Matlab/Simulink Benchmark Simulation Model No 2 (BSM2)

by

Dr Ulf Jeppsson, IEA, Lund University, Sweden

(version: September 2011)

e-mail: Ulf.Jeppsson@iea.lth.se

Note that this short document does not by any means provide a complete description of how to use the BSM2 system but it may give some useful hints on the topic and avoid some unnecessary frustration. The document assumes that you are familiar with Matlab/Simulink and wastewater treatment process modelling and simulation. No references are given in this document. Also read the provided BSM2_agreement_LU.pdf document.

INTRODUCTION

The benchmark files provided here represent an exact implementation of what is described for the benchmark project (related to the BSM2 system) at

http://www.benchmarkwwtp.org

Note that during the years of development many different modification and updates have been done to BSM2, which means that this final implementation is somewhat different compared to descriptions provided at conferences and published in scientific journals during the years 2004 to 2008. However, this is an exact implementation of the description that will be provided in the (soon to appear) IWA Scientific and Technical Report on Benchmarking and the associated documents therein. You should first study the documents on the web site and the ones included in this distribution carefully so that you realise what this implementation is supposed to do. The provided models should work fine with Matlab 7 and later versions. For reasons of simulation performance the actual models have all been written in C (using Simulink's C-mex principles and S-function capability). You do not need a separate C-compiler, instead just use the built-in Matlab C-compiler (if you do not have it then install a freeware C-compiler (see Matlab's Help system for more information on this)).

UNPACKING THE FILES

The files have been archived using *zip*. Just unzip the file using whatever software you normally use for this purpose.

FILE DESCRIPTIONS

When the files are unpacked you will find in the main directory (*BSM2*) most of the files that are associated with BSM2. In the subdirectory *Results* there is an Excel document providing correct results for the different standard test cases of BSM2 (only the columns related to Matlab/Simulink are completely updated). In the subdirectory *Influent_data* the various influent data files are provided in .mat format and in ascii format (in case you have to regenerate them). Also a noise file (.mat) is provided in case you want to use the standardized predefined noise data for your sensors and actuators. These files are also available for download from the Benchmark home page. Finally, in the directory *risk_calculation* all the required files for performing the calculation of the BSM2 risk index are collected (the Matlab script that does this calculation will expect to find the files in a subdirectory with this name). These are .fis files generated from Matlab's Fuzzy Logics toolbox and will not be further discussed here (see special technical reports for more information).

Five Simulink models are available:

- *bsm2_ol.mdl* – simulate the plant without active control, i.e. in open loop, using dynamic input data;
- *bsm2_ss.mdl* – simulate the plant without active control, i.e. in open loop, using constant input data;
- *bsm2_cl.mdl* – simulate the plant with active control (closed loop) including noise and delays (noise, delays etc. can easily be deactivated by changing a few parameters in the init files);
- *sensor_actuators.mdl* – a provided 'library' of sensor and actuator models that can be used with BSM2, by simply copying them into the BSM2 layout and connecting them in the proper way (the models are based on the defined sensor classes for BSM2). Some of these models are already in use in the standard configuration and some are there for your benefit when implementing control strategies of your own.
- *adm1_versions.mdl* – a provided 'library' containing the three different ADM1 implementations (one ODE and two DAE versions).

bsm2_ss and *bsm2_ol* are almost identical systems but the input data selection and simulation parameters in Simulink have been preset so that one is ready for steady state calculations and the other for dynamic open loop simulations. Moreover, the ADM1 model used in the steady state implementation is based on the ODE implementation and the ADM1 model in the open loop implementation (and in the closed loop implementation) is based on a DAE implementation (DAE2) for reasons of simulation speed.

25 C-files are included in this distribution:

- *adm1_ODE_bsm2.c, adm1_DAE1_bsm2.c, adm1_DAE2_bsm2.c* – three different ADM1 implementations based on either ordinary differential equations (ODE) or differerence algebraic equations. In principle the ODE is used for steady state calculations (solver ode15s) and the DAE2 for dynamic simulations (solver ode45). More information in the special technical report on the ADM1 implementation for BSM2.

- *adm2asm_v3_bsm2.c* and *asm2adm_v3_bsm2.c* – the interface models which enable the connection of the ADM1 to the rest of BSM2 (based on ASM1 variables). More information in the special technical report on the ADM1 and ASM1 interfacing for BSM2.
- *asm1_bsm2.c* – C-file containing the AS Model no. 1;
- *combiner_bsm2.c*, *combiner3_bsm2.c*, *combiner4_bsm2.c* – adds two, three or four separate streams into one based on loads (the standard 21 variables vector).
- *combiner_adm1_bsm2.c*, *DAE1_combiner_bsm2.c*, *DAE2_combiner_bsm2.c* – special combiners used for the different ADM1 implementations (ADM1 has its own set of variables).
- *flowsplitter_bsm2.c* – divides one stream into two.
- *carboncombiner_bsm2.c* – adds the external carbon flow to the rest of the wastewater;
- *dewatering_bsm2.c* – the 'ideal' dewatering process model.
- *hyddelayv3_bsm2.c* – a special delay function (a fast first-order exponential filter) to avoid algebraic loops.
- *pHdelay_bsm2.c* – small delay model for pH used for all ADM implementations when the current pH value of the ADM1 is returned to the ADM2AS interface at each integration step.
- *pHsolv_bsm2.c* – calculates the S_{H+} concentration at each integration step using a Newton-Raphson algorithm. Used by *adm1_DAE1_bsm2.c* and *adm1_DAE2_bsm2.c.*
- *primclar_bsm2.c* – the Otterpohl-Freund primary clarifier model.
- *settler1dv5_bsm2.c* – C-file for a 10-layer one-dimensional settler model;
- *Sh2solv_bsm2.c* – calculates the S_{h2} concentration at each integration step using a Newton-Raphson algorithm. Used only by *adm1_DAE2_bsm2.c.*
- *storage_bsm2.c, storagebypass_bsm2.c, storagedelay_bsm2.c* – three models files that together make up the overall model for the BSM2 reject water storage tank including automatic bypass when full or stop of pumping when becoming empty. The storage tank is a process model but it is also an actuator which can be used to manipulate the behaviour of the BSM2 system.
- *thickener_bsm2.c* – the 'ideal' thickener process model.

These 25 files must be compiled on your local machine using the Matlab *mex* command before you can use the models (use the *mexall_bsm2.m* script). All the above files are predefined for BSM2 and should not be modified when BSM2 is used for its original purpose (objective comparisons of control strategies). If the models are changed then the system does no longer represent a true BSM2.

In principle all parameters and variables are defined in different m-files and the actual Simulink blocks and the models therein never need to be adjusted (unless you want to rebuild the plant layout to something else than BSM2). Within Simulink you only define parameters that are related to the numerical solver, storing of data and selecting which input data file should be used. The initialisation m-files have names associated with the model they influence: *adm1init_bsm2.m, asm1init_bsm2.m, dewateringinit_bsm2.m, hyddelayinit_bsm2.m, primclarinit_bsm2.m, reginit_bsm2.m, sensorinit_bsm2.m, settler1dinit_bsm2.m, storageinit_bsm2.m, thickenerinit_bsm2.m.* As long as you are using the true BSM2 system you do not need to modify anything within the init files associated with the process models. However, it is valuable to study the init files to see how the models are set up. Only

reginit_bsm2.m and *sensorinit_bsm2.m* will require modifications when you add new sensors (basic set up for all sensor classes is given in *sensorinit_bsm2.m* – you only need to add the values and activate the code) or new controllers. In *sensorinit_bsm2.m* you can also easily deactivate noise for sensors and actuators, chose a different noise source (noise generator or read from file) or make sensors and actuators completely ideal. Parameter values for the controllers are defined in *reginit_bsm2.m*.

All init files are called from the main BSM2 initialisation script *init_bsm2.m*, in which also the input data files are loaded. Moreover, here you select which temperature propagation model to use and if you want to activate the five dummy variables.

Three input data files are provided in the subdirectory *Influent_data*:

- *constinfluent_bsm2.mat* – the constant value influent file, which actually represents the average vales for one full year of dynamic data (also available as .txt file);
- *dyninfluent_bsm2.mat* – full dynamic influent data file for 609 days (15 min samples, also available as .txt file);
- *sensornoise_bsm2.mat* – a predefined noise file (first column = time and 25 columns of noise) that can be used for sensor noise when exact comparisons of different simulation platforms are required. Note that the noise file only contains data for the first 14 days (new value every minute) and must be cyclically repeated to produce data for full 609 days of simulations. A small script *create_noisematrix.m* is available in the directory Influent_data, which can be used to create a full 609-days noisefile. The new file will be stored in the Influent_data directory and loaded the next time init_bsm2 is run The original noise file is stored as *sensornoise_org_bsm2.mat*.

If your Matlab version cannot load the .mat files then load the general .txt files and then save them again into binary format (.mat) using Matlab's save command. The initialisation script for BSM2 expects to find the input files as .mat files in the subdirectory *Influent_data*. The two influent data files contain 22 columns of variables in the following order:

1. *time (d);*
2. S_I *(inert soluble material, g COD.m^{-3});*
3. S_S *(readily biodegradable substrate, g COD.m^{-3});*
4. X_I *(inert particulate material, g COD.m^{-3});*
5. X_S *(slowly biodegradable substrate, g COD.m^{-3});*
6. $X_{B,H}$ *(heterotrophic biomass, g COD.m^{-3});*
7. $X_{B,A}$ *(autotrophic biomass, g COD.m^{-3});*
8. X_P *(inert particulate material from biomass decay, g COD.m^{-3});*
9. S_O *(dissolved oxygen, g (-COD).m^{-3});*
10. S_{NO} *(nitrate and nitrite, g N.m^{-3});*
11. S_{NH} *(ammonia and ammonium, g N.m^{-3});*
12. S_{ND} *(soluble organic nitrogen associated with S_S, g N.m^{-3});*
13. X_{ND} *(particulate organic nitrogen associated with X_S, g N.m^{-3});*
14. S_{ALK} *(alkalinity);*
15. *TSS (total suspended solids, g SS.m^{-3});*
16. *flow rate, m^3.d^{-1};*
17. *temperature, °C;*

18. *dummy variable (soluble) no 1;*
19. *dummy variable (soluble) no 2;*
20. *dummy variable (soluble) no 3;*
21. *dummy variable (particulate) no 1;*
22. *dummy variable (particulate) no 2.*

The output variables are all stored during simulations in the same order in Matlab workspace. Note that time is not stored together with all the other variables so there is a one-step-shift in the order (S_I being number 1 etc.). The output time information is stored as a general individual variable in Matlab workspace called *t* (unit days). In principle all inputs and outputs to all processes, controllers, sensors etc are stored during a simulation. The names of all these variables are found directly in the BSM2 Simulink graphical layout. By default data are stored as grab sample values every 15 minutes.

Note that the dummy variables data are all set to zero and are used primarily in the BSM1_LT system. However, the models in this BSM2 implementation have all been prepared for the use of three extra soluble and two extra particulate variables as inputs for easy extension of the system without having to rewrite the C-code completely.

For the purpose of evaluating a simulated control strategy three files exist:

- *perf_plant_bsm2.m* – prints to the screen the values of all performance criteria related to the overall plant and plots important variables according to the detailed definitions provided in a special technical report. It also calls the risk calculation module.
- *perf_risk_bsm2.m* – prints to the screen the values of all performance criteria related to risk and plots important variables according to the detailed definitions provided in a special technical report. Called from *perf_plant_bsm2.m.* Note that at this time the Matlab fuzzy toolbox is required to run *perf_risk_bsm2.m* (work is on-going in order to avoid this).
- *perf_controller_bsm2.m* – similar as *perf_plant_bsm2* but prints to the screen the performance criteria of all the controllers and control handles according to the detailed definitions provided in a special technical report.

These three files may appear quite complicated and it is important that you understand how they work. The *perf_risk_bsm2.m* would normally not require modifications but depending on what new control strategies you implement the *perf_plant_bsm2.m* may require and *perf_controller_bsm2.m* will require that you modify the code. Obviously this must be done in accordance to BSM2 specifications otherwise your evaluation will not be valid. Therefore you must understand the code and use the same principles for any modifications. The existing code will be a great help for any modifications you do. For the default test cases the evaluation works properly but if you modify names of stored output variables then of course they will not work. If you do not want to evaluate the BSM2 for the default time period (days 245 to 609) then you simply need to modify the two variables start time and stop time in the files. The evaluations are based on that workspace data are available for all the needed variables (with correct names) and stored every 15 minutes for the period of evaluation.

Finally there are three more files: *printadmresults.m, stateset_bsm2.m* and *statevalues_bsm2.m*. Their purposes are:

- *statevalues_bsm2.m* – this script will print on the screen a list of all values for all state variables for the last time sample of the latest simulation. It uses also the script *printadmresults.m*.
- *stateset_bsm2.m* – this script saves all the process variables that are printed by *statevalues_bsm2* into a file called *states_bsm2.mat* and also sets them in Matlab workspace using the proper names so that the BSM2 is initialised and ready for a new simulation (this is very useful when you want to rerun simulations for different cases and want to be sure that you always start from the same initial values, then you just load the *states_bsm2.mat* file and starts the simulation). Obviously you can also use *stateset_bsm2* for 'remembering' the final states of any dynamic simulations and continue another simulation where the first one left off. It works with all three BSM2 Simulink implementations.

Naturally you can create any other helpful scripts for your specific purposes on your own.

RUNNING THE BSM2

When the archive has been unzipped you are ready to run the BSM2. A few simple instructions are given below to help you through the first time and to test the system on your computer.

- start Matlab and move to the BSM2 directory.
- command `mexall_bsm2` (if you have problems with the C-compiler you must solve this). Once successfully completed you only need to re-mex C-files that you have actually modified (which should normally not be done).
- command `bsm2_ss` (the Simulink window will appear).
- command `init_bsm2` (initiates all variables and parameters, loads the data files etc.);
- in the Simulink simulation menu select `Start`. The system will simulated 500 days forward in time using the constant influent data, the open loop configuration and solver ode15s. This should only take a few seconds.
- You now have a steady state solution. Give command `statevalues_bsm2` and all the results for all the processes will be displayed on the screen together with some extra information. Your results should be identical to the results provided in the pdf file *BSM2_steady_state* in the subdirectory *Results*.
- Command `stateset_bsm2`. The final values of the previous simulation have now been used to initialise all BSM2 models so that you can start your next simulation at the exact same position as where the last one ended. The same data are also automatically stored in a file *states_bsm2.mat*, which you can rename and then use at any time in the future to initialise the BSM2 system (works for all three Simulink models). If you want to save all the generated data from a simulation you should simply use the Matlab `save` command.
- Command `bsm2_ol` (the Simulink window will appear). In the Simulink simulation menu select `Start` (you will start from the previously reached steady state). The system will be simulated 609 days forward in time using the dynamic influent data, the open loop

configuration and solver ode45. Output data will be stored every 15 minutes. This simulation will require maybe 30-90 minutes (depending on computer power).

- After the simulation all data are stored in Matlab workspace and not to files. Use the `who` command to see what variables you have available. You could use the *statevalues_bsm2* script to see the results at the end of the simulation but you really want to evaluate the entire dynamic simulation.
- When the simulation is finished: command `perf_plant_bsm2`. The script will calculate and print to the screen the complete set of evaluation criteria for the overall plant performance and the risk indices for the simulation from day 245 to 609. It wil also plot some relevant variables. The risk index calculations will require quite some time (30 minutes) so you will be asked if you want to this or not. But do it the first time to make sure everything works. The results should be identical to the results shown in the Matlab column in the Excel file *Ringtest_BSM2full_110223* (sheet BSM2 OL (1 year)) in the subdirectory *Results*.
- We now do the same simulation for the closed loop case (starting from the same initial steady state situation). Command `load states_bsm2` (if your Matlab would have been shut down after the open loop simulation you should simply start Matlab, go to the BSM2 directory, give command `init_bsm2` and then command `load states_bsm2` and then you would be at the same point). Command `bsm2_cl` (the Simulink window will appear). In the Simulink simulation menu select `Start` (you will start from the previously reached steady state). The system will be simulated 609 days forward in time using the dynamic influent data, the closed loop configuration including noise and non-ideal sensor models and solver ode45. Output data will be stored every 15 minutes. This simulation will take maybe 60-90 minutes (depending on computer power). When the simulation is finished: command `perf_plant_bsm2`. The script will calculate and print to the screen the complete set of evaluation criteria for the overall plant performance and the risk indices for the simulation from day 245 to 609. The risk index calculations will require quite some time (30 minutes) so you will be asked if you want to this or not. But do it the first time to make sure everything works. The results should be more or less identical to the results shown in the Matlab column in the Excel file *Ringtest_BSM2full_110223* (sheet BSM2 CL (1 year)) in the subdirectory *Results*. There may be some minute differences depending on Matlab version, the noise generation, operating system etc.
- command `perf_controller_bsm2`. The script will calculate and print to the screen the complete set of evaluation criteria related to sensors and actuators for the simulation from day 245 to 609. It will also plot some relevant variables. The results should be more or less identical to the results shown in the Matlab column in the Excel file *Ringtest_BSM2full_110223* (sheet BSM2 CL (1 year)) in the subdirectory *Results*. There may be some minute differences depending on Matlab version, the noise generation, operating system etc.

When you have reached this point you can be sure that the BSM2 on your computer works properly. The full dynamic simulations require some time to finish. However, when you are testing new control strategies you do not always have to run the simulation for 609 days. A shorter period of time will give you a good indication of how it is working. You would then simply change the values of the parameters starttime and stoptime in the evaluation scripts

and evaluate the system for whatever time duration you want. For initial testing you may also want to increase the tolerances set for the solver, using ideal sensors and actuators etc. to speed up simulations. When you feel comfortable with your control strategy then you should test if for the full 609 days dynamic period including noise etc.

Note that for the BSM2 system there is rule saying that all influent flow above 60000 m^3/d should be bypassed. This is automatically done in the implementations provided here. However, if you are not aware of this fact it may cause some confusion for you when looking at detailed results of the simulations.

FINAL COMMENTS

Read the documentation about benchmarking from the web site carefully plus other available reports on BSM. Try to understand the structure of the different m-files and c-files to grasp how they relate to each other. Run *bsm2_ss*, *bsm2_ol* and *bsm2_cl* according to the description above to make sure everything works. You should compare your results with the results provided in the provided Excel document. If your results are different then you are doing something wrong.

When you feel confident that you understand this benchmark implementation you may start to add your own control strategies and try to improve the overall performance of the plant (which should not be too hard). Remember, the provided BSM2 system is simply a starting point and a fully verified simulation platform for you to start your own work.

Enjoy!