The WEST++ Wastewater Treatment Plant Modelling and Simulation Environment

Hans Vangheluwe, Filip Claeyts, Ghislain Vansteenkiste
Department of Applied Mathematics, Biometrics and Process Control (BIOMATH)
University of Ghent, Belgium

e-mail: Hans.Vangheluwe@rug.ac.be

1 Abstract

WEST++ is a modelling and simulation environment for any kind of process which can be described (in an Object-Oriented fashion) as a structured collection of Differential Algebraic Equations (DAE’s). Currently however, WEST++ is applied to Waste-Water Treatment Plant (WWTP) design and optimization. In particular, the IAWQ1 model [1] is the basis for a simple model of the dynamics of a basic WWTP. This article gives a brief overview of the WEST++ structure and functionality.

2 The WEST++ Environment

The problem of modelling and simulation of wastewater treatment plants is gaining importance as a result of growing environmental awareness. Compared to the modeling of well-defined (e.g., electrical, mechanical) systems, ill-defined systems modeling is more complex. In particular, choosing the “right” model is a non-trivial task. The wastewater treatment process dealt with in WEST++ is of the “activated sludge” type. This means the reduction of waste is performed by bacteria which consider (non-toxic) waste components as food. The (non-linear) dynamics and properties of the biological processes are still not very well understood. As a consequence, a unique model cannot always be identified. This, in contrast to traditional mechanical and electrical systems where the model can be derived from physical laws. Also, the calibration of wastewater treatment models is particularly hard: many expensive experiments may be required to accurately determine model parameters. Increasingly, the “system” modeled transcends the WWTP and includes the “environment” (in the engineering sense). The WWTP model is integrated in the overall model of a polluting plant, or of the river (and its natural water purification properties or toxicity tolerance) in which the effluent is dumped.

The aim of modelling—the representation, storage and meaningful re-use of knowledge about the dynamic behaviour of systems—on the one hand and the aim of simulation—the efficient and accurate solving of models to make system behaviour explicit in the form of state trajectories—are almost contradictory. Hence, the WEST++ Modelling and Simulation Environment makes a strict distinction between a modelling environment which aims to provide insight and enable re-use of model knowledge, on the one hand and the experimentation environment which aims to maximise accuracy and performance, on the other hand. As depicted in Figure 1, the Modelling Environment (ME) lets the user describe models in MSL-USER, a high-level, declarative language.

A hierarchical graphical editor (HGE) allows for the interactive composition of complex configurations from basic building blocks. The HGE starts from MSL-USER atomic models and outputs Coupled Model MSL-USER. The model compiler generates low-level (C++) MSL-EXEC code for execution within the Experimentation Environment (EE). Solvers within the EE generate data which can be solved interactively in the Plot Environment.

3 MSL-USER versus MSL-EXEC

MSL-USER is an object-oriented language which allows for the declarative representation of the dynamics of systems. Declarative means that the model (what) is presented without specifying how to solve it. The main characteristics
are:

- MSL-USER is built around a number of generic types corresponding to abstract mathematical concepts such as integer number, real number, product set, mapping, matrix, . . . . The “meaning” of objects of these types is given in a “denotational” fashion by referring to the corresponding mathematical concepts. By means of these types and with extra semantic rules, it becomes possible to represent different formalisms such as Petri Nets, Bond Graphs, System Dynamics, Differential Algebraic Equations (DAE), . . . , unambiguously.

- MSL-USER allows one to represent abstract models of the behaviour of (physical) systems. In particular, it is possible to represent “systems” in the system theoretical (state, transition function, output function, . . . ) sense.

- MSL-USER allows one to express physical knowledge such as units (m, kg, . . . ), quantity type (Length, Mass, . . . ), physical nature (across, through), boundary conditions, . . . . The semantics of these are known to the MSL-USER compiler which will check model consistency and, where appropriate, apply this knowledge in the translation towards MSL-EXEC.

- MSL-USER allows for the declarative, non-causal (implicit) representation of models. Through causality assignment, causal (explicit) equations are obtained. Non-causal models are far more reusable than causal ones.

- Re-use of models becomes possible thanks to the EXTENDS inheritance mechanism. This mechanism allows for the extension of an existing model. Thus, starting from generic models, a tree of extended models can be built.

- Classification is made possible through the SPECIALISES mechanism. Hereby, it is possible to indicate that a particular type is a sub-type of another type. This not only allows for classification, but also for rigorous type-checking.

The declarative specification is automatically checked (syntax and semantics) and transformed into a low-level representation (MSL-EXEC, with C++ binding in this case) suitable for efficient (numerical) simulation. The translation involves symbolic manipulation (computer algebra) and graph manipulation. The MSL-USER compiler is implemented in lex(flex), yacc(bison), and C++ and makes intense use of LEDA (Library of Efficient Data structures and Algorithms). MSL-EXEC is generated automatically from MSL-USER and contains both code to describe dynamics as well as code to represent the symbolic model information. MSL-USER is evolving towards a merge with the upcoming Modelica standard [2].

4 Activated Sludge WWTPs

As mentioned before, the WWTP processes dealt with in WEST++ are of the “activated sludge” type with Figure 2 depicting the basic structure of a plant. The principle of activated sludge is that in a reactor a community of micro-organisms is constantly supplied with organic matter and oxygen. The micro–organisms consume the organic matter and transform it by means of aerobic metabolism, partly into new microbial biomass and partly into carbon dioxide, water and minerals. The flow of water brings about a constant washout of the micro–organisms from the reactor to the settler or clarifier. Here, the micro-organisms which grow in flocs and have acquired a density sufficient to decant, are retained and removed with the underflow. Part of this sludge is then recycled to provide biomass to treat the new influent. The surplus amount is wasted (not shown in the Figure). In many respects, the aeration basin is comparable to a conventional fermentation reactor or chemostat. However, the purpose of the process is not to produce microbial biomass or a particular metabolite, but to mineralize incoming waste materials as much as possible. It is hereby of paramount importance to minimize biomass production since the latter has to be removed and treated in a subsequent phase.

Two important characteristics further distinguish the activated sludge system from conventional microbial fermentations. First, the active biological component comprises not a pure culture but an association of bacteria, yeast, fungi, protozoa and higher organisms as rotifers. These organisms grow on the incoming waste and interact with one another. Second, the sludge consists, in contrast with its qualification “active”, for an important part out of dead cells and cell debris. Indeed, young active microbial cells tend to grow in a dispersed way. The system is therefore operated in such a way that the substrate is limiting and the microbial biomass is quasi starving. Under these conditions cells grow slowly and in flocs. Because of this, the water in the decantor separates in a clear supernatant and a thick layer. Hence, the crucial part of activated sludge treatment is to select a microbial community which mineralizes at a fair rate the incoming waste and thereby produces a minimum of new biomass which furthermore sediments readily and completely out of the water when the latter reaches the decantor.
4.1 Activated Sludge Model No.1 (IAWQ1)

Wastewater treatment practice has now progressed to the point where the removal of organic matter, nitrification and nitrogen removal by biological denitrification, can be accomplished in a single sludge system. Because of the interactions within such systems, the mathematical models describing them are quite complex, which has detracted from their use. This is unfortunate because it is with such complex systems that the engineer has the most to gain from the use of mathematical models.

Realizing the benefits to be derived from mathematical modeling, while recognizing the reluctance of many engineers to use it, the International Association on Water Pollution Research and Control (IAWPRC) formed a task group in 1983 to promote the development of, and facilitate the application of, practical models to the design and operation of biological wastewater treatment systems. The goal was first to review existing models and second to reach a consensus concerning the simplest one having the capability of realistic predictions of the performance of single sludge systems carrying out carbon oxidation, nitrification, and denitrification. The model was to be presented in a way that made clear the processes incorporated into it and the procedures for its use.

The task group chose a matrix format for the presentation of the model. The first step in setting up the matrix is to identify the components of relevance in the model. The second step in developing the matrix is to identify the biological processes occurring in the system; i.e., the conversions or transformations which affect the components listed.

Within a system, the concentration of a single component may be affected by a number of different processes. An important benefit of the matrix representation is that it allows rapid and easy recognition of the fate of each component, which aids in the preparation of mass balance equations. The basic equation for a mass balance within any defined system boundary is:

\[
\frac{dM}{dt} = Input - Output + Reaction
\]

The input and output terms are transport terms and depend upon the physical characteristics of the system being modelled. The system reaction term can be easily obtained from the matrix terms.

13 components are considered in the IAWQ1 model [1]: soluble and particulate inert organic matter, readily and slowly biodegradable substrate, active heterotrophic and autotrophic biomass, particulate products arising from biomass decay, oxygen, nitrate and nitrite nitrogen, \(NH_3\) and \(NH_4^+\) nitrogen, soluble biodegradable organic nitrogen, particulate biodegradable organic nitrogen. There are 8 processes: aerobic and anoxic growth of heterotrophs, aerobic growth of autotrophs, ‘decay’ of heterotrophs and autotrophs, ammonification of soluble organic nitrogen, ‘hydrolysis’ of entrapped organics and of entrapped organic nitrogen.

In MSL-USER, the components are easily described as an enumerated type:

```plaintext
TYPE Components = ENUM {H_2O, S_S,...,X_{NH}};
```

Figure 3: Simple WWTP Model

A mass balance is written for each of the components:

```plaintext
FOREACH comp IN {H_2O, S_S,...,X_{NH}}:
    DERIV(M[comp],t) = inflow[comp] - outflow[comp] + reaction[comp]
```

whereby the reaction term is automatically generated from the IAWQ matrix (encoded in MSL-USER). The class hierarchy below is used to encode the diverse WWTP models: indentation denotes an extension of a class.

Physical DAE Models
- tanks without volume
  - pointsettler model
  - splitter models
  - combiner models
- tanks with volume
  - variable volume tanks
    - buffertank model
    - stormtank model
    - river detention tank model
    - VarVolumeConversionModel
    - VarVolumeIAWQConversionModel
    - VarVolumeInOutIAWQ
    - VarVolumeActivatedSludgeUnit
- fixed volume tanks
  - egalisationtank model
  - Otterpohl and Freund (primary clarifier)
  - FixVolumeConversionModel
    - FixVolumeIAWQConversionModel
    - FixVolumeInOutIAWQ
    - FixVolumeActivatedSludgeUnit

Coupled Models

5 Hierarchical Graphical Editor

As depicted in Figure 3, the WEST++ modelling environment allows for component based modelling (in this case of a simple WWTP): the user connects model icons in a hierarchical fashion. From this abstract specification, together with an MSL-USER library of dynamic models, one single MSL-USER model is produced. In particular, each icon put on the canvas results in the instantiation of an MSL-USER object of the appropriate class. Connections between icons result in MSL-USER connect statements,
which, within the CoupledModel formalism are expanded to the appropriate algebraic equalities. A compiler generates MSL-EXEC from this model for use within the experimentation environment.

6 Experimentation Environment

The Experimentation Environment depicted in Figure 4 enables the user to perform experiments on (compiled) models. As such, it is the interface between the user and the “simulator”. Figure 5 depicts the general structure of a “simulator”. The solver communicates efficiently (vectors are being passed) with the model dynamics part of the MSL-EXEC model. The simulator as a whole can be asked to perform a numerical simulation. This entails using the solver to generate a state trajectory for the particular MSL-EXEC model. Different numerical solvers may be chosen interactively. The simulator can also be queried for symbolic information. This will be retrieved from the symbolic information part of the MSL-EXEC model. An example of such symbolic information is the model structure visible in Figure 4.

The following distinguishes, based on general modelling and simulation theory, between different “experiment types” as implemented within the WEST++ environment. It is important to make a strict distinction between the user view of an experiment (experiment types) and the kernel view of an experiment (the solvers used to implement the experiments).

6.1 Kernel View

The kernel consists of a number of “solvers” which, when linked to a “model” (in MSL-EXEC), and given some parameters pertaining to the kernel’s operation (such as accuracy, termination conditions, ...), will “solve” that model. The “model” is just a “standard” defined representation (in C) of the mathematical object (such as an ODE) to be solved. In WEST++, a general MSL-EXEC model may contain an arbitrary sequence of Explicit Algebraic and Ordinary Differential Equations (ALGODE), Implicit Algebraic Equations (IALG), and Implicit Differential Algebraic Equations (DAE).

6.2 User View

The user thinks in terms of different “virtual experiments” on the model of a system. Typically, the sequence of these experiments, the “modelling and simulation life-cycle” comprises: structure characterisation, sensitivity analysis w.r.t. parameters, parameter estimation, simulation (either initial value or shooting problem), optimisation of a goal function using the model with known parameters, and sensitivity analysis w.r.t. inputs.

The following experiment types are implemented in WEST++:

1. Simulation Experiment (Initial Value and Shooting)

Mathematically, a model is specified by its equations (EQ), Initial or Terminal Values (TV), and Boundary Conditions (BC), in case of a system with more than one independent variable. Currently, in WEST++, there are two types of simulation experiments:

- Initial Value problem: state variable values are given at $t_{in}$. The simulator calculates the trajectory over $[t_{in}, t_{fin}]$. This is usually implemented using a forward integrator.
- Terminal Value, or End Value, or “Shooting” problem: state variable values are given at $t_{fin}$. The simulator calculates the trajectory over $[t_{fin}, t_{fin}]$. As a side-effect, the state variable values at $t_{in}$ result. These could be used as initial values in an Initial Value problem. This is implemented in WEST++ using an optimisation algorithm whereby the varied entities are the unknown initial conditions and the goal function is the sum of absolute values of differences between simulated end-value and desired end-value. In theory, a mix of both is possible.

Sometimes, it is necessary to “synchronise” with external data. This is for example the case when the input $u(t)$ is given as a table of measurements. Two options are available:
(a) the integrator can be forced to “synchronise” with the external data. This is done by means of time-events (as the input times are known in advance).

(b) the integrator can determine its own integration times and when a value is needed –at those times–, interpolation is used (0-th, or 1-st order). When the input is given as a continuous function, no interpolation is required.

Obviously, the latter solution is faster, though possibly incorrect.

2. Parameter Estimation Experiment
Certain (determined by user, possibly after hints from model sensitivity –w.r.t. parameter variations– analysis) model parameters are varied to minimise the “distance” between a simulated trajectory and a given (measured) trajectory. The distance measure is typically a sum of squares of differences between measured and simulated values though absolute values are also used. Differences between measured and simulated values can be calculated at different points in time: as described above, the integrator can be forced to “synchronise” with the external data, or interpolation can be used. In general, the differences can be weighted. The use of parameter estimation to “calibrate” WWTP models has proven to be a crucial factor in the usability of WEST++.

3. Optimisation Experiment
The most general use of the optimiser where some parameters are varied (possibly constrained) as to extremise a goal function.

4. Sensitivity Analysis Experiment
The sensitivity of the model with respect to initial value or parameter variations can be investigated. If sufficient symbolic information (partial derivatives) is available (provided in the MSL-EXEC by the MSL-USER compiler), this is straightforward. If this information is not available, either a Monte Carlo technique or some numerical technique can be used. Figure 6 depicts the simulated behaviour of the simple WWTP model in Figure 3. The effluent in the bottom window clearly contains the “purified” version of the influent in the top window. The bioreactor was modelled by means of an IAWQ1 model.

Figure 6: A Simple WWTP Experiment
References


