 0.087261;
X_ac = 0.57682;
X_h2 =  0.28774;
X_I =  18.6685;
S_cat =  3.3531e-042;
S_an =  1.5293e-041;
S_hva = 0.023204;   % is actually Sva-
S_hbu = 0.031017;   % is actually Sbu-
S_hpro =  0.043803; % is actually Spro-
S_hac =  0.50616;   % is actually Sac-
S_hco3 = 0.092928;
S_nh3 = 0.0021958;
S_gas_h2 = 1.9096e-005;
S_gas_ch4 = 1.5103;
S_gas_co2 = 0.013766;
Q_D = 166.5395;
T_D = 35;
S_D1_D = 0;
S_D2_D = 0;
S_D3_D = 0;
X_D4_D = 0;
X_D5_D = 0;

S_H_ion = 5.3469e-008;
```

```
% used by all three ADM implementations, adm1_ODE, adm1_DAE1 and adm1_DAE2.
DIGESTERINIT = [ S_su S_aa S_fa S_va S_bu S_pro S_ac S_h2 S_ch4 S_IC S_IN
S_I X_xc X_ch X_pr X_li X_su X_aa X_fa X_c4 X_pro X_ac X_h2 ...
                X_I S_cat S_an S_hva S_hbu S_hpro S_hac S_hco3 S_nh3
S_gas_h2 S_gas_ch4 S_gas_co2 Q_D T_D S_D1_D S_D2_D S_D3_D X_D4_D X_D5_D ];

% used by both DAE ADM implementations, adm1_DAE1 and adm1_DAE2.
PHSOLVINIT = [ S_H_ion S_hva S_hbu S_hpro S_hac S_hco3 S_nh3 ];

% used by one DAE ADM implementation, adm1_DAE2.
SH2SOLVINIT = [ S_h2 ];

f_sI_xc = 0.1;
f_xI_xc = 0.2;
f_ch_xc = 0.2;
f_pr_xc = 0.2;
f_li_xc = 0.3;
N_xc = 0.0376/14.0;
N_I = 0.06/14.0;
N_aa = 0.007;
C_xc = 0.02786;
C_sI = 0.03;
C_ch = 0.0313;
C_pr = 0.03;
C_li = 0.022;
C_xI = 0.03;
C_su = 0.0313;
C_aa = 0.03;
f_fa_li = 0.95;
C_fa = 0.0217;
f_h2_su = 0.19;
f_bu_su = 0.13;
f_pro_su = 0.27;
f_ac_su = 0.41;
N_bac = 0.08/14.0;
C_bu = 0.025;
C_pro = 0.0268;
C_ac = 0.0313;
C_bac = 0.0313;
Y_su = 0.1;
f_h2_aa = 0.06;
f_va_aa = 0.23;
f_bu_aa = 0.26;
f_pro_aa = 0.05;
f_ac_aa = 0.40;
C_va = 0.024;
Y_aa = 0.08;
Y_fa = 0.06;
Y_c4 = 0.06;
Y_pro = 0.04;
C_ch4 = 0.0156;
Y_ac = 0.05;
Y_h2 = 0.06;
k_dis = 0.5;
k_hyd_ch = 10.0;
k_hyd_pr = 10.0;
k_hyd_li = 10.0;
K_S_IN = 1.0e-4;
k_m_su = 30.0;
```

```
K_S_su = 0.5;
pH_UL_aa = 5.5;
pH_LL_aa = 4.0;
k_m_aa = 50.0;
K_S_aa = 0.3;
k_m_fa = 6.0;
K_S_fa = 0.4;
K_Ih2_fa = 5.0e-6;
k_m_c4 = 20.0;
K_S_c4 = 0.2;
K_Ih2_c4 = 1.0e-5;
k_m_pro = 13.0;
K_S_pro = 0.1;
K_Ih2_pro = 3.5e-6;
k_m_ac = 8.0;
K_S_ac = 0.15;
K_I_nh3 = 0.0018;
pH_UL_ac = 7.0;
pH_LL_ac = 6.0;
k_m_h2 = 35.0;
K_S_h2 = 7.0e-6;
pH_UL_h2 = 6.0;
pH_LL_h2 = 5.0;
k_dec_Xsu = 0.02;
k_dec_Xaa = 0.02;
k_dec_Xfa = 0.02;
k_dec_Xc4 = 0.02;
k_dec_Xpro = 0.02;
k_dec_Xac = 0.02;
k_dec_Xh2 = 0.02;
R = 0.083145;       % universal gas constant dm3*bar/(mol*K) = 8.3145
J/(mol*K)
T_base = 298.15;    % 25 degC = 298.15 K
T_op = 308.15;      % operational temperature of AD and interfaces, 35
degC, should be an input
pK_w_base = 14.0;
pK_a_va_base = 4.86;
pK_a_bu_base = 4.82;
pK_a_pro_base = 4.88;
pK_a_ac_base = 4.76;
pK_a_co2_base = 6.35;
pK_a_IN_base = 9.25;
k_A_Bva = 1.0e10;     % 1e8; according to STR
k_A_Bbu = 1.0e10;     % 1e8; according to STR
k_A_Bpro = 1.0e10;    % 1e8; according to STR
k_A_Bac = 1.0e10;     % 1e8; according to STR
k_A_Bco2 = 1.0e10;    % 1e8; according to STR
k_A_BIN = 1.0e10;     % 1e8; according to STR
P_atm = 1.013;        % bar
kLa = 200.0;
K_H_h2o_base = 0.0313;
K_H_co2_base = 0.035;
K_H_ch4_base = 0.0014;
K_H_h2_base = 7.8e-4;
k_P = 1.0e7;

DIGESTERPAR = [ f_sI_xc f_xI_xc f_ch_xc f_pr_xc f_li_xc N_xc N_I N_aa C_xc
C_sI C_ch C_pr C_li C_xI C_su C_aa f_fa_li C_fa ...
f_h2_su f_bu_su f_pro_su f_ac_su N_bac C_bu C_pro C_ac C_bac Y_su f_h2_aa
f_va_aa f_bu_aa f_pro_aa f_ac_aa C_va Y_aa Y_fa ...
```

```
Y_c4 Y_pro C_ch4 Y_ac Y_h2 k_dis k_hyd_ch k_hyd_pr k_hyd_li K_S_IN k_m_su
K_S_su pH_UL_aa pH_LL_aa k_m_aa K_S_aa k_m_fa ...
K_S_fa K_Ih2_fa k_m_c4 K_S_c4 K_Ih2_c4 k_m_pro K_S_pro K_Ih2_pro k_m_ac
K_S_ac K_I_nh3 pH_UL_ac pH_LL_ac k_m_h2 K_S_h2 ...
pH_UL_h2 pH_LL_h2 k_dec_Xsu k_dec_Xaa k_dec_Xfa k_dec_Xc4 k_dec_Xpro
k_dec_Xac k_dec_Xh2 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 k_A_Bva k_A_Bbu
k_A_Bpro k_A_Bac k_A_Bco2 k_A_BIN P_atm kLa ...
K_H_h2o_base K_H_co2_base K_H_ch4_base K_H_h2_base k_P ];

V_liq = 38900; %m3, size of Henriksdal AD
V_gas = 3000;  %m3, size of Henriksdal AD

DIM_D = [ V_liq V_gas ];

% parameters for ASM2ADM and ADM2ASM interfaces
CODequiv = 40.0/14.0;
fnaa = N_aa*14.0;     % fraction of N in amino acids and Xpr as in ADM1
report
fnxc = N_xc*14.0;     % N content of composite material based on BSM2
fnbac = N_bac*14.0;   % N content of biomass based on BSM1, same in AS and
AD
fxni = N_I*14.0;      % N content of inerts XI and XP, same in AS and AD
fsni = 0.0;           % N content of SI, assumed zero in ASM1 and BSM1
fsni_adm = N_I*14.0; % N content of SI in the AD system
% fnbac, fxni and fsni are adjusted to fit the benchmark values of iXB=0.08
and
% iXP=0.06 in the AS.
frlixs = 0.7;            % lipid fraction of non-nitrogenous XS in BSM2
frlibac = 0.4;           % lipid fraction of non-nitrogenous biomass in BSM2
frxs_adm = 0.68;         % anaerobically degradable fraction of AS biomass
in BSM2
fdegrade_adm = 0;        % amount of AS XI and XP degradable in AD, zero in
BSM2
frxs_as = 0.79;          % aerobically degradable fraction of AD biomass in
BSM2
fdegrade_as = 0;         % amount of AD XI and XP degradable in AS, zero in
BSM2

pH_adm_init = 7.0;       % initial value of pH in ADM to be used by
interfaces for the first sample

INTERFACEPAR = [ CODequiv fnaa fnxc fnbac fxni fsni fsni_adm frlixs frlibac
frxs_adm fdegrade_adm frxs_as fdegrade_as ...
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 ];

% parameters for the pHdelay function
PHINIT = pH_adm_init;
PHTIMECONST = 0.01;
```

Appendix D. C code of ADM1

```
/*
 * adm1_ODE.c is a C-file S-function for IAWQ AD Model No 1.
 * In addition to the ADM1, temperature dependency and dummy states are
added.
 * Some details are adjusted for BSM2 (pH inhibition, gas flow output etc).
 *
 * Copyright (2006):
 * Dr Christian Rosen, Dr Darko Vrecko and Dr Ulf Jeppsson
 * Dept. Industrial Electrical Engineering and Automation (IEA)
 * Lund University, Sweden
 * http://www.iea.lth.se/
 */

#define S_FUNCTION_NAME adm1_ODE

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

#define XINIT   ssGetArg(S,0)
#define PAR     ssGetArg(S,1)
#define V       ssGetArg(S,2)


/*
 * mdlInitializeSizes - initialize the sizes array
 */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumContStates(    S, 42);  /* number of continuous states
*/
    ssSetNumDiscStates(    S, 0);   /* number of discrete states
*/
    ssSetNumInputs(        S, 33);  /* number of inputs
*/
    ssSetNumOutputs(       S, 51);  /* number of outputs
*/
    ssSetDirectFeedThrough(S, 1);   /* direct feedthrough flag
*/
    ssSetNumSampleTimes(   S, 1);   /* number of sample times
*/
    ssSetNumSFcnParams(    S, 3);   /* 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)
{
int i;

for (i = 0; i < 42; i++) {
   x0[i] = mxGetPr(XINIT)[i];
}
}


/*
 * mdlOutputs - compute the outputs
 */
static void mdlOutputs(double *y, double *x, double *u, SimStruct *S, int
tid)
{
  double R, T_base, T_op, P_atm, p_gas_h2o, P_gas, k_P, q_gas, V_liq,
procT8, procT9, procT10, p_gas_h2, p_gas_ch4, p_gas_co2;
  double kLa, K_H_h2_base, K_H_ch4_base, K_H_co2_base, phi, S_H_ion,
pK_w_base, K_H_h2o_base, K_H_h2, K_H_ch4, K_H_co2, K_w, factor;
  int i;

  R = mxGetPr(PAR)[77];
  T_base = mxGetPr(PAR)[78];
  T_op = mxGetPr(PAR)[79];
  P_atm = mxGetPr(PAR)[93];
  p_gas_h2o = mxGetPr(PAR)[94];
  V_liq = mxGetPr(V)[0];
  kLa = mxGetPr(PAR)[95];
  K_H_h2_base = mxGetPr(PAR)[98];
  K_H_ch4_base = mxGetPr(PAR)[97];
  K_H_co2_base = mxGetPr(PAR)[96];
  K_H_h2o_base = mxGetPr(PAR)[95];
  pK_w_base = mxGetPr(PAR)[80];
  k_P = mxGetPr(PAR)[99];

  factor = (1.0/T_base - 1.0/T_op)/(100.0*R);
  K_H_h2 = K_H_h2_base*exp(-4180.0*factor);     /* T adjustment for K_H_h2
*/
  K_H_ch4 = K_H_ch4_base*exp(-14240.0*factor);  /* T adjustment for K_H_ch4
*/
  K_H_co2 = K_H_co2_base*exp(-19410.0*factor);  /* T adjustment for K_H_co2
*/
  K_w = pow(10,-pK_w_base)*exp(55900.0*factor); /* T adjustment for K_w */
  p_gas_h2o = K_H_h2o_base*exp(5290.0*(1.0/T_base - 1.0/T_op));  /* T
adjustement for water vapour saturation pressure */

  for (i = 0; i < 26; i++) {
      y[i] = x[i];
  }

  y[26] = u[26];   /* flow */
```

```
  y[27] = T_op - 273.15;       /* Temp = 35 degC */

  y[28] = u[28];   /* Dummy state 1, soluble */
  y[29] = u[29];   /* Dummy state 2, soluble */
  y[30] = u[30];   /* Dummy state 3, soluble */
  y[31] = u[31];   /* Dummy state 1, particulate */
  y[32] = u[32];   /* Dummy state 2, particulate */

  p_gas_h2 = x[32]*R*T_op/16.0;
  p_gas_ch4 = x[33]*R*T_op/64.0;
  p_gas_co2 = x[34]*R*T_op;
  P_gas = p_gas_h2 + p_gas_ch4 + p_gas_co2 + p_gas_h2o;

  procT8 = kLa*(x[7] - 16.0*K_H_h2*p_gas_h2);
  procT9 = kLa*(x[8] - 64.0*K_H_ch4*p_gas_ch4);
  procT10 = kLa*((x[9] - x[30]) - K_H_co2*p_gas_co2);

  /* q_gas = k_P*(P_gas - P_atm); */
  /*q_gas = R*T_op*V_liq*(procT8/16.0 + procT9/64.0 + procT10)/(P_atm-
p_gas_h2o); */

  q_gas = k_P*(P_gas - P_atm);
  if (q_gas < 0)
    q_gas = 0.0;

  phi = x[24]+(x[10]-x[31])-x[30]-x[29]/64.0-x[28]/112.0-x[27]/160.0-
x[26]/208.0-x[25];
  S_H_ion = -phi*0.5+0.5*sqrt(phi*phi+4.0*K_w);

  y[33] = -log10(S_H_ion);     /* pH */
  y[34] = S_H_ion;             /* SH+ */
  y[35] = x[26];               /* Sva- */
  y[36] = x[27];               /* Sbu- */
  y[37] = x[28];               /* Spro- */
  y[38] = x[29];               /* Sac- */
  y[39] = x[30];               /* Shco3- */
  y[40] = x[9] - x[30];        /* SCO2 */
  y[41] = x[31];               /* Snh3 */
  y[42] = x[10] - x[31];       /* SNH4+ */
  y[43] = x[32];               /* Sgas,h2 */
  y[44] = x[33];               /* Sgas,ch4 */
  y[45] = x[34];               /* Sgas,co2 */
  y[46] = p_gas_h2;
  y[47] = p_gas_ch4;
  y[48] = p_gas_co2;
  y[49] = P_gas;                /* total head space pressure from H2, CH4,
CO2 and H2O */
  y[50] = q_gas * P_gas/P_atm; /*The output gas flow is recalculated to
atmospheric pressure (normalization) */


}


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

```
/*
 * mdlDerivatives - compute the derivatives
 */
static void mdlDerivatives(double *dx, double *x, double *u, SimStruct *S,
int tid)
{

double f_sI_xc, f_xI_xc, f_ch_xc, f_pr_xc, f_li_xc, N_xc, N_I, N_aa, C_xc,
C_sI, C_ch;
double C_pr, C_li, C_xI, C_su, C_aa, f_fa_li, C_fa, f_h2_su, f_bu_su,
f_pro_su, f_ac_su;
double N_bac, C_bu, C_pro, C_ac, C_bac, Y_su, f_h2_aa, f_va_aa, f_bu_aa,
f_pro_aa, f_ac_aa;
double C_va, Y_aa, Y_fa, Y_c4, Y_pro, C_ch4, Y_ac, Y_h2;
double k_dis, k_hyd_ch, k_hyd_pr, k_hyd_li, K_S_IN, k_m_su, K_S_su,
pH_UL_aa, pH_LL_aa;
double k_m_aa, K_S_aa, k_m_fa, K_S_fa, K_Ih2_fa, k_m_c4, K_S_c4, K_Ih2_c4,
k_m_pro, K_S_pro;
double K_Ih2_pro, k_m_ac, K_S_ac, K_I_nh3, pH_UL_ac, pH_LL_ac, k_m_h2,
K_S_h2, pH_UL_h2, pH_LL_h2;
double k_dec_Xsu, k_dec_Xaa, k_dec_Xfa, k_dec_Xc4, k_dec_Xpro, k_dec_Xac,
k_dec_Xh2;
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 K_w, K_a_va, K_a_bu, K_a_pro, K_a_ac, K_a_co2, K_a_IN, K_H_co2,
K_H_ch4, K_H_h2;
double K_A_Bva, K_A_Bbu, K_A_Bpro, K_A_Bac, K_A_Bco2, K_A_BIN;
double P_atm, p_gas_h2o, P_gas, k_P, kLa, K_H_h2o_base, K_H_co2_base,
K_H_ch4_base, K_H_h2_base, factor;
double V_liq, V_gas;
double eps, pH_op, phi, S_H_ion;
double proc1, proc2, proc3, proc4, proc5, proc6, proc7, proc8, proc9,
proc10, proc11, proc12, proc13;
double proc14, proc15, proc16, proc17, proc18, proc19, procA4, procA5,
procA6, procA7, procA10, procA11;
double procT8, procT9, procT10;
double I_pH_aa, I_pH_ac, I_pH_h2, I_IN_lim, I_h2_fa, I_h2_c4, I_h2_pro,
I_nh3;
double reac1, reac2, reac3, reac4, reac5, reac6, reac7, reac8, reac9,
reac10, reac11, reac12, reac13;
double reac14, reac15, reac16, reac17, reac18, reac19, reac20, reac21,
reac22, reac23, reac24;
double stoich1, stoich2, stoich3, stoich4, stoich5, stoich6, stoich7,
stoich8, stoich9, stoich10, stoich11, stoich12, stoich13;
double xtemp[42], inhib[6];
double p_gas_h2, p_gas_ch4, p_gas_co2, q_gas;
double pHLim_aa, pHLim_ac, pHLim_h2, a_aa, a_ac, a_h2, n_aa, n_ac, n_h2;
int i;

eps = 0.000001;

f_sI_xc = mxGetPr(PAR)[0];
f_xI_xc = mxGetPr(PAR)[1];
f_ch_xc = mxGetPr(PAR)[2];
f_pr_xc = mxGetPr(PAR)[3];
f_li_xc = mxGetPr(PAR)[4];
N_xc = mxGetPr(PAR)[5];
N_I = mxGetPr(PAR)[6];
```

```
N_aa = mxGetPr(PAR)[7];
C_xc = mxGetPr(PAR)[8];
C_sI = mxGetPr(PAR)[9];
C_ch = mxGetPr(PAR)[10];
C_pr = mxGetPr(PAR)[11];
C_li = mxGetPr(PAR)[12];
C_xI = mxGetPr(PAR)[13];
C_su = mxGetPr(PAR)[14];
C_aa = mxGetPr(PAR)[15];
f_fa_li = mxGetPr(PAR)[16];
C_fa = mxGetPr(PAR)[17];
f_h2_su = mxGetPr(PAR)[18];
f_bu_su = mxGetPr(PAR)[19];
f_pro_su = mxGetPr(PAR)[20];
f_ac_su = mxGetPr(PAR)[21];
N_bac = mxGetPr(PAR)[22];
C_bu = mxGetPr(PAR)[23];
C_pro = mxGetPr(PAR)[24];
C_ac = mxGetPr(PAR)[25];
C_bac = mxGetPr(PAR)[26];
Y_su = mxGetPr(PAR)[27];
f_h2_aa = mxGetPr(PAR)[28];
f_va_aa = mxGetPr(PAR)[29];
f_bu_aa = mxGetPr(PAR)[30];
f_pro_aa = mxGetPr(PAR)[31];
f_ac_aa = mxGetPr(PAR)[32];
C_va = mxGetPr(PAR)[33];
Y_aa = mxGetPr(PAR)[34];
Y_fa = mxGetPr(PAR)[35];
Y_c4 = mxGetPr(PAR)[36];
Y_pro = mxGetPr(PAR)[37];
C_ch4 = mxGetPr(PAR)[38];
Y_ac = mxGetPr(PAR)[39];
Y_h2 = mxGetPr(PAR)[40];
k_dis = mxGetPr(PAR)[41];
k_hyd_ch = mxGetPr(PAR)[42];
k_hyd_pr = mxGetPr(PAR)[43];
k_hyd_li = mxGetPr(PAR)[44];
K_S_IN = mxGetPr(PAR)[45];
k_m_su = mxGetPr(PAR)[46];
K_S_su = mxGetPr(PAR)[47];
pH_UL_aa = mxGetPr(PAR)[48];
pH_LL_aa = mxGetPr(PAR)[49];
k_m_aa = mxGetPr(PAR)[50];
K_S_aa = mxGetPr(PAR)[51];
k_m_fa = mxGetPr(PAR)[52];
K_S_fa = mxGetPr(PAR)[53];
K_Ih2_fa = mxGetPr(PAR)[54];
k_m_c4 = mxGetPr(PAR)[55];
K_S_c4 = mxGetPr(PAR)[56];
K_Ih2_c4 = mxGetPr(PAR)[57];
k_m_pro = mxGetPr(PAR)[58];
K_S_pro = mxGetPr(PAR)[59];
K_Ih2_pro = mxGetPr(PAR)[60];
k_m_ac = mxGetPr(PAR)[61];
K_S_ac = mxGetPr(PAR)[62];
K_I_nh3 = mxGetPr(PAR)[63];
pH_UL_ac = mxGetPr(PAR)[64];
pH_LL_ac = mxGetPr(PAR)[65];
k_m_h2 = mxGetPr(PAR)[66];
```

```
K_S_h2 = mxGetPr(PAR)[67];
pH_UL_h2 = mxGetPr(PAR)[68];
pH_LL_h2 = mxGetPr(PAR)[69];
k_dec_Xsu = mxGetPr(PAR)[70];
k_dec_Xaa = mxGetPr(PAR)[71];
k_dec_Xfa = mxGetPr(PAR)[72];
k_dec_Xc4 = mxGetPr(PAR)[73];
k_dec_Xpro = mxGetPr(PAR)[74];
k_dec_Xac = mxGetPr(PAR)[75];
k_dec_Xh2 = mxGetPr(PAR)[76];
R = mxGetPr(PAR)[77];
T_base = mxGetPr(PAR)[78];
T_op = mxGetPr(PAR)[79];
pK_w_base = mxGetPr(PAR)[80];
pK_a_va_base = mxGetPr(PAR)[81];
pK_a_bu_base = mxGetPr(PAR)[82];
pK_a_pro_base = mxGetPr(PAR)[83];
pK_a_ac_base = mxGetPr(PAR)[84];
pK_a_co2_base = mxGetPr(PAR)[85];
pK_a_IN_base = mxGetPr(PAR)[86];
K_A_Bva = mxGetPr(PAR)[87];
K_A_Bbu = mxGetPr(PAR)[88];
K_A_Bpro = mxGetPr(PAR)[89];
K_A_Bac = mxGetPr(PAR)[90];
K_A_Bco2 = mxGetPr(PAR)[91];
K_A_BIN = mxGetPr(PAR)[92];
P_atm = mxGetPr(PAR)[93];
kLa = mxGetPr(PAR)[94];
K_H_h2o_base = mxGetPr(PAR)[95];
K_H_co2_base = mxGetPr(PAR)[96];
K_H_ch4_base = mxGetPr(PAR)[97];
K_H_h2_base = mxGetPr(PAR)[98];
k_P = mxGetPr(PAR)[99];

V_liq = mxGetPr(V)[0];
V_gas = mxGetPr(V)[1];

for (i = 0; i < 42; i++) {
   if (x[i] < 0)
     xtemp[i] = 0;
   else
     xtemp[i] = x[i];
}

factor = (1.0/T_base - 1.0/T_op)/(100.0*R);
K_w = pow(10,-pK_w_base)*exp(55900.0*factor); /* T adjustment for K_w */
K_a_va = pow(10,-pK_a_va_base);
K_a_bu = pow(10,-pK_a_bu_base);
K_a_pro = pow(10,-pK_a_pro_base);
K_a_ac = pow(10,-pK_a_ac_base);
K_a_co2 = pow(10,-pK_a_co2_base)*exp(7646.0*factor); /* T adjustment for
K_a_co2 */
K_a_IN = pow(10,-pK_a_IN_base)*exp(51965.0*factor); /* T adjustment for
K_a_IN */

K_H_h2 = K_H_h2_base*exp(-4180.0*factor);      /* T adjustment for K_H_h2 */
K_H_ch4 = K_H_ch4_base*exp(-14240.0*factor);   /* T adjustment for K_H_ch4
*/
K_H_co2 = K_H_co2_base*exp(-19410.0*factor);   /* T adjustment for K_H_co2
*/
```

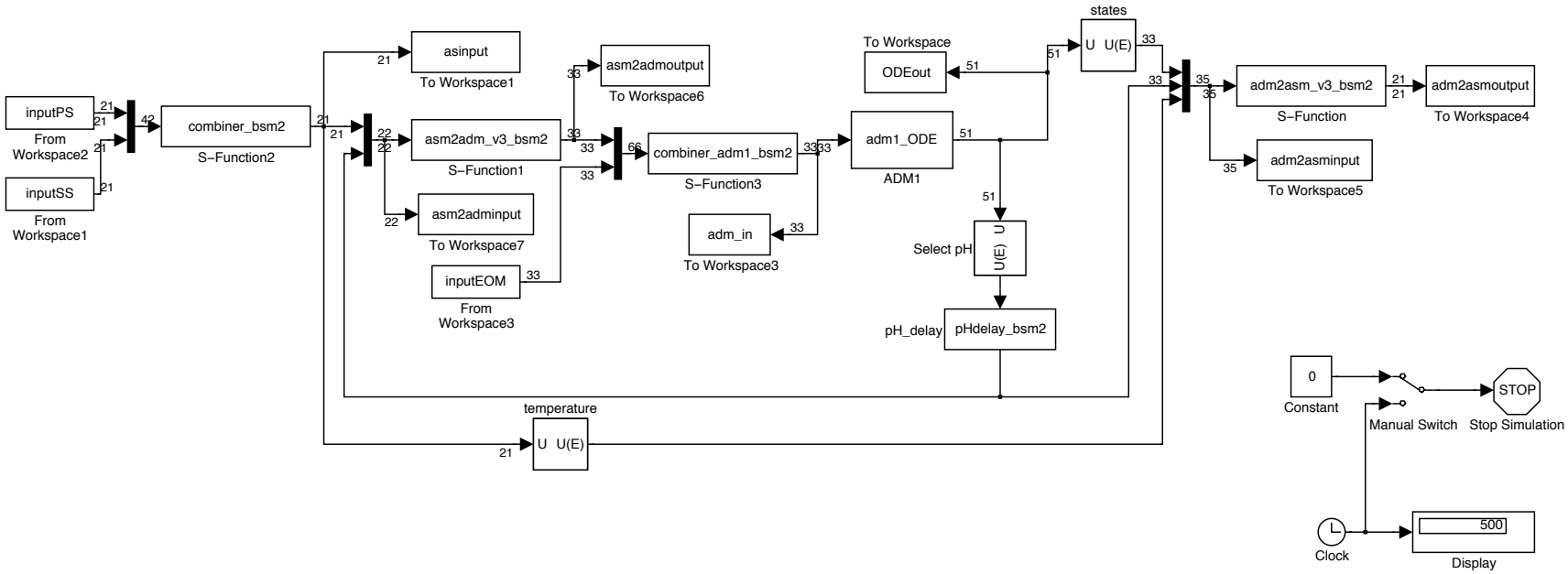
```
p_gas_h2o = K_H_h2o_base*exp(5290.0*(1.0/T_base - 1.0/T_op));  /* T
adjustement for water vapour saturation pressure */

phi = xtemp[24]+(xtemp[10]-xtemp[31])-xtemp[30]-xtemp[29]/64.0-
xtemp[28]/112.0-xtemp[27]/160.0-xtemp[26]/208.0-xtemp[25];
S_H_ion = -phi*0.5+0.5*sqrt(phi*phi+4.0*K_w); /* SH+ */
pH_op = -log10(S_H_ion); /* pH */

p_gas_h2 = xtemp[32]*R*T_op/16.0;
p_gas_ch4 = xtemp[33]*R*T_op/64.0;
p_gas_co2 = xtemp[34]*R*T_op;
P_gas = p_gas_h2 + p_gas_ch4 + p_gas_co2 + p_gas_h2o;

/* STRs pH inhibition function
if (pH_op < pH_UL_aa)
   I_pH_aa = exp(-3.0*(pH_op-pH_UL_aa)*(pH_op-pH_UL_aa)/((pH_UL_aa-
pH_LL_aa)*(pH_UL_aa-pH_LL_aa)));
else
   I_pH_aa = 1.0;
if (pH_op < pH_UL_ac)
   I_pH_ac = exp(-3.0*(pH_op-pH_UL_ac)*(pH_op-pH_UL_ac)/((pH_UL_ac-
pH_LL_ac)*(pH_UL_ac-pH_LL_ac)));
else
   I_pH_ac = 1.0;
if (pH_op < pH_UL_h2)
   I_pH_h2 = exp(-3.0*(pH_op-pH_UL_h2)*(pH_op-pH_UL_h2)/((pH_UL_h2-
pH_LL_h2)*(pH_UL_h2-pH_LL_h2)));
else
   I_pH_h2 = 1.0;
*/

/* Hill function on pH inhibition
pHLim_aa = (pH_UL_aa + pH_LL_aa)/2.0;
pHLim_ac = (pH_UL_ac + pH_LL_ac)/2.0;
pHLim_h2 = (pH_UL_h2 + pH_LL_h2)/2.0;
I_pH_aa = pow(pH_op,24)/(pow(pH_op,24)+pow(pHLim_aa ,24));
I_pH_ac = pow(pH_op,45)/(pow(pH_op,45)+pow(pHLim_ac ,45));
I_pH_h2 = pow(pH_op,45)/(pow(pH_op,45)+pow(pHLim_h2 ,45));
*/

/* MNPs function on pH inhibition, ADM1 Workshop, Copenhagen 2005.
a_aa = 25.0/(pH_UL_aa-pH_LL_aa+eps);
a_ac = 25.0/(pH_UL_ac-pH_LL_ac+eps);
a_h2 = 25.0/(pH_UL_h2-pH_LL_h2+eps);

I_pH_aa = 0.5*(1+tanh(a_aa*(pH_op/pHLim_aa - 1.0)));
I_pH_ac = 0.5*(1+tanh(a_ac*(pH_op/pHLim_ac - 1.0)));
I_pH_h2 = 0.5*(1+tanh(a_h2*(pH_op/pHLim_h2 - 1.0)));
*/

/* Hill function on SH+ used within BSM2, ADM1 Workshop, Copenhagen 2005.
*/
pHLim_aa = pow(10,(-(pH_UL_aa + pH_LL_aa)/2.0));
pHLim_ac = pow(10,(-(pH_UL_ac + pH_LL_ac)/2.0));
pHLim_h2 = pow(10,(-(pH_UL_h2 + pH_LL_h2)/2.0));
n_aa = 3.0/(pH_UL_aa-pH_LL_aa);
n_ac = 3.0/(pH_UL_ac-pH_LL_ac);
n_h2 = 3.0/(pH_UL_h2-pH_LL_h2);
I_pH_aa = pow(pHLim_aa,n_aa)/(pow(S_H_ion,n_aa)+pow(pHLim_aa ,n_aa));
I_pH_ac = pow(pHLim_ac,n_ac)/(pow(S_H_ion,n_ac)+pow(pHLim_ac ,n_ac));
```

```
I_pH_h2 = pow(pHLim_h2,n_h2)/(pow(S_H_ion,n_h2)+pow(pHLim_h2 ,n_h2));

I_IN_lim = 1.0/(1.0+K_S_IN/xtemp[10]);
I_h2_fa = 1.0/(1.0+xtemp[7]/K_Ih2_fa);
I_h2_c4 = 1.0/(1.0+xtemp[7]/K_Ih2_c4);
I_h2_pro = 1.0/(1.0+xtemp[7]/K_Ih2_pro);
I_nh3 = 1.0/(1.0+xtemp[31]/K_I_nh3);

inhib[0] = I_pH_aa*I_IN_lim;
inhib[1] = inhib[0]*I_h2_fa;
inhib[2] = inhib[0]*I_h2_c4;
inhib[3] = inhib[0]*I_h2_pro;
inhib[4] = I_pH_ac*I_IN_lim*I_nh3;
inhib[5] = I_pH_h2*I_IN_lim;

proc1 = k_dis*xtemp[12];
proc2 = k_hyd_ch*xtemp[13];
proc3 = k_hyd_pr*xtemp[14];
proc4 = k_hyd_li*xtemp[15];
proc5 = k_m_su*xtemp[0]/(K_S_su+xtemp[0])*xtemp[16]*inhib[0];
proc6 = k_m_aa*xtemp[1]/(K_S_aa+xtemp[1])*xtemp[17]*inhib[0];
proc7 = k_m_fa*xtemp[2]/(K_S_fa+xtemp[2])*xtemp[18]*inhib[1];
proc8 =
k_m_c4*xtemp[3]/(K_S_c4+xtemp[3])*xtemp[19]*xtemp[3]/(xtemp[3]+xtemp[4]+eps
)*inhib[2];
proc9 =
k_m_c4*xtemp[4]/(K_S_c4+xtemp[4])*xtemp[19]*xtemp[4]/(xtemp[3]+xtemp[4]+eps
)*inhib[2];
proc10 = k_m_pro*xtemp[5]/(K_S_pro+xtemp[5])*xtemp[20]*inhib[3];
proc11 = k_m_ac*xtemp[6]/(K_S_ac+xtemp[6])*xtemp[21]*inhib[4];
proc12 = k_m_h2*xtemp[7]/(K_S_h2+xtemp[7])*xtemp[22]*inhib[5];
proc13 = k_dec_Xsu*xtemp[16];
proc14 = k_dec_Xaa*xtemp[17];
proc15 = k_dec_Xfa*xtemp[18];
proc16 = k_dec_Xc4*xtemp[19];
proc17 = k_dec_Xpro*xtemp[20];
proc18 = k_dec_Xac*xtemp[21];
proc19 = k_dec_Xh2*xtemp[22];

procA4 = K_A_Bva*(xtemp[26]*(K_a_va+S_H_ion)-K_a_va*xtemp[3]);
procA5 = K_A_Bbu*(xtemp[27]*(K_a_bu+S_H_ion)-K_a_bu*xtemp[4]);
procA6 = K_A_Bpro*(xtemp[28]*(K_a_pro+S_H_ion)-K_a_pro*xtemp[5]);
procA7 = K_A_Bac*(xtemp[29]*(K_a_ac+S_H_ion)-K_a_ac*xtemp[6]);
procA10 = K_A_Bco2*(xtemp[30]*(K_a_co2+S_H_ion)-K_a_co2*xtemp[9]);
procA11 = K_A_BIN*(xtemp[31]*(K_a_IN+S_H_ion)-K_a_IN*xtemp[10]);

procT8 = kLa*(xtemp[7]-16.0*K_H_h2*p_gas_h2);
procT9 = kLa*(xtemp[8]-64.0*K_H_ch4*p_gas_ch4);
procT10 = kLa*((xtemp[9]-xtemp[30])-K_H_co2*p_gas_co2);

stoich1 = -
C_xc+f_sI_xc*C_sI+f_ch_xc*C_ch+f_pr_xc*C_pr+f_li_xc*C_li+f_xI_xc*C_xI;
stoich2 = -C_ch+C_su;
stoich3 = -C_pr+C_aa;
stoich4 = -C_li+(1.0-f_fa_li)*C_su+f_fa_li*C_fa;
stoich5 = -C_su+(1.0-
Y_su)*(f_bu_su*C_bu+f_pro_su*C_pro+f_ac_su*C_ac)+Y_su*C_bac;
stoich6 = -C_aa+(1.0-
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;
stoich7 = -C_fa+(1.0-Y_fa)*0.7*C_ac+Y_fa*C_bac;
```

```
stoich8 = -C_va+(1.0-Y_c4)*0.54*C_pro+(1.0-Y_c4)*0.31*C_ac+Y_c4*C_bac;
stoich9 = -C_bu+(1.0-Y_c4)*0.8*C_ac+Y_c4*C_bac;
stoich10 = -C_pro+(1.0-Y_pro)*0.57*C_ac+Y_pro*C_bac;
stoich11 = -C_ac+(1.0-Y_ac)*C_ch4+Y_ac*C_bac;
stoich12 = (1.0-Y_h2)*C_ch4+Y_h2*C_bac;
stoich13 = -C_bac+C_xc;

reac1 = proc2+(1.0-f_fa_li)*proc4-proc5;
reac2 = proc3-proc6;
reac3 = f_fa_li*proc4-proc7;
reac4 = (1.0-Y_aa)*f_va_aa*proc6-proc8;
reac5 = (1.0-Y_su)*f_bu_su*proc5+(1.0-Y_aa)*f_bu_aa*proc6-proc9;
reac6 = (1.0-Y_su)*f_pro_su*proc5+(1.0-Y_aa)*f_pro_aa*proc6+(1.0-
Y_c4)*0.54*proc8-proc10;
reac7 = (1.0-Y_su)*f_ac_su*proc5+(1.0-Y_aa)*f_ac_aa*proc6+(1.0-
Y_fa)*0.7*proc7+(1.0-Y_c4)*0.31*proc8+(1.0-Y_c4)*0.8*proc9+(1.0-
Y_pro)*0.57*proc10-proc11;
reac8 = (1.0-Y_su)*f_h2_su*proc5+(1.0-Y_aa)*f_h2_aa*proc6+(1.0-
Y_fa)*0.3*proc7+(1.0-Y_c4)*0.15*proc8+(1.0-Y_c4)*0.2*proc9+(1.0-
Y_pro)*0.43*proc10-proc12-procT8;
reac9 = (1.0-Y_ac)*proc11+(1.0-Y_h2)*proc12-procT9;
reac10 = -stoich1*proc1-stoich2*proc2-stoich3*proc3-stoich4*proc4-
stoich5*proc5-stoich6*proc6-stoich7*proc7-stoich8*proc8-stoich9*proc9-
stoich10*proc10-stoich11*proc11-stoich12*proc12-stoich13*proc13-
stoich13*proc14-stoich13*proc15-stoich13*proc16-stoich13*proc17-
stoich13*proc18-stoich13*proc19-procT10;
reac11 = (N_xc-f_xI_xc*N_I-f_sI_xc*N_I-f_pr_xc*N_aa)*proc1-
Y_su*N_bac*proc5+(N_aa-Y_aa*N_bac)*proc6-Y_fa*N_bac*proc7-Y_c4*N_bac*proc8-
Y_c4*N_bac*proc9-Y_pro*N_bac*proc10-Y_ac*N_bac*proc11-
Y_h2*N_bac*proc12+(N_bac-
N_xc)*(proc13+proc14+proc15+proc16+proc17+proc18+proc19);
reac12 = f_sI_xc*proc1;
reac13 = -proc1+proc13+proc14+proc15+proc16+proc17+proc18+proc19;
reac14 = f_ch_xc*proc1-proc2;
reac15 = f_pr_xc*proc1-proc3;
reac16 = f_li_xc*proc1-proc4;
reac17 = Y_su*proc5-proc13;
reac18 = Y_aa*proc6-proc14;
reac19 = Y_fa*proc7-proc15;
reac20 = Y_c4*proc8+Y_c4*proc9-proc16;
reac21 = Y_pro*proc10-proc17;
reac22 = Y_ac*proc11-proc18;
reac23 = Y_h2*proc12-proc19;
reac24 = f_xI_xc*proc1;

q_gas = k_P*(P_gas - P_atm);
if (q_gas < 0)
   q_gas = 0.0;

dx[0] = 1.0/V_liq*(u[26]*(u[0]-x[0]))+reac1;
dx[1] = 1.0/V_liq*(u[26]*(u[1]-x[1]))+reac2;
dx[2] = 1.0/V_liq*(u[26]*(u[2]-x[2]))+reac3;
dx[3] = 1.0/V_liq*(u[26]*(u[3]-x[3]))+reac4; /* Sva */
dx[4] = 1.0/V_liq*(u[26]*(u[4]-x[4]))+reac5; /* Sbu */
dx[5] = 1.0/V_liq*(u[26]*(u[5]-x[5]))+reac6; /* Spro */
dx[6] = 1.0/V_liq*(u[26]*(u[6]-x[6]))+reac7; /* Sac */
dx[7] = 1.0/V_liq*(u[26]*(u[7]-x[7]))+reac8;
dx[8] = 1.0/V_liq*(u[26]*(u[8]-x[8]))+reac9;
dx[9] = 1.0/V_liq*(u[26]*(u[9]-x[9]))+reac10;    /* SIC */
dx[10] = 1.0/V_liq*(u[26]*(u[10]-x[10]))+reac11; /* SIN */
```

```
dx[11] = 1.0/V_liq*(u[26]*(u[11]-x[11]))+reac12;
dx[12] = 1.0/V_liq*(u[26]*(u[12]-x[12]))+reac13;
dx[13] = 1.0/V_liq*(u[26]*(u[13]-x[13]))+reac14;
dx[14] = 1.0/V_liq*(u[26]*(u[14]-x[14]))+reac15;
dx[15] = 1.0/V_liq*(u[26]*(u[15]-x[15]))+reac16;
dx[16] = 1.0/V_liq*(u[26]*(u[16]-x[16]))+reac17;
dx[17] = 1.0/V_liq*(u[26]*(u[17]-x[17]))+reac18;
dx[18] = 1.0/V_liq*(u[26]*(u[18]-x[18]))+reac19;
dx[19] = 1.0/V_liq*(u[26]*(u[19]-x[19]))+reac20;
dx[20] = 1.0/V_liq*(u[26]*(u[20]-x[20]))+reac21;
dx[21] = 1.0/V_liq*(u[26]*(u[21]-x[21]))+reac22;
dx[22] = 1.0/V_liq*(u[26]*(u[22]-x[22]))+reac23;
dx[23] = 1.0/V_liq*(u[26]*(u[23]-x[23]))+reac24;

dx[24] = 1.0/V_liq*(u[26]*(u[24]-x[24])); /* Scat+ */
dx[25] = 1.0/V_liq*(u[26]*(u[25]-x[25])); /* San- */

dx[26] = -procA4;  /* Sva- */
dx[27] = -procA5;  /* Sbu- */
dx[28] = -procA6;  /* Spro- */
dx[29] = -procA7;  /* Sac- */
dx[30] = -procA10; /* SHCO3- */
dx[31] = -procA11; /* SNH3 */

dx[32] = -xtemp[32]*q_gas/V_gas+procT8*V_liq/V_gas;
dx[33] = -xtemp[33]*q_gas/V_gas+procT9*V_liq/V_gas;
dx[34] = -xtemp[34]*q_gas/V_gas+procT10*V_liq/V_gas;

dx[35] = 0; /* Flow */

dx[36] = 0; /* Temp */

/* Dummy states for future use */
dx[37] = 0;
dx[38] = 0;
dx[39] = 0;
dx[40] = 0;
dx[41] = 0;
}


/*
 * 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
```

Appendix E. Example of simulation layout (case 1A)


inputPS
From Workspace2
inputSS
From Workspace1
combiner_bsm2
S–Function2
asinput
To Workspace1
asm2admoutput
To Workspace6
asm2adm_v3_bsm2
S–Function1
asm2adminput
To Workspace7
inputEOM
From Workspace3
combiner_adm1_bsm2
S–Function3
adm_in
To Workspace3
To Workspace
ODEout
adm1_ODE
ADM1
states
U U(E)
Select pH
U(E) U
pH_delay
pHdelay_bsm2
temperature
U U(E)
adm2asm_v3_bsm2
S–Function
adm2asmoutput
To Workspace4
adm2asminput
To Workspace5
0
Constant
STOP
Manual Switch
Stop Simulation
Clock
500
Display

Appendix F. Detailed report of ADM1 and added modifications

Aspects on ADM1 implementation within the BSM2 framework

Christian Rosen and Ulf Jeppsson
Department of Industrial Electrical Engineering and Automation
Lund University, Box 118, SE-221 00 Lund, Sweden
christian.rosen@iea.lth.se, ulf.jeppsson@iea.lth.se

28th November 2006

1 Introduction

The current work of developing a control simulation benchmark goes back to the work carried out within the COST Action 682 ("Integrated Wastewater Management") and the IAWQ Task Group on Respirometry-Based control of the Activated Sludge Process in the late 90s. The work was continued in COST Action 624 until 2003 (Copp 2002). In 2005, the IWA Task Group on Benchmarking of Control Strategies for Wastewater Treatment Plants (BSM TG) was initiated and is now responsible for the continued development of the benchmark work.

This report describes the development of the Lund implementation of the Anaerobic Digester Model No 1 (ADM1). The aim of the report is to give an insight and rationale for the various changes and extensions made to the original model as reported in Batstone *et al.* (2002). Since the ADM1 was developed for general modelling of anaerobic digestion, opposed to for instance the ASM models, which were specifically developed for wastewater treatment, Batstone *et al.* (2002) leave some choices to the model implementor and this report will discuss how these choices were made in order to make this specific implementation as suitable for wastewater treatment sludge digestion as possible. The implementation was initiated by the inclusion of the sludge treatment in the IWA benchmark simulation model (BSM) to form a plant-wide or "within-the-fence" model.

2 The IWA benchmark simulation models

2.1 BSM1

The COST benchmark was originally defined as "a protocol to obtain a measure of performance of control strategies for activated sludge plants based on numerical, realistic simulations of the controlled plant". It was decided that the official plant layout would be aimed at carbon and nitrogen removal in a series of activated sludge reactors followed by a secondary settling tank, as this is perhaps the most common layout for full-scale WWT plants today. The selected model description for the biological processes was the recognised Activated Sludge Model no. 1, ASM1 (Henze *et al.* 2000), and the chosen settler model was a one-dimensional 10-layer model together with the double-exponential settling velocity model (Takacs *et al.* 1991).

To allow for uniform testing and evaluation three dynamic input data files have been developed, each representing different weather conditions (dry, rain and storm events) with realistic variations of the influent flow rate and composition. These files can be downloaded from the BSM TG website and have been widely used by various research groups also for other purposes than actual benchmarking. To represent a true challenge for on-line control strategies, the simulated plant is a high-loaded plant with significant influent variations.

In order to apply different control strategies a number of control handles (i.e. actuators) and measurement signals (i.e. sensors) must be available. A high degree of flexibility is required so that the implementation and evaluation of an innovative strategy is not limited. The default benchmark control strategy only use two measurement signals (dissolved oxygen and nitrate concentrations) and two control handles (air flow rate and internal recirculation flow rate). However, the benchmark simulation model allows for approximately 30 different control handles (different types of step-feed and step-recycling, external carbon source addition, etc.) and a wide variety of sensors. At this stage of development, all actuators are assumed to behave ideally whereas the sensors are divided into different classes

depending on their characteristics (delay, noise, detection limit, etc.). Consequently, just about every conceivable real-time control strategy for activated sludge systems can be implemented within the framework of the benchmark.

A general set of criteria has been defined within the benchmark description to assess the overall performance of the applied control strategy. Two system performance levels exist: (1) process performance and (2) control loop performance. The first level of assessment quantifies the effect of the control strategy on the plant in terms of an effluent quality index, effluent violations (related to defined limits for the plant) and operational costs (energy for pumping and aeration as well as sludge production). The second level of assessment quantifies the effect of the control strategy on controller performance by means of different statistical criteria on the controlled and manipulated variables. This more detailed analysis makes it possible to estimate the effect of a control strategy in terms of wear and tear of actuators, controller robustness, disturbance attenuation, etc.

All details of the current benchmark are available at the IWA TG website and also as a publication (Copp 2002). A substantial effort has gone into verifying the steady state and dynamic output data included in the description. Cross-platform checking of the benchmark has successfully demonstrated its use on a number of commercially available simulation software tools. The benchmark manual (Copp 2002) summarises the various tested implementations with helpful hints for new "benchmarkers". The complete manual can also be downloaded from the website. So far, results have been verified using BioWin™, EFOR™, GPS-X™, MATLAB/SIMULINK™, SIMBA®, STOAT™, WEST®, JASS and a user defined FORTRAN code.

2.2 BSM2

Although a valuable tool, the basic BSM1 does not allow for evaluation of control strategies on a plant-wide basis. BSM1 includes only an activated sludge system and a secondary clarifier. Consequently, only local control strategies can be evaluated. During the last decade the importance of integrated and plant-wide control has been stressed by the research community and the wastewater industry is starting to realise the benefits of such an approach. A WWTP should be considered as a unit, where primary and secondary clarification units, activated sludge reactors, anaerobic digesters, thickeners, dewatering systems, etc. are linked together and need to be 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.

It is the intent of the proposed extended benchmark system, BSM2, to take the issues stated above into account. Consequently, wastewater pre-treatment and the sludge train of the WWTP are included in the BSM2. To allow for more thorough evaluation and additional control handles operating on longer time-scales, the benchmark evaluation period is extended to one year (compared to one week in BSM1). The slow dynamics of anaerobic digestion processes also necessitates a pro-longed evaluation period. With this extended evaluation period, it is reasonable to include seasonal effects on the WWTP in terms of temperature variations. The data files describing the BSM1 influent wastewater (dry, storm and rain weather data) have been used extensively by researchers. However, for the extended benchmark system a phenomenological mathematical model has been developed for raw wastewater and storm water generation, including urban drainage and sewer system effects (Gernaey *et al.* 2005, Gernaey *et al.* 2006). Additionally, intermittent loads reaching the plant by other means of transportation (e.g. trucks) may be included. A more detailed description of the BSM2 is given in Jeppsson *et al.* (2006).

3 ADM1 ODE implementation

In this section, the ordinary differential equation (ODE) model implementation reported in earlier versions of this document is presented.

3.1 Introduction

The ADM1 implementation in Matlab/Simulink 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 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. As the BSM TG had access to the "official" ADM1 implementation in Aquasim, which was developed by the ADM TG during their development of the model, the unknown parameter values were first chosen in accordance with the suggestions given there.

3.1.1 Mass balances

To maintain complete mass balances for all model components (COD, N, etc.) is a fundamental issue of any model. ASM1 only assures mass balances on a COD basis (as not all nitrogen components are included) whereas ASM2d maintains balances of COD, N, P and charge. ASM3 adds theoretical oxygen demand as a conservation variable. Based on Batstone *et al.* (2002), a few comments are required to avoid problems for anybody implementing the model.

The ADM1 includes a process referred to as disintegration, where a composite material (X_C) is transformed into various compounds (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 get:

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

(note that carbohydrates and lipids contain no nitrogen). This means that for every kg of COD that disintegrates 0.1 mole of N is created (5% more than was originally there). 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 naturally 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 to $0.06/14 \approx 0.00429$ kmole N/kg COD to be consistent with the ASM1 model, where inert particulate organic material is assumed to have a nitrogen content of 6 g N/g COD. Because of the latter modification N_{xc} is set to $0.0376/14 \approx 0.00269$ kmole N/kg COD to maintain the nitrogen balance.

The ADM1 contains the state variables inorganic carbon and inorganic nitrogen. These 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. And what happens to the excess nitrogen within the biomass? 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 similar modification of the stoichiometric matrix should be done for the inorganic carbon for the processes "uptake of LCFA, valerate and butyrate" as well as for the disintegration and hydrolysis processes (both carbon and nitrogen).

The recommendation is to include stoichiometric relationships for all 19 processes regarding inorganic carbon and inorganic nitrogen. Basically all the required information is already given by Batstone *et al.* (2002). In many cases the expressions will be zero (depending on the selected values of the stoichiometric parameters) but there will be a 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 (finding coding errors in an implementation) much easier.

Using the proposed values of carbon content in the original ADM1 implementation it is stated that the carbon content of X_C is equal to 0.03 kmole C/kg COD. However, if we examine the carbon contents of the products arising from disintegration of composite material (based on the new fractionation parameters defined above) we get:

$$0.03 \cdot 0.1 \cdot S_I + 0.03 \cdot 0.2 \cdot X_I + 0.0313 \cdot 0.2 \cdot X_{ch} + 0.03 \cdot 0.2 \cdot X_{pr} + 0.022 \cdot 0.3 \cdot X_{li} = 0.02786 \text{ kmoleC/kgCOD}$$

If the original fractionation of composite material is used (i.e. $f_{xI,xc} = 0.25$ and $f_{li,xc} = 0.25$) we get a carbon content of the disintegrated products equal to 0.02826 kmole C/kg COD. In both cases, it is obvious that a significant amount of carbon "disappears" as a result of disintegration (6-7%). If the model is updated by adding the stoichiometric relationships to guarantee mass balances of carbon and nitrogen (described above) this disappearing carbon will end up as inorganic carbon and eventually it with lead to production of carbon dioxide in the gas phase. If the model is not updated as discussed above then 6-7% of the carbon content of composite material will simply be removed and the carbon mass balance will not hold. Moreover, as this extra carbon eventually ends up as carbon dioxide in the gas phase the original ADM1 model shows a tendency to produce a somewhat high percentage of CO_2 (32-35%) and somewhat low percentage of CH_4 (55-58%) in the produced gas using a reasonable sludge input. Note that the mass of produced CH_4 is still reasonable as this carbon unbalance leads to higher gas flow rate.

To avoid this problem it is suggested to use a value of 0.02786 kmole C/kg COD to describe the carbon content of composite material in the benchmark implementation. For reasonable sludge inputs this modification will normally lead to a production of 60-65% methane in the gas phase.

3.1.2 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 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:

$$K_{A,Bva}(S_{va^-}S_{H^+} - K_{a,va}S_{hva})$$

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

$$K_{A,Bva}(S_{va^-}(K_{a,va} + S_{H^+}) - K_{a,va}S_{va})$$

and, consequently, change the stoichiometry since S_{va} is not affected when the equilibrium ofS_{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 modelling the pH should not affect the overall results of the model. The DAE can be said to be a 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, we need to find the rate coefficients $k_{A,Bi}$ 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 1e8 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 1e10 M^{-1} d^{-1} is more appropriate. This will be illustrated in the results section of this report.

3.1.3 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 vapour 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 contants (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. Discussions between the ADM1 TG and the BSM TG on this topic are ongoing.

3.2 Model equations

3.2.1 Process rates

Biochemical process rates

$$
\begin{aligned}
\rho_1 &= k_{dis} \cdot X_c \\
\rho_2 &= k_{hyd,ch} \cdot X_{ch} \\
\rho_3 &= k_{hyd,pr} \cdot X_{pr} \\
\rho_4 &= k_{hyd,li} \cdot X_{li} \\
\rho_5 &= k_{m,su} \cdot \frac{S_{su}}{K_{S,su} + S_{su}} \cdot X_{su} \cdot I_5 \\
\rho_6 &= k_{m,aa} \cdot \frac{S_{aa}}{K_{S,aa} + S_{aa}} \cdot X_{aa} \cdot I_6 \\
\rho_7 &= k_{m,fa} \cdot \frac{S_{fa}}{K_{S,fa} + S_{fa}} \cdot X_{fa} \cdot I_7 \\
\rho_8 &= k_{m,c4} \cdot \frac{S_{va}}{K_{S,c4} + S_{va}} \cdot X_{c4} \cdot \frac{S_{va}}{S_{bu} + S_{va} + 1e-6} \cdot I_8 \\
\rho_9 &= k_{m,c4} \cdot \frac{S_{bu}}{K_{S,c4} + S_{bu}} \cdot X_{c4} \cdot \frac{S_{bu}}{S_{va} + S_{bu} + 1e-6} \cdot I_9 \\
\rho_{10} &= k_{m,pr} \cdot \frac{S_{pro}}{K_{S,pro} + S_{pro}} \cdot X_{pro} \cdot I_{10} \\
\rho_{11} &= k_{m,ac} \cdot \frac{S_{ac}}{K_{S,ac} + S_{ac}} \cdot X_{ac} \cdot I_{11} \\
\rho_{12} &= k_{m,h2} \cdot \frac{S_{h2}}{K_{S,h2} + S_{h2}} \cdot X_{h2} \cdot I_{12} \\
\rho_{13} &= k_{dec,Xsu} \cdot X_{su} \\
\rho_{14} &= k_{dec,Xaa} \cdot X_{aa} \\
\rho_{15} &= k_{dec,Xfa} \cdot X_{fa} \\
\rho_{16} &= k_{dec,Xc4} \cdot X_{c4} \\
\rho_{17} &= k_{dec,Xpro} \cdot X_{pro} \\
\rho_{18} &= k_{dec,Xac} \cdot X_{ac} \\
\rho_{19} &= k_{dec,Xh2} \cdot X_{h2}
\end{aligned}
$$

In the expressions for ρ_8 and ρ_9, a small constant (1e-6) has been added in order to avoid division by zero in the case of poor choice of initial conditions for S_{va} and S_{bu}, respectively.

Acid-base rates:

$$\rho_{A,4} = k_{A,Bva} \left(S_{va^-} (K_{a,va} + S_{H^+}) - K_{a,va} S_{va} \right)$$

$$
\begin{aligned}
\rho_{A,5} &= k_{A,Bbu}\left(S_{bu^-}(K_{a,bu}+S_{H^+})-K_{a,bu}S_{bu}\right)\\
\rho_{A,6} &= k_{A,Bpro}\left(S_{pro^-}(K_{a,pro}+S_{H^+})-K_{a,pro}S_{pro}\right)\\
\rho_{A,7} &= k_{A,Bac}\left(S_{ac^-}(K_{a,ac}+S_{H^+})-K_{a,ac}S_{ac}\right)\\
\rho_{A,10} &= k_{A,Bco2}\left(S_{hco3^-}(K_{a,co2}+S_{H^+})-K_{a,co2}S_{IC}\right)\\
\rho_{A,11} &= k_{A,BIN}\left(S_{nh3}(K_{a,IN}+S_{H^+})-K_{a,IN}S_{IN}\right)
\end{aligned}
$$

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

$$
\begin{aligned}
\rho_{T,8} &= k_La\left(S_{h2}-16\cdot K_{H,h2}p_{gas,h2}\right)\\
\rho_{T,9} &= k_La\left(S_{ch4}-64\cdot K_{H,ch4}p_{gas,ch4}\right)\\
\rho_{T,10} &= k_La\left(S_{co2}-K_{H,co2}p_{gas,co2}\right)
\end{aligned}
$$

3.2.2 Process inhibition

Inhibition:

$$I_{5,6} = I_{pH,aa}\cdot I_{IN,lim}$$

$$I_7 = I_{pH,aa}\cdot I_{IN,lim}\cdot I_{h2,fa}$$

$$I_{8,9} = I_{pH,aa}\cdot I_{IN,lim}\cdot I_{h2,c4}$$

$$I_{10} = I_{pH,aa}\cdot I_{IN,lim}\cdot I_{h2,pro}$$

$$I_{11} = I_{pH,ac}\cdot I_{IN,lim}\cdot I_{nh3}$$

$$I_{12} = I_{pH,h2}\cdot I_{IN,lim}$$

$$
\begin{aligned}
I_{pH,aa} &= \begin{cases} exp\left(-3\left(\frac{pH-pH_{UL,aa}}{pH_{UL,aa}-pH_{LL,aa}}\right)^2\right) & : \quad pH<pH_{UL,aa}\\ 1 & : \quad pH>pH_{UL,aa}\end{cases}\\
I_{pH,ac} &= \begin{cases} exp\left(-3\left(\frac{pH-pH_{UL,ac}}{pH_{UL,ac}-pH_{LL,ac}}\right)^2\right) & : \quad pH<pH_{UL,ac}\\ 1 & : \quad pH>pH_{UL,ac}\end{cases}\\
I_{pH,h2} &= \begin{cases} exp\left(-3\left(\frac{pH-pH_{UL,h2}}{pH_{UL,h2}-pH_{LL,h2}}\right)^2\right) & : \quad pH<pH_{UL,h2}\\ 1 & : \quad pH>pH_{UL,h2}\end{cases}\\
I_{IN,lim} &= \frac{1}{1+K_{S,IN}/S_{IN}}\\
I_{h2,fa} &= \frac{1}{1+S_{h2}/K_{I,h2,fa}}\\
I_{h2,c4} &= \frac{1}{1+S_{h2}/K_{I,h2,c4}}\\
I_{h2,pro} &= \frac{1}{1+S_{h2}/K_{I,h2,pro}}\\
I_{nh3} &= \frac{1}{1+S_{nh3}/K_{I,nh3}}
\end{aligned}
$$

$$pH = -log_{10}(S_{H^+})$$

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:

$$I_{pH} = \frac{1}{2}(1 + tan(a\varphi)) \tag{1}$$

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:

$$I_{pH} = \frac{pH^n}{pH^n + K_{pH}^n} \tag{2}$$

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:

$$I_{pH} = \frac{K_{pH}^n}{S_{H^+}^n + K_{pH}^n} \tag{3}$$

with

$$K_{pH} = 10^{-\frac{pH_{LL}+pH_{UL}}{2}}$$

The appropriate values of n 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.

NOTE! For any practical purpose, the choice of function among the three above is arbitrary but for BSM2 the hydrogen ion based function, i.e. Expression 3 above is chosen.

It should be noted that the appropriate values of n are dependent of the values of pH_{LL} and pH_{UL}. Therefore, if these limits are to be changed, it may be wise to implement:

$$n_{aa} = \frac{3.0}{pH_{UL,aa} - pH_{LL,aa}}$$
$$n_{ac} = \frac{3.0}{pH_{UL,ac} - pH_{LL,ac}}$$
$$n_{h2} = \frac{3.0}{pH_{UL,h2} - pH_{LL,h2}}$$

for the hydrogen ion based function.

3.2.3 Water phase equations

Differential equations 1-4, soluble matter: State No.

$$\frac{dS_{su}}{dt} = \frac{q_{in}}{V_{liq}}(S_{su,in} - S_{su}) + \rho_2 + (1 - f_{fa,li})\,\rho_4 - \rho_5 \quad (1)$$

$$\frac{dS_{aa}}{dt} = \frac{q_{in}}{V_{liq}}(S_{aa,in} - S_{aa}) + \rho_3 - \rho_6 \quad (2)$$

$$\frac{dS_{fa}}{dt} = \frac{q_{in}}{V_{liq}}(S_{fa,in} - S_{fa}) + f_{fa,li}\rho_4 - \rho_7 \quad (3)$$

$$\frac{dS_{va}}{dt} = \frac{q_{in}}{V_{liq}}(S_{va,in} - S_{va}) + (1 - Y_{aa})\,f_{va,aa}\rho_6 - \rho_8 \quad (4)$$

Differential equations 5-8, soluble matter: State No.

$$\frac{dS_{bu}}{dt} = \frac{q_{in}}{V_{liq}}(S_{bu,in} - S_{bu}) + (1 - Y_{su})\,f_{bu,su}\rho_5 + (1 - Y_{aa})\,f_{bu,aa}\rho_6 - \rho_9 \quad (5)$$

$$\frac{dS_{pro}}{dt} = \frac{q_{in}}{V_{liq}}(S_{pro,in} - S_{pro}) + (1 - Y_{su})\,f_{pro,su}\rho_5 + (1 - Y_{aa})\,f_{pro,aa}\rho_6 + (1 - Y_{c4})\,0.54\rho_8 - \rho_{10} \quad (6)$$

$$\frac{dS_{ac}}{dt} = \frac{q_{in}}{V_{liq}}(S_{ac,in} - S_{ac}) + (1 - Y_{su})\,f_{ac,su}\rho_5 + (1 - Y_{aa})\,f_{ac,aa}\rho_6 + (1 - Y_{fa})\,0.7\rho_7 + (1 - Y_{c4})\,0.31\rho_8 + (1 - Y_{c4})\,0.8\rho_9 + (1 - Y_{pro})\,0.57\rho_{10} - \rho_{11} \quad (7)$$

$$\frac{dS_{h2}}{dt} = \frac{q_{in}}{V_{liq}}(S_{h2,in} - S_{h2}) + (1 - Y_{su})\,f_{h2,su}\rho_5 + (1 - Y_{aa})\,f_{h2,aa}\rho_6 + (1 - Y_{fa})\,0.3\rho_7 + (1 - Y_{c4})\,0.15\rho_8 + (1 - Y_{c4})\,0.2\rho_9 + (1 - Y_{pro})\,0.43\rho_{10} - \rho_{12} - \rho_{T,8} \quad (8)$$

Differential equations 9-12, soluble matter: State No.

$$\frac{dS_{ch4}}{dt} = \frac{q_{in}}{V_{liq}}(S_{ch4,in} - S_{ch4}) + (1 - Y_{ac})\,\rho_{11} + (1 - Y_{h2})\,\rho_{12} - \rho_{T,9} \quad (9)$$

$$\frac{dS_{IC}{}^{*}}{dt} = \frac{q_{in}}{V_{liq}}(S_{IC,in} - S_{IC}) - \sum_{j=1}^{19}\left(\sum_{i=1-9,11-24} C_i v_{i,j}\rho_j\right) - \rho_{T,10} \quad (10)$$

$$\frac{dS_{IN}}{dt} = \frac{q_{in}}{V_{liq}}(S_{IN,in} - S_{IN}) - Y_{su}N_{bac}\rho_5 + (N_{aa} - Y_{aa}N_{bac})\,\rho_6 - Y_{fa}N_{bac}\rho_7 - Y_{c4}N_{bac}\rho_8 - Y_{c4}N_{bac}\rho_9 - Y_{pro}N_{bac}\rho_{10} - Y_{ac}N_{bac}\rho_{11} - Y_{h2}N_{bac}\rho_{12} + (N_{bac} - N_{xc})\sum_{i=13}^{19}\rho_i + (N_{xc} - f_{xI,xc}N_I - f_{sI,xc}N_I - f_{pr,xc}N_{aa})\,\rho_1 \quad (11)$$

$$\frac{dS_I}{dt} = \frac{q_{in}}{V_{liq}}(S_{I,in} - S_I) + f_{sI,xc}\rho_1 \quad (12)$$

* See next page for further explanation.

More specifically, the sum in Equation 10 is calculated as:

$$\sum_{j=1}^{19}\left(\sum_{i=1-9,11-24} C_i v_{i,j}\rho_j\right) = \sum_{k=1}^{12} s_k\rho_k + s_{13}(\rho_{13}+\rho_{14}+\rho_{15}+\rho_{16}+\rho_{17}+\rho_{18}+\rho_{19})$$

where

$$\begin{aligned}
s_1 &= -C_{xc} + f_{sI,xc}C_{sI} + f_{ch,xc}C_{ch} + f_{pr,xc}C_{pr} + f_{li,xc}C_{li} + f_{xI,xc}C_{xI}\\
s_2 &= -C_{ch} + C_{su}\\
s_3 &= -C_{pr} + C_{aa}\\
s_4 &= -C_{li} + (1-f_{fa,li})C_{su} + f_{fa,li}C_{fa}\\
s_5 &= -C_{su} + (1-Y_{su})(f_{bu,su}C_{bu} + f_{pro,su}C_{pro} + f_{ac,su}C_{ac}) + Y_{su}C_{bac}\\
s_6 &= -C_{aa} + (1-Y_{aa})(f_{va,aa}C_{va} + f_{bu,aa}C_{bu} + f_{pro,aa}C_{pro} + f_{ac,aa}C_{ac}) + Y_{aa}C_{bac}\\
s_7 &= -C_{fa} + (1-Y_{fa})\,0.7C_{ac} + Y_{fa}C_{bac}\\
s_8 &= -C_{va} + (1-Y_{c4})\,0.54C_{pro} + (1-Y_{c4})\,0.31C_{ac} + Y_{c4}C_{bac}\\
s_9 &= -C_{bu} + (1-Y_{c4})\,0.8C_{ac} + Y_{c4}C_{bac}\\
s_{10} &= -C_{pro} + (1-Y_{pro})\,0.57C_{ac} + Y_{pro}C_{bac}\\
s_{11} &= -C_{ac} + (1-Y_{ac})C_{ch4} + Y_{ac}C_{bac}\\
s_{12} &= (1-Y_{h2})C_{ch4} + Y_{h2}C_{bac}\\
s_{13} &= -C_{bac} + C_{xc}
\end{aligned}$$

Differential equations 13-16, particulate matter: State No.

$$\frac{dX_c}{dt} = \frac{q_{in}}{V_{liq}}(X_{c,in} - X_c) - \rho_1 + \sum_{i=13}^{19}\rho_i \quad (13)$$

$$\frac{dX_{ch}}{dt} = \frac{q_{in}}{V_{liq}}(X_{ch,in} - X_{ch}) + f_{ch,xc}\rho_1 - \rho_2 \quad (14)$$

$$\frac{dX_{pr}}{dt} = \frac{q_{in}}{V_{liq}}(X_{pr,in} - X_{pr}) + f_{pr,xc}\rho_1 - \rho_3 \quad (15)$$

$$\frac{dX_{li}}{dt} = \frac{q_{in}}{V_{liq}}(X_{li,in} - X_{li}) + f_{li,xc}\rho_1 - \rho_4 \quad (16)$$

Differential equations 17-20, particulate matter: State No.

$$\frac{dX_{su}}{dt} = \frac{q_{in}}{V_{liq}}(X_{su,in} - X_{su}) + Y_{su}\rho_5 - \rho_{13} \quad (17)$$

$$\frac{dX_{aa}}{dt} = \frac{q_{in}}{V_{liq}}(X_{aa,in} - X_{aa}) + Y_{aa}\rho_6 - \rho_{14} \quad (18)$$

$$\frac{dX_{fa}}{dt} = \frac{q_{in}}{V_{liq}}(X_{fa,in} - X_{fa}) + Y_{fa}\rho_7 - \rho_{15} \quad (19)$$

$$\frac{dX_{c4}}{dt} = \frac{q_{in}}{V_{liq}}(X_{c4,in} - X_{c4}) + Y_{c4}\rho_8 + Y_{c4}\rho_9 - \rho_{16} \quad (20)$$

Differential equations 21-24, particulate matter: State No.

$$\frac{dX_{pro}}{dt} = \frac{q_{in}}{V_{liq}}(X_{pro,in} - X_{pro}) + Y_{pro}\rho_{10} - \rho_{17} \quad (21)$$

$$\frac{dX_{ac}}{dt} = \frac{q_{in}}{V_{liq}}(X_{ac,in} - X_{ac}) + Y_{ac}\rho_{11} - \rho_{18} \quad (22)$$

$$\frac{dX_{h2}}{dt} = \frac{q_{in}}{V_{liq}}(X_{h2,in} - X_{h2}) + Y_{h2}\rho_{12} - \rho_{19} \quad (23)$$

$$\frac{dX_I}{dt} = \frac{q_{in}}{V_{liq}}(X_{I,in} - X_I) + f_{xI,xc}\rho_1 \quad (24)$$

Differential equations 25-26, cations and anions: State No.

$$\frac{dS_{cat^+}}{dt} = \frac{q_{in}}{V_{liq}}\left(S_{cat^+,in} - S_{cat^+}\right) \quad (25)$$

$$\frac{dS_{an^-}}{dt} = \frac{q_{in}}{V_{liq}}\left(S_{an^-,in} - S_{an^-}\right) \quad (26)$$

Differential equations 27-32, ion states: State No.

$$\frac{dS_{va^-}}{dt} = -\rho_{A,4} \quad (27)$$

$$\frac{dS_{bu^-}}{dt} = -\rho_{A,5} \quad (28)$$

$$\frac{dS_{pro^-}}{dt} = -\rho_{A,6} \quad (29)$$

$$\frac{dS_{ac^-}}{dt} = -\rho_{A,7} \quad (30)$$

$$\frac{dS_{hco3^-}}{dt} = -\rho_{A,10} \quad (31)$$

$$\frac{dS_{nh3}}{dt} = -\rho_{A,11} \quad (32)$$

Algebraic equation:

$$S_{H^+} = -\frac{\Theta}{2} + \frac{1}{2}\sqrt{\Theta^2 + 4K_W}$$

$$\Theta = S_{cat^+} + S_{nh4^+} - S_{hco3^-} - \frac{S_{ac^-}}{64} - \frac{S_{pro^-}}{112} - \frac{S_{bu^-}}{160} - \frac{S_{va^-}}{208} - S_{an^-}$$

$$S_{nh4^+} = S_{IN} - S_{nh3}$$

$$S_{co2} = S_{IC} - S_{hco3^-}$$

3.2.4 Gas phase equations

Differential equations: State No.

$$\frac{dS_{gas,h2}}{dt} = -\frac{S_{gas,h2}q_{gas}}{V_{gas}} + \rho_{T,8} \cdot \frac{V_{liq}}{V_{gas}} \quad (33)$$

$$\frac{dS_{gas,ch4}}{dt} = -\frac{S_{gas,ch4}q_{gas}}{V_{gas}} + \rho_{T,9} \cdot \frac{V_{liq}}{V_{gas}} \quad (34)$$

$$\frac{dS_{gas,co2}}{dt} = -\frac{S_{gas,co2}q_{gas}}{V_{gas}} + \rho_{T,10} \cdot \frac{V_{liq}}{V_{gas}} \quad (35)$$

Algebraic equations:

$$\begin{aligned} p_{gas,h2} &= S_{gas,h2} \cdot \frac{RT_{op}}{16} \\ p_{gas,ch4} &= S_{gas,ch4} \cdot \frac{RT_{op}}{64} \\ p_{gas,co2} &= S_{gas,co2} \cdot RT_{op} \\ q_{gas} &= \frac{RT_{op}}{P_{atm} - p_{gas,H_2O}} \cdot V_{liq} \left(\frac{\rho_{T,8}}{16} + \frac{\rho_{T,9}}{64} + \rho_{T,10} \right) \end{aligned}$$

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):

$$q_{gas} = k_p(P_{gas} - P_{atm})$$

with

$$P_{gas} = p_{gas,h2} + p_{gas,ch4} + p_{gas,co2} + p_{gas,h20}$$

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

$$q_{gas} = k_p(P_{gas} - P_{atm}) \cdot \frac{P_{gas}}{P_{atm}}$$

to obtain the flow rate at atmospheric pressure. Although this compensation factor is included, the two expressions will not yield 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 1e-5).

NOTE! For BSM2 the alternative way (assuming an overpressure in the head space) of calculating the gas flow rate is used.

4 ADM1 DAE implementation

It has been realised that the ODE implementation may be problematic for use in the BSM2 framework. Therefore, 2 DAE versions have been developed (Rosen *et al.* 2006). 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 (DAE$_{pH}$) and a model with algebraic pH and S_{h2} calculations (DAE$_{pH,S_{h2}}$). The second model implementation is motivated by the stiffness problem of the ODE and DAE$_{pH}$ implementations.

4.1 Introduction

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.

4.1.1 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 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 with new data every 15 minutes (Gernaey *et al.* 2005).

4.1.2 Control actions

The BSM2 is a control benchmark. It should, thus, 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.

4.1.3 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 uncorrelated.

4.1.4 Simulating stiff systems in Matlab/Simulink

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 in order to avoid excessively long simulation times, one needs to be somewhat creative when implementing the model.

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 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 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.

4.1.5 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.

4.1.6 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.

4.2 DAE equations

4.2.1 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. An implicit algebraic equation for the pH calculation is given in (Batstone *et al.* 2002) (Table B.3). 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 this case, the Newton-Raphson method used in Volcke *et al.* (2005) 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:

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

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.:

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

The gradient of the algebraic equation $dE(S_{H^+,k})/dS_{H^+}|_{S_{H^+,k}}$ is also needed for calculation of the new state value. Since this expression is rather complicated it is not presented here. The reader is referred to Appendix A for details on the expression. The iteration is repeated as long as $E(S_{H^+,k})$ 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.

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), see Equation 8. The iteration is carried out in the same way as for the S_{H^+} calculation,

this time using

$$E(S_{h2,k}) = \frac{q_{in}}{V_{liq}}(S_{h2,in} - S_{h2,k}) + (1 - Y_{su})\, f_{h2,su}\rho_5 + (1 - Y_{aa})\, f_{h2,aa}\rho_6 + (1 - Y_{fa})\, 0.3\rho_7 + (1 - Y_{c4})\, 0.15\rho_8 + (1 - Y_{c4})\, 0.2\rho_9 + (1 - Y_{pro})\, 0.43\rho_{10} - \rho_{12} - \rho_{T,8}$$

and the gradient of $E(S_{h2,k+1})$. The expression of the gradient is quite complex and the reader is referred to Appendix A for details. 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).

5 Comparison between ODE and DAE implementations

5.1 Introduction

In order to verify the DAE implementation suggested here, it is compared with the ODE implementation. In steady state, the differences should be very small (close to machine precision). However, the dynamic errors must be studied closer – both for the BSM2 anticipated operating region as well as for other possible operational regions.

5.2 Steady-state comparison

The three model implementations discussed in this report, i.e. ODE, DAE with only pH solver and DAE with pH solver and S_{h2} solver, were simulated for 200 days to reach steady state. Both relative and absolute errors were investigated using the ODE simulation as a reference. Only minor errors were encountered – the largest relative errors in the range of 10^{-6}. The largest absolute errors, 10^{-5}, were found in the states with large steady-state values (no scaling of states in the implementations). The result is not surprising since the difference between the models is in the dynamic description of the equations.

5.3 Dynamic comparison

To test the dynamic differences between the model implementations in the BSM2 operational region, a preliminary version of the BSM2 is run to create sensible input data for the digester. The simulation period is 50 days and includes recycling streams from the digester. Thus, the digester is included in the model when data is produced. The input data are stored at 15 minute sampling intervals.

To evaluate the dynamic differences, the digester model is simulated alone with the stored input data as input for the whole duration. The last 7 days are used for evaluation be means of calculating the mean absolute relative error of each variable using the ODE implementation as reference. The main difference between DAE1 and DAE2 is in state variable 8, i.e. the hydrogen state. The mean error is slightly larger than 0.01% which must be considered as acceptable. Especially, since all the other state errors (perhaps except state 23 X_{h2}) are more or less equal.

The fact that the mean errors are in the range of 1e-4 when each sample time is investigated independently does not mean that the relative error of the mean value of each state variable is in the same range. The most deviating variable is the gas flow rate, which is not surprising for the reasons discussed earlier in conjunction with the choice of gas flow rate expression. Also here, the DAE1 and DAE2 behaves similar in terms of dynamic errors.

NOTE! The choice between ODE, DAE1 and DAE2 is up to the user. If acceptable computation times can be achieved with ODE or DAE1 implementations there is no other advantage to use DAE2. But for Matlab/Simulink it seems that with currently available solvers, DAE2 is the only feasible choice.

6 ADM1 benchmark model parameters

Stoichiometric 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: $1 - f_{ch,xc} - f_{pr,xc} - f_{sI,xc} - f_{xI,xc} - f_{li,xc} = 0$
N_{xc}	0.0376/14	kmole N (kg COD)$^{-1}$	1, 13-19	to maintain N mass balance for disintegration
N_I	0.06/14	kmole N (kg COD)$^{-1}$	1	6% on weight basis in benchmark ASM
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. 10
C_{ch}	0.0313	kmole C (kg COD)$^{-1}$	1,2	C_{14} in Eq. 10
C_{pr}	0.03	kmole C (kg COD)$^{-1}$	1,3	C_{15} in Eq. 10
C_{li}	0.022	kmole C (kg COD)$^{-1}$	1,4	C_{16} in Eq. 10
C_{xI}	0.03	kmole C (kg COD)$^{-1}$	1	C_{24} in Eq. 10
C_{su}	0.0313	kmole C (kg COD)$^{-1}$	2,5	C_1 in Eq. 10
C_{aa}	0.03	kmole C (kg COD)$^{-1}$	3,6	C_2 in Eq. 10
$f_{fa,li}$	0.95	–	4	
C_{fa}	0.0217	kmole C (kg COD)$^{-1}$	4,7	C_3 in Eq. 10
$f_{h2,su}$	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 benchmark ASM
C_{bu}	0.025	kmole C (kg COD)$^{-1}$	5,6,9	C_5 in Eq. 10
C_{pro}	0.0268	kmole C (kg COD)$^{-1}$	5,6,8,10	C_6 in Eq. 10
C_{ac}	0.0313	kmole C (kg COD)$^{-1}$	5-11	C_7 in Eq. 10
C_{bac}	0.0313	kmole C (kg COD)$^{-1}$	5-19	C_{17-23} in Eq. 10
Y_{su}	0.1	–	5	kg 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. 10
Y_{aa}	0.08	–	6	kg COD_x (kg COD_s)$^{-1}$
Y_{fa}	0.06	–	7	kg COD_x (kg COD_s)$^{-1}$
Y_{c4}	0.06	–	8,9	kg COD_x (kg COD_s)$^{-1}$
Y_{pro}	0.04	–	10	kg 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	kg COD_x (kg COD_s)$^{-1}$
Y_{h2}	0.06	–	12	kg COD_s (kg COD_x)$^{-1}$

-Note that C_{h2} and C_{IN}, i.e. C_8 and C_{11}, are equal to zero in Equation 10.

Biochemical parameter values

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	
$K_{S,IN}$	1e-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_{5-10}
$pH_{LL,aa}$	4	–	5-10	in I_{5-10}
$k_{m,aa}$	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	
$K_{Ih2,fa}$	5e-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	
$K_{Ih2,c4}$	1e-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	
$K_{Ih2,pro}$	3.5e-6	kg COD m^{-3}	10	in I_{10}
$k_{m,ac}$	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	
$K_{S,h2}$	7e-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}	19	

The unit M is defined as kmole m^{-3} according to Batstone *et al.* (2002).

Physiochemical parameter values

Parameter	Value	Unit	Comment
R	0.083145	bar M^{-1} K^{-1}	
T_{base}	298.15	K	
T_{op}	308.15	K	user defined
K_w	$\exp\left(\frac{55900}{R*100} * \left(\frac{1}{T_{base}} - \frac{1}{T_{op}}\right)\right)$	M 10^{-14}	$\approx$ 2.08e-14
$K_{a,va}$	$10^{-4.86}$	M	$\approx$ 1.38e-5
$K_{a,bu}$	$10^{-4.82}$	M	$\approx$ 1.5e-5
$K_{a,pro}$	$10^{-4.88}$	M	$\approx$ 1.32e-5
$K_{a,ac}$	$10^{-4.76}$	M	$\approx$ 1.74e-5
$K_{a,co2}$	$10^{-6.35}\exp\left(\frac{7646}{R*100}\left(\frac{1}{T_{base}} - \frac{1}{T_{op}}\right)\right)$	M	$\approx$ 4.94e-7
$K_{a,IN}$	$10^{-9.25}\exp\left(\frac{51965}{R*100}\left(\frac{1}{T_{base}} - \frac{1}{Top}\right)\right)$	M	$\approx$ 1.11e-9
$k_{A,Bva}$	1e10	M^{-1} d^{-1}	set to be at least three
$k_{A,Bbu}$	1e10	M^{-1} d^{-1}	orders of magnitude higher
$k_{A,Bpro}$	1e10	M^{-1} d^{-1}	than the fastest time
$k_{A,Bac}$	1e10	M^{-1} d^{-1}	constant of the system
$k_{A,Bco2}$	1e10	M^{-1} d^{-1}	
$k_{A,BIN}$	1e10	M^{-1} d^{-1}	
P_{atm}	1.013	bar	
$p_{gas,h2o}$	$0.0313\exp\left(5290\left(\frac{1}{T_{base}} - \frac{1}{T_{op}}\right)\right)$	bar	$\approx$ 0.0557
k_p	5e4	m^3 d^{-1} bar^{-1}	
k_La	200	d^{-1}	
$K_{H,co2}$	$0.035\exp\left(\frac{-19410}{R*100}\left(\frac{1}{T_{base}} - \frac{1}{T_{op}}\right)\right)$	M_{liq} bar^{-1}	$\approx$ 0.0271
$K_{H,ch4}$	$0.0014\exp\left(\frac{-14240}{R*100}\left(\frac{1}{T_{base}} - \frac{1}{T_{op}}\right)\right)$	M_{liq} bar^{-1}	$\approx$ 0.00116
$K_{H,h2}$	$7.8e\text{-}4\exp\left(\frac{-4180}{R*100}\left(\frac{1}{T_{base}} - \frac{1}{T_{op}}\right)\right)$	M_{liq} bar^{-1}	$\approx$ 7.38e-4

Physical parameter values

Parameter	Value	Unit	Comment
V_{liq}	3400	m^3	
V_{gas}	300	m^3	

7 ADM1 benchmark model steady-state results

In this section, the results of a steady state simulation are given. It should serve as a starting point for verification of any model implementation according to the description given in this report. The inputs may not be completely realistic for all variables but have been chosen so that every input is active (i.e. non-zero) thereby allowing all internal modes of the ADM model to be excited. Note that the resulting pH in the AD is about 7.5, i.e. the pH-inhibition functions are not really active in this test case. The retention time is 20 days and the operating temperature 35 řC. As a part of the BSM TG work model interfaces for connecting ASM1 and ADM1 (and vice versa) have been developed (although not presented here). The AS to AD interface fractionates the incoming COD into inerts, carbohydrates, proteins and lipids rather than putting the incoming COD into the AD as composite material. Consequently, the composite material value is low and the values of the other input variables are comparably high. The influent TSS concentration is about 4.5%. The produced gas contains about 61% methane and 34% carbon dioxide (the rest is mainly water vapour). For a more thorough dynamic verification, further information is required.

Steady-state input variable values

State No.	Variable	Value	Unit
1	$S_{su,in}$	0.01	kg COD m^{-3}
2	$S_{aa,in}$	0.001	kg COD m^{-3}
3	$S_{fa,in}$	0.001	kg COD m^{-3}
4	$S_{va,in}$	0.001	kg COD m^{-3}
5	$S_{bu,in}$	0.001	kg COD m^{-3}
6	$S_{pro,in}$	0.001	kg COD m^{-3}
7	$S_{ac,in}$	0.001	kg COD m^{-3}
8	$S_{h2,in}$	1.0E-8	kg COD m^{-3}
9	$S_{ch4,in}$	1.0E-5	kg COD m^{-3}
10	$S_{IC,in}$	0.04	kmole C m^{-3}
11	$S_{IN,in}$	0.01	kmole N m^{-3}
12	$S_{I,in}$	0.02	kg COD m^{-3}
13	$X_{xc,in}$	2.0	kg COD m^{-3}
14	$X_{ch,in}$	5.0	kg COD m^{-3}
15	$X_{pr,in}$	20.0	kg COD m^{-3}
16	$X_{li,in}$	5.0	kg COD m^{-3}
17	$X_{su,in}$	0.0	kg COD m^{-3}
18	$X_{aa,in}$	0.01	kg COD m^{-3}
19	$X_{fa,in}$	0.01	kg COD m^{-3}
20	$X_{c4,in}$	0.01	kg COD m^{-3}
21	$X_{pro,in}$	0.01	kg COD m^{-3}
22	$X_{ac,in}$	0.01	kg COD m^{-3}
23	$X_{h2,in}$	0.01	kg COD m^{-3}
24	$X_{I,in}$	25.0	kg COD m^{-3}
25	$S_{cat,in}$	0.04	kmole m^{-3}
26	$S_{an,in}$	0.02	kmole m^{-3}
–	q_{in}	170.0	$m^3 d^{-1}$
–	T_{op}	35.0	°C

Steady-state output variable values

State No.	Variable	Value	Unit
1	S_{su}	0.0119548297170	kg COD m^{-3}
2	S_{aa}	0.0053147401716	kg COD m^{-3}
3	S_{fa}	0.0986214009308	kg COD m^{-3}
4	S_{va}	0.0116250064639	kg COD m^{-3}
5	S_{bu}	0.0132507296663	kg COD m^{-3}
6	S_{pro}	0.0157836662845	kg COD m^{-3}
7	S_{ac}	0.1976297169375	kg COD m^{-3}
8	S_{h2}	0.0000002359451	kg COD m^{-3}
9	S_{ch4}	0.0550887764460	kg COD m^{-3}
10	S_{IC}	0.1526778706263	kmole C m^{-3}
11	S_{IN}	0.1302298158037	kmole N m^{-3}
12	S_I	0.3286976637215	kg COD m^{-3}
13	X_{xc}	0.3086976637215	kg COD m^{-3}
14	X_{ch}	0.0279472404350	kg COD m^{-3}
15	X_{pr}	0.1025741061067	kg COD m^{-3}
16	X_{li}	0.0294830497073	kg COD m^{-3}
17	X_{su}	0.4201659824546	kg COD m^{-3}
18	X_{aa}	1.1791717989237	kg COD m^{-3}
19	X_{fa}	0.2430353447194	kg COD m^{-3}
20	X_{c4}	0.4319211056360	kg COD m^{-3}
21	X_{pro}	0.1373059089340	kg COD m^{-3}
22	X_{ac}	0.7605626583132	kg COD m^{-3}
23	X_{h2}	0.3170229533613	kg COD m^{-3}
24	X_I	25.6173953274430	kg COD m^{-3}
25	S_{cat}	0.0400000000000	kmole m^{-3}
26	S_{an}	0.0200000000000	kmole m^{-3}
–	Q	170.000000000000	$m^{-3} d^{-1}$
–	T_{op}	35.0000000000000	°C
–	pH	7.4655377698929	
–	S_{H+}	0.0000000342344	kmole H^+ m^{-3}
27	S_{va^-}	0.0115962470726	kg COD m^{-3}
28	S_{bu^-}	0.0132208262485	kg COD m^{-3}
29	S_{pro^-}	0.0157427831916	kg COD m^{-3}
30	S_{ac^-}	0.1972411554365	kg COD m^{-3}
31	S_{hco3^-}	0.1427774793921	kmole C m^{-3}
–	S_{co2}	0.0099003912343	kmole C m^{-3}
32	S_{nh3}	0.0040909284584	kmole N m^{-3}
–	S_{nh4+}	0.1261388873452	kmole N m^{-3}
33	$S_{gas,h2}$	0.0000102410356	kg COD m^{-3}
34	$S_{gas,ch4}$	1.6256072099814	kg COD m^{-3}
35	$S_{gas,co2}$	0.0141505346784	kmole C m^{-3}
–	$p_{gas,h2}$	0.0000163991826	bar
–	$p_{gas,ch4}$	0.6507796328232	bar
–	$p_{gas,co2}$	0.3625527133281	bar
–	p_{gas}	1.0690164904089	bar
–	q_{gas}	2955.70345419378	$Nm^3 d^{-1}$

Note that the gas flow, q_{gas}, is the flow rate at atmospheric pressure and **not** at the pressure of the head space.

References

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*. ISBN 92-894-1658-0, Office for Official Publications of the European Communities, Luxembourg.

Gernaey, K.V., Rosen, C., Benedetti, L. and Jeppsson, U. (2005). Phenomenological modelling of wastewater treatment plant influent disturbances scenarios. *10th International Conference on Urban Drainage (10ICUD)* 21-26 August, Copenhagen, Denmark.

Gernaey, K.V., Rosen, C and Jeppsson, U. (2006). WWTP dynamic disturbance modelling - an eesential module for long-term benchmarking development. *Wat. Sci. Tech.* 53(4-5), 225-234.

Henze, M., Gujer, W., Mino, T. and van Loosdrecht, M. (2000). Activated sludge models AMS1, ASM2, ASM2d and ASM3. *IWA STR No. 9*, IWA Publishing, London, UK.

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.* 51(1), 287-295.

Rieger, L., Alex, J., Winkler, S., Boehler, M., Thomann, M. and Siegrist, H. (2003). Towards a benchmark simulation model for plant-wide control strategy performance evaluation of WWTPs. *Wat. Sci. Tech.* 47(2), 103-112.

Rosen, C., Vrecko, D., Gernaey, K.V., Pons, M.N. and Jeppsson, U. (2006). Implementing ADM1 for plant-wide benchmark simulations in Matiab/Simulink. *Wat. Sci. Tech.* 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.

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

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.

A Code for DAE implementation in Matlab/Simulink

In this appendix, the solvers for pH and $Sh2$ used in the Lund implementation of ADM1 are presented. The solver files are written in C for Matlab/Simulink S-function utility. If used on this platform, they should work just as they are presented below. If another platform is used, the reader should focus on the functions `Equ` and `gradEqu` for calculation of the equations $E(S_X)$ and $dE(S_X)/dS_X|_{S_{X,k}}$, respectively, and the functions `pHsolver` and `Sh2solver`, respectively, for the iteration (the Newton-Raphson algorithm) to find the solution S_X.

A.1 C-file for pH solver

```
/*
 * pHsolv.c is a C-file S-function level 2 that calculates
 * the algebraic equations for pH and ion states of the ADM1 model.
 * This solver is based on the implementation proposed by Dr Eveline
 * Volcke, BIOMATH, Ghent University, Belgium.
 * Computational speed could be further enhanced by sending all
 * parameters directly from the adm1 module
 * instead of recalculating them within this module.
 *
 * Copyright (2006):
 * Dr Christian Rosen, Dr Darko Vrecko and Dr Ulf Jeppsson
 * Dept. Industrial Electrical Engineering and Automation (IEA)
 * Lund University, Sweden
 * http://www.iea.lth.se/
 */
#define S_FUNCTION_NAME pHsolv
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
#include <math.h>
#define XINIT(S) ssGetSFcnParam(S,0)
#define PAR(S) ssGetSFcnParam(S,1)

/*
 * mdlInitializeSizes:
 * The sizes information is used by Simulink to determine the S-function
 * block's characteristics (number of inputs, outputs, states, etc.).
 */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 2); /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
    /* Return if number of expected != number of actual parameters */
    return;
}
    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 7);
    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 51); /*(S, port index, port width)*/
    ssSetInputPortDirectFeedThrough(S, 0, 0);
    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 7);
    ssSetNumSampleTimes(S, 1);
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}
```

```
/*
 * mdlInitializeSampleTimes:
 * This function is used to specify the sample time(s) for your
 * S-function. You must register the same number of sample times as
 * specified in ssSetNumSampleTimes.
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    /* executes whenever driving block executes */
    ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS /* Change to #undef to remove function */
#if defined(MDL_INITIALIZE_CONDITIONS)
/* mdlInitializeConditions:
 * In this function, you should initialize the continuous and discrete
 * states for your S-function block. The initial states are placed
 * in the state vector, ssGetContStates(S) or ssGetRealDiscStates(S).
 * You can also perform any other initialization activities that your
 * S-function may require. Note, this routine will be called at the
 * start of simulation and if it is present in an enabled subsystem
 * configured to reset states, it will be call when the enabled subsystem
 * restarts execution to reset the states.
 */
static void mdlInitializeConditions(SimStruct *S)
{
    real_T *x0 = ssGetDiscStates(S); /* x0 is pointer */
    int_T i;

    for (i = 0;i < 7; i++) {
        x0[i] = mxGetPr(XINIT(S))[i];
    }
}
#endif /* MDL_INITIALIZE_CONDITIONS */

#undef MDL_START /* Change to #undef to remove function */
#if defined(MDL_START)
/* mdlStart:
 * This function is called once at start of model execution. If you
 * have states that should be initialized once, this is the place
 * to do it.
 */
static void mdlStart(SimStruct *S)
{
}
#endif /* MDL_START */

/*
 * mdlOutputs
 * In this function, you compute the outputs of your S-function
 * block. Generally outputs are placed in the output vector, ssGetY(S).
 */
```

```
static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T *y = ssGetOutputPortRealSignal(S,0);
    real_T *x = ssGetDiscStates(S);
    int_T i;

    for (i = 0; i < 7; i++) {
        y[i] = x[i]; /* state variables are passed on as output variables */
    }
}

static real_T Equ(SimStruct *S)
{
     real_T *x = ssGetDiscStates(S);
     InputRealPtrsType u = ssGetInputPortRealSignalPtrs(S,0);

     static real_T K_w, pK_w_base, K_a_va, pK_a_va_base, K_a_bu, pK_a_bu_base,
K_a_pro, pK_a_pro_base, K_a_ac, pK_a_ac_base, K_a_co2, pK_a_co2_base,
K_a_IN,pK_a_IN_base, T_base, T_op, R, factor;

     R = mxGetPr(PAR(S))[77];
     T_base = mxGetPr(PAR(S))[78];
     T_op = mxGetPr(PAR(S))[79];
     pK_w_base = mxGetPr(PAR(S))[80];
     pK_a_va_base = mxGetPr(PAR(S))[81];
     pK_a_bu_base = mxGetPr(PAR(S))[82];
     pK_a_pro_base = mxGetPr(PAR(S))[83];
     pK_a_ac_base = mxGetPr(PAR(S))[84];
     pK_a_co2_base = mxGetPr(PAR(S))[85];
     pK_a_IN_base = mxGetPr(PAR(S))[86];

     factor = (1.0/T_base - 1.0/T_op)/(100.0*R);
     K_w = pow(10,-pK_w_base)*exp(55900.0*factor);
/* T adjustment for K_w */
     K_a_va = pow(10,-pK_a_va_base);
     K_a_bu = pow(10,-pK_a_bu_base);
     K_a_pro = pow(10,-pK_a_pro_base);
     K_a_ac = pow(10,-pK_a_ac_base);
     K_a_co2 = pow(10,-pK_a_co2_base)*exp(7646.0*factor);
/* T adjustment for K_a_co2 */
     K_a_IN = pow(10,-pK_a_IN_base)*exp(51965.0*factor);
/* T adjustment for K_a_IN */

     x[1] = K_a_va**u[3]/(K_a_va+x[0]);   /* Sva- */
     x[2] = K_a_bu**u[4]/(K_a_bu+x[0]);   /* Sbu- */
     x[3] = K_a_pro**u[5]/(K_a_pro+x[0]); /* Spro- */
     x[4] = K_a_ac**u[6]/(K_a_ac+x[0]);   /* Sac- */
     x[5] = K_a_co2**u[9]/(K_a_co2+x[0]); /* SHCO3- */
     x[6] = K_a_IN**u[10]/(K_a_IN+x[0]);  /* SNH3 */

     return *u[24]+(*u[10]-x[6])+x[0]-x[5]-x[4]/64-x[3]/112-
x[2]/160-x[1]/208-K_w/x[0]-*u[25]; /* SH+ equation */
}
```

```
static real_T gradEqu(SimStruct *S)
{
    real_T *x = ssGetDiscStates(S);
    InputRealPtrsType u = ssGetInputPortRealSignalPtrs(S,0);

    static real_T K_w, pK_w_base, K_a_va, pK_a_va_base, K_a_bu,
pK_a_bu_base, K_a_pro, pK_a_pro_base, K_a_ac, pK_a_ac_base, K_a_co2,
pK_a_co2_base, K_a_IN, pK_a_IN_base, T_base, T_op, R, factor;

     R = mxGetPr(PAR(S))[77];
     T_base = mxGetPr(PAR(S))[78];
     T_op = mxGetPr(PAR(S))[79];
     pK_w_base = mxGetPr(PAR(S))[80];
     pK_a_va_base = mxGetPr(PAR(S))[81];
     pK_a_bu_base = mxGetPr(PAR(S))[82];
     pK_a_pro_base = mxGetPr(PAR(S))[83];
     pK_a_ac_base = mxGetPr(PAR(S))[84];
     pK_a_co2_base = mxGetPr(PAR(S))[85];
     pK_a_IN_base = mxGetPr(PAR(S))[86];

     factor = (1.0/T_base - 1.0/T_op)/(100.0*R);
     K_w = pow(10,-pK_w_base)*exp(55900.0*factor); /* T adjustment for K_w */
     K_a_va = pow(10,-pK_a_va_base);
     K_a_bu = pow(10,-pK_a_bu_base);
     K_a_pro = pow(10,-pK_a_pro_base);
     K_a_ac = pow(10,-pK_a_ac_base);
     K_a_co2 = pow(10,-pK_a_co2_base)*exp(7646.0*factor);
/* T adjustment for K_a_co2 */
     K_a_IN = pow(10,-pK_a_IN_base)*exp(51965.0*factor);
/* T adjustment for K_a_IN */

    return 1+K_a_IN**u[10]/((K_a_IN+x[0])*(K_a_IN+x[0]))
/* Gradient of SH+ equation */
           +K_a_co2**u[9]/((K_a_co2+x[0])*(K_a_co2+x[0]))
           +1/64.0*K_a_ac**u[6]/((K_a_ac+x[0])*(K_a_ac+x[0]))
           +1/112.0*K_a_pro**u[5]/((K_a_pro+x[0])*(K_a_pro+x[0]))
           +1/160.0*K_a_bu**u[4]/((K_a_bu+x[0])*(K_a_bu+x[0]))
           +1/208.0*K_a_va**u[3]/((K_a_va+x[0])*(K_a_va+x[0]))
           +K_w/(x[0]*x[0]);
}

static void pHsolver(SimStruct *S)
{
    real_T *x = ssGetDiscStates(S);
    static real_T delta;
    static real_T S_H_ion0;
    static int_T i;
    static const real_T TOL = 1e-12;
    static const real_T MaxSteps= 1000;

    S_H_ion0 = x[0]; /* SH+ */
    i = 1;
    delta = 1.0;
```

```
    /* Newton-Raphson algorithm */
    while ( (delta > TOL || delta < -TOL) && (i <= MaxSteps) ) {
        delta = Equ(S);
        x[0] = S_H_ion0 - delta/gradEqu(S); /* Calculate the new SH+ */

        if (x[0] <= 0) {
            x[0] = 1e-12; /* to avoid numerical problems */
        }
        S_H_ion0 = x[0];
        ++i;
    }
}

#define MDL_UPDATE /* Change to #undef to remove function */
#if defined(MDL_UPDATE)
/*
 * mdlUpdate:
 * This function is called once for every major integration time step.
 * Discrete states are typically updated here, but this function is useful
 * for performing any tasks that should only take place once per
 * integration step.
 */
static void mdlUpdate(SimStruct *S, int_T tid)
{
   pHsolver(S);
}
#endif /* MDL_UPDATE */

#undef MDL_DERIVATIVES /* Change to #undef to remove function */
#if defined(MDL_DERIVATIVES)
/*
 * mdlDerivatives:
 * In this function, you compute the S-function block's derivatives.
 * The derivatives are placed in the derivative vector, ssGetdX(S).
 */
static void mdlDerivatives(SimStruct *S)
{
}
#endif /* MDL_DERIVATIVES */

/*
 * mdlTerminate:
 * In this function, you should perform any actions that are necessary
 * at the termination of a simulation. For example, if memory was
 * allocated in mdlStart, this is the place to free it.
 */
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
```

A.2 C-file for S_{h2} solver

```
/*
 * Sh2solv.c is a C-file S-function level 2 that solves the algebraic
 * equation for Sh2 of the ADM1 model,thereby reducing the stiffness of the
 * system considerably (if used together with a pHsolver).
 *
 * Copyright (2006):
 * Dr Christian Rosen, Dr Darko Vrecko and Dr Ulf Jeppsson
 * Dept. Industrial Electrical Engineering and Automation (IEA)
 * Lund University, Sweden
 * http://www.iea.lth.se/
 */

#define S_FUNCTION_NAME Sh2solv
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
#include <math.h>
#define XINIT(S) ssGetSFcnParam(S,0)
#define PAR(S) ssGetSFcnParam(S,1)
#define V(S)      ssGetSFcnParam(S,2)

/*
 * mdlInitializeSizes:
 * The sizes information is used by Simulink to determine the S-function
 * block's characteristics (number of inputs, outputs, states, etc.).
 */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 3); /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
    /* Return if number of expected != number of actual parameters */
    return;
}
    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 1);
    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 52); /*(S, port index, port width)*/
    ssSetInputPortDirectFeedThrough(S, 0, 0);
    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 1);
    ssSetNumSampleTimes(S, 1);
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}

/*
 * mdlInitializeSampleTimes:
 * This function is used to specify the sample time(s) for your
 * S-function. You must register the same number of sample times as
 * specified in ssSetNumSampleTimes.
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
```

```
    /* executes whenever driving block executes */
    ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS /* Change to #undef to remove function */
#if defined(MDL_INITIALIZE_CONDITIONS)
/* mdlInitializeConditions:
 * In this function, you should initialize the continuous and discrete
 * states for your S-function block. The initial states are placed
 * in the state vector, ssGetContStates(S) or ssGetRealDiscStates(S).
 * You can also perform any other initialization activities that your
 * S-function may require. Note, this routine will be called at the
 * start of simulation and if it is present in an enabled subsystem
 * configured to reset states, it will be call when the enabled subsystem
 * restarts execution to reset the states.
 */
static void mdlInitializeConditions(SimStruct *S)
{
    real_T *x0 = ssGetDiscStates(S); /*x0 is pointer*/

    x0[0] = mxGetPr(XINIT(S))[0];
}
#endif /* MDL_INITIALIZE_CONDITIONS */


#undef MDL_START /* Change to #undef to remove function */
#if defined(MDL_START)
/* mdlStart:
 * This function is called once at start of model execution. If you
 * have states that should be initialized once, this is the place
 * to do it.
 */
static void mdlStart(SimStruct *S)
{
}
#endif /* MDL_START */


/*
 * mdlOutputs
 * In this function, you compute the outputs of your S-function
 * block. Generally outputs are placed in the output vector, ssGetY(S).
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T *y = ssGetOutputPortRealSignal(S,0);
    real_T *x = ssGetDiscStates(S);

    y[0] = x[0]; /* state variable is passed on as output variable */
}


static real_T Equ(SimStruct *S)
{
```

```
    real_T *x = ssGetDiscStates(S);
    InputRealPtrsType u = ssGetInputPortRealSignalPtrs(S,0);

    static real_T eps, f_h2_su, Y_su, f_h2_aa, Y_aa, Y_fa, Y_c4, Y_pro,
K_S_IN, k_m_su, K_S_su, pH_UL_aa, pH_LL_aa, k_m_aa;
    static real_T K_S_aa, k_m_fa, K_S_fa, K_Ih2_fa, k_m_c4, K_S_c4,
K_Ih2_c4, k_m_pro, K_S_pro, K_Ih2_pro;
    static real_T pH_UL_ac, pH_LL_ac, k_m_h2, K_S_h2, pH_UL_h2,
pH_LL_h2, R, T_base, T_op, kLa, K_H_h2, K_H_h2_base, V_liq, pH_op,I_pH_aa;
    static real_T I_pH_h2, I_IN_lim, I_h2_fa, I_h2_c4, I_h2_pro, inhib[6];
    static real_T proc5, proc6, proc7, proc8, proc9, proc10, proc12,
p_gas_h2, procT8, reac8;
    static real_T pHLim_aa, pHLim_h2, a_aa, a_h2, S_H_ion, n_aa, n_h2;

    eps = 0.000001;

    f_h2_su = mxGetPr(PAR(S))[18];
    Y_su = mxGetPr(PAR(S))[27];
    f_h2_aa = mxGetPr(PAR(S))[28];
    Y_aa = mxGetPr(PAR(S))[34];
    Y_fa = mxGetPr(PAR(S))[35];
    Y_c4 = mxGetPr(PAR(S))[36];
    Y_pro = mxGetPr(PAR(S))[37];
    K_S_IN = mxGetPr(PAR(S))[45];
    k_m_su = mxGetPr(PAR(S))[46];
    K_S_su = mxGetPr(PAR(S))[47];
    pH_UL_aa = mxGetPr(PAR(S))[48];
    pH_LL_aa = mxGetPr(PAR(S))[49];
    k_m_aa = mxGetPr(PAR(S))[50];
    K_S_aa = mxGetPr(PAR(S))[51];
    k_m_fa = mxGetPr(PAR(S))[52];
    K_S_fa = mxGetPr(PAR(S))[53];
    K_Ih2_fa = mxGetPr(PAR(S))[54];
    k_m_c4 = mxGetPr(PAR(S))[55];
    K_S_c4 = mxGetPr(PAR(S))[56];
    K_Ih2_c4 = mxGetPr(PAR(S))[57];
    k_m_pro = mxGetPr(PAR(S))[58];
    K_S_pro = mxGetPr(PAR(S))[59];
    K_Ih2_pro = mxGetPr(PAR(S))[60];
    pH_UL_ac = mxGetPr(PAR(S))[64];
    pH_LL_ac = mxGetPr(PAR(S))[65];
    k_m_h2 = mxGetPr(PAR(S))[66];
    K_S_h2 = mxGetPr(PAR(S))[67];
    pH_UL_h2 = mxGetPr(PAR(S))[68];
    pH_LL_h2 = mxGetPr(PAR(S))[69];
    R = mxGetPr(PAR(S))[77];
    T_base = mxGetPr(PAR(S))[78];
    T_op = mxGetPr(PAR(S))[79];
    kLa = mxGetPr(PAR(S))[94];
    K_H_h2_base = mxGetPr(PAR(S))[98];
    V_liq = mxGetPr(V(S))[0];

    K_H_h2 = K_H_h2_base*exp(-4180.0*(1.0/T_base - 1.0/T_op)/(100.0*R));
 /* T adjustment for K_H_h2 */
```

```
     pH_op = *u[33];   /* pH */
     S_H_ion = *u[34]; /* SH+ */

   /* STRs function
   if (pH_op < pH_UL_aa)
    I_pH_aa = exp(-3.0*(pH_op-pH_UL_aa)*(pH_op-pH_UL_aa)/((pH_UL_aa-pH_LL_aa)*
(pH_UL_aa-pH_LL_aa)));
   else
    I_pH_aa = 1.0;
   if (pH_op < pH_UL_h2)
    I_pH_h2 = exp(-3.0*(pH_op-pH_UL_h2)*(pH_op-pH_UL_h2)/((pH_UL_h2-pH_LL_h2)*
(pH_UL_h2-pH_LL_h2)));
   else
    I_pH_h2 = 1.0;
   */

/* Hill function on pH inhibition
pHLim_aa = (pH_UL_aa + pH_LL_aa)/2.0;
pHLim_ac = (pH_UL_ac + pH_LL_ac)/2.0;
pHLim_h2 = (pH_UL_h2 + pH_LL_h2)/2.0;
I_pH_aa = pow(pH_op,24)/(pow(pH_op,24)+pow(pHLim_aa ,24));
I_pH_ac = pow(pH_op,45)/(pow(pH_op,45)+pow(pHLim_ac ,45));
I_pH_h2 = pow(pH_op,45)/(pow(pH_op,45)+pow(pHLim_h2 ,45));
*/

/* MNPs function on pH inhibition, ADM1 Workshop, Copenhagen 2005.
a_aa = 25.0/(pH_UL_aa-pH_LL_aa+eps);
a_ac = 25.0/(pH_UL_ac-pH_LL_ac+eps);
a_h2 = 25.0/(pH_UL_h2-pH_LL_h2+eps);

I_pH_aa = 0.5*(1+tanh(a_aa*(pH_op/pHLim_aa - 1.0)));
I_pH_ac = 0.5*(1+tanh(a_ac*(pH_op/pHLim_ac - 1.0)));
I_pH_h2 = 0.5*(1+tanh(a_h2*(pH_op/pHLim_h2 - 1.0)));
*/

/* Hill function on SH+ used within BSM2, ADM1 Workshop, Copenhagen 2005. */
     pHLim_aa = pow(10,(-(pH_UL_aa + pH_LL_aa)/2.0));
     pHLim_h2 = pow(10,(-(pH_UL_h2 + pH_LL_h2)/2.0));
     n_aa=3.0/(pH_UL_aa-pH_LL_aa);
     n_h2=3.0/(pH_UL_h2-pH_LL_h2);
     I_pH_aa = pow(pHLim_aa,n_aa)/(pow(S_H_ion,n_aa)+pow(pHLim_aa ,n_aa));
     I_pH_h2 = pow(pHLim_h2,n_h2)/(pow(S_H_ion,n_h2)+pow(pHLim_h2 ,n_h2));

     I_IN_lim = 1.0/(1.0+K_S_IN/(*u[10]));
     I_h2_fa = 1.0/(1.0+x[0]/K_Ih2_fa);
     I_h2_c4 = 1.0/(1.0+x[0]/K_Ih2_c4);
     I_h2_pro = 1.0/(1.0+x[0]/K_Ih2_pro);

     inhib[0] = I_pH_aa*I_IN_lim;
     inhib[1] = inhib[0]*I_h2_fa;
     inhib[2] = inhib[0]*I_h2_c4;
     inhib[3] = inhib[0]*I_h2_pro;
     inhib[5] = I_pH_h2*I_IN_lim;
```

```
     proc5 = k_m_su**u[0]/(K_S_su+*u[0])**u[16]*inhib[0];
     proc6 = k_m_aa**u[1]/(K_S_aa+*u[1])**u[17]*inhib[0];
     proc7 = k_m_fa**u[2]/(K_S_fa+*u[2])**u[18]*inhib[1];
     proc8 = k_m_c4**u[3]/(K_S_c4+*u[3])**u[19]**u[3]/(*u[3]+
*u[4]+eps)*inhib[2];
     proc9 = k_m_c4**u[4]/(K_S_c4+*u[4])**u[19]**u[4]/(*u[3]+
*u[4]+eps)*inhib[2];
     proc10 = k_m_pro**u[5]/(K_S_pro+*u[5])**u[20]*inhib[3];

     proc12 = k_m_h2*x[0]/(K_S_h2+x[0])**u[22]*inhib[5];

     p_gas_h2 = *u[43]*R*T_op/16.0;
     procT8 = kLa*(x[0]-16.0*K_H_h2*p_gas_h2);

     reac8 = (1.0-Y_su)*f_h2_su*proc5+(1.0-Y_aa)*f_h2_aa*proc6+
(1.0-Y_fa)*0.3*proc7+(1.0-Y_c4)*0.15*proc8+(1.0-Y_c4)*0.2*proc9+
(1.0-Y_pro)*0.43*proc10-proc12-procT8;

     return 1/V_liq**u[26]*(*u[51]-x[0])+reac8; /* Sh2 equation */
}


static real_T gradEqu(SimStruct *S)
{
     real_T *x = ssGetDiscStates(S);
     InputRealPtrsType u = ssGetInputPortRealSignalPtrs(S,0);

     static real_T eps, f_h2_su, Y_su, f_h2_aa, Y_aa, Y_fa, Y_c4,
Y_pro, K_S_IN, k_m_su, K_S_su, pH_UL_aa, pH_LL_aa, k_m_aa;
     static real_T K_S_aa, k_m_fa, K_S_fa, K_Ih2_fa, k_m_c4,
K_S_c4, K_Ih2_c4, k_m_pro, K_S_pro, K_Ih2_pro;
     static real_T pH_UL_ac, pH_LL_ac, k_m_h2, K_S_h2, pH_UL_h2, pH_LL_h2,
R, T_base, T_op, kLa, K_H_h2, K_H_h2_base, V_liq, pH_op, I_pH_aa, I_pH_h2;
     static real_T pHLim_aa, pHLim_h2, a_aa, a_h2, S_H_ion, n_aa, n_h2;

     eps = 0.000001;

     f_h2_su = mxGetPr(PAR(S))[18];
     Y_su = mxGetPr(PAR(S))[27];
     f_h2_aa = mxGetPr(PAR(S))[28];
     Y_aa = mxGetPr(PAR(S))[34];
     Y_fa = mxGetPr(PAR(S))[35];
     Y_c4 = mxGetPr(PAR(S))[36];
     Y_pro = mxGetPr(PAR(S))[37];
     K_S_IN = mxGetPr(PAR(S))[45];
     k_m_su = mxGetPr(PAR(S))[46];
     K_S_su = mxGetPr(PAR(S))[47];
     pH_UL_aa = mxGetPr(PAR(S))[48];
     pH_LL_aa = mxGetPr(PAR(S))[49];
     k_m_aa = mxGetPr(PAR(S))[50];
     K_S_aa = mxGetPr(PAR(S))[51];
     k_m_fa = mxGetPr(PAR(S))[52];
     K_S_fa = mxGetPr(PAR(S))[53];
```

```
    K_Ih2_fa = mxGetPr(PAR(S))[54];
    k_m_c4 = mxGetPr(PAR(S))[55];
    K_S_c4 = mxGetPr(PAR(S))[56];
    K_Ih2_c4 = mxGetPr(PAR(S))[57];
    k_m_pro = mxGetPr(PAR(S))[58];
    K_S_pro = mxGetPr(PAR(S))[59];
    K_Ih2_pro = mxGetPr(PAR(S))[60];
    pH_UL_ac = mxGetPr(PAR(S))[64];
    pH_LL_ac = mxGetPr(PAR(S))[65];
    k_m_h2 = mxGetPr(PAR(S))[66];
    K_S_h2 = mxGetPr(PAR(S))[67];
    pH_UL_h2 = mxGetPr(PAR(S))[68];
    pH_LL_h2 = mxGetPr(PAR(S))[69];
    R = mxGetPr(PAR(S))[77];
    T_base = mxGetPr(PAR(S))[78];
    T_op = mxGetPr(PAR(S))[79];
    kLa = mxGetPr(PAR(S))[94];
    K_H_h2_base = mxGetPr(PAR(S))[98];
    V_liq = mxGetPr(V(S))[0];

    K_H_h2 = K_H_h2_base*exp(-4180.0*(1.0/T_base - 1.0/T_op)/(100.0*R));
/* T adjustment for K_H_h2 */

    pH_op = *u[33];   /* pH */
    S_H_ion = *u[34]; /* SH+ */

  /* STRs function
  if (pH_op < pH_UL_aa)
   I_pH_aa = exp(-3.0*(pH_op-pH_UL_aa)*(pH_op-pH_UL_aa)/((pH_UL_aa-pH_LL_aa)*
(pH_UL_aa-pH_LL_aa)));
  else
   I_pH_aa = 1.0;
  if (pH_op < pH_UL_h2)
   I_pH_h2 = exp(-3.0*(pH_op-pH_UL_h2)*(pH_op-pH_UL_h2)/((pH_UL_h2-pH_LL_h2)*
(pH_UL_h2-pH_LL_h2)));
  else
   I_pH_h2 = 1.0;
  */

  /* Hill function on pH inhibition
  pHLim_aa = (pH_UL_aa + pH_LL_aa)/2.0;
  pHLim_ac = (pH_UL_ac + pH_LL_ac)/2.0;
  pHLim_h2 = (pH_UL_h2 + pH_LL_h2)/2.0;
  I_pH_aa = pow(pH_op,24)/(pow(pH_op,24)+pow(pHLim_aa ,24));
  I_pH_ac = pow(pH_op,45)/(pow(pH_op,45)+pow(pHLim_ac ,45));
  I_pH_h2 = pow(pH_op,45)/(pow(pH_op,45)+pow(pHLim_h2 ,45));
  */
  /* MNPs function on pH inhibition, ADM1 Workshop, Copenhagen 2005.
  a_aa = 25.0/(pH_UL_aa-pH_LL_aa+eps);
  a_ac = 25.0/(pH_UL_ac-pH_LL_ac+eps);
  a_h2 = 25.0/(pH_UL_h2-pH_LL_h2+eps);

  I_pH_aa = 0.5*(1+tanh(a_aa*(pH_op/pHLim_aa - 1.0)));
  I_pH_ac = 0.5*(1+tanh(a_ac*(pH_op/pHLim_ac - 1.0)));
```

```
   I_pH_h2 = 0.5*(1+tanh(a_h2*(pH_op/pHLim_h2 - 1.0)));
   */
   /* Hill function on SH+ used within BSM2, ADM1 Workshop, Copenhagen 2005. */
   pHLim_aa = pow(10,(-(pH_UL_aa + pH_LL_aa)/2.0));
   pHLim_h2 = pow(10,(-(pH_UL_h2 + pH_LL_h2)/2.0));
   n_aa=3.0/(pH_UL_aa-pH_LL_aa);
   n_h2=3.0/(pH_UL_h2-pH_LL_h2);
   I_pH_aa = pow(pHLim_aa,n_aa)/(pow(S_H_ion,n_aa)+pow(pHLim_aa ,n_aa));
   I_pH_h2 = pow(pHLim_h2,n_h2)/(pow(S_H_ion,n_h2)+pow(pHLim_h2 ,n_h2));

    /* Gradient of Sh2 equation */
    return -1/V_liq**u[26]
            -3.0/10.0*(1-Y_fa)*k_m_fa**u[2]/(K_S_fa+*u[2])*
            *u[18]*I_pH_aa/(1+K_S_IN/(*u[10]))/((1+x[0]/K_Ih2_fa)*
            (1+x[0]/K_Ih2_fa))/K_Ih2_fa -3.0/20.0*(1-Y_c4)*k_m_c4*
            *u[3]**u[3]/(K_S_c4+*u[3])**u[19]/(*u[4]+*u[3]+eps)*
            I_pH_aa/(1+K_S_IN/(*u[10]))/((1+x[0]/K_Ih2_c4)*
            (1+x[0]/K_Ih2_c4))/K_Ih2_c4-1.0/5.0*(1-Y_c4)*k_m_c4**u[4]*
            *u[4]/(K_S_c4+*u[4])**u[19]/(*u[4]+*u[3]+eps)*
            I_pH_aa/(1+K_S_IN/(*u[10]))/((1+x[0]/K_Ih2_c4)*
            (1+x[0]/K_Ih2_c4))/K_Ih2_c4-43.0/100.0*(1-Y_pro)*k_m_pro**u[5]/
            (K_S_pro+*u[5])**u[20]*I_pH_aa/(1+K_S_IN/(*u[10]))/
            ((1+x[0]/K_Ih2_pro)*(1+x[0]/K_Ih2_pro))/K_Ih2_pro
            -k_m_h2/(K_S_h2+x[0])**u[22]*I_pH_h2/(1+K_S_IN/
            (*u[10]))+k_m_h2*x[0]/((K_S_h2+x[0])*(K_S_h2+x[0]))*
            *u[22]*I_pH_h2/(1+K_S_IN/(*u[10]))-kLa;
}

static void Sh2solver(SimStruct *S)
{

    real_T *x = ssGetDiscStates(S);

    static real_T delta;
    static real_T Sh20;
    static int_T i;

    static const real_T TOL = 1e-12;
    static const real_T MaxSteps= 1000;

    Sh20 = x[0]; /* Sh2 */

    i = 1;
    delta = 1.0;

    /* Newton-Raphson algorithm */

    while ( (delta > TOL || delta < -TOL) && (i <= MaxSteps) ) {
        delta = Equ(S);
        x[0] = Sh20-delta/gradEqu(S);     /* Calculate the new Sh2 */

        if (x[0] <= 0) {
            x[0] = 1e-12; /* to avoid numerical problems */
        }
```

```
        Sh20 = x[0];
        ++i;
    }
}


#define MDL_UPDATE /* Change to #undef to remove function */
#if defined(MDL_UPDATE)
/*
 * mdlUpdate:
 * This function is called once for every major integration time step.
 * Discrete states are typically updated here, but this function is useful
 * for performing any tasks that should only take place once per
 * integration step.
 */
static void mdlUpdate(SimStruct *S, int_T tid)
{
   Sh2solver(S);
}
#endif /* MDL_UPDATE */


#undef MDL_DERIVATIVES /* Change to #undef to remove function */
#if defined(MDL_DERIVATIVES)
/*
 * mdlDerivatives:
 * In this function, you compute the S-function block's derivatives.
 * The derivatives are placed in the derivative vector, ssGetdX(S).
 */
static void mdlDerivatives(SimStruct *S)
{
}
#endif /* MDL_DERIVATIVES */


/*
 * mdlTerminate:
 * In this function, you should perform any actions that are necessary
 * at the termination of a simulation. For example, if memory was
 * allocated in mdlStart, this is the place to free it.
 */
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
```