BOOSTING THE EFFICIENCY OF COMPOUND VIRTUAL EXPERIMENTS THROUGH A PRIORI EXPLORATION OF THE SOLVER SETTING SPACE

Filip H.A. Claeys
Department of Applied Mathematics,
Biometrics and Process Control (BIOMATH)
Ghent University
Coupure Links 653
B-9000 Gent
Belgium
E-mail: fc@biomath.ugent.be

Peter A. Vanrolleghem
model*EAU*
Département de génie civil
Université Laval
Pavillon Pouliot
Québec, G1K 7P4
QC, Canada
E-mail: peter@modelEAU.org

Peter Fritzson
Programming Environments
Laboratory (PELAB)
Linköping University
Campus Valla
SE-581 83 Linköping
Sweden
E-mail: petfr@ida.liu.se

KEYWORDS

modelling & simulation, software development, numerical solvers, distributed execution

ABSTRACT

Tornado is a new advanced kernel for modelling and virtual experimentation (*i.e.*, any evaluation of a model) in the water quality domain. Although primarily intended for use within this particular domain, the kernel is generic in nature and has a *plethora* of generally applicable features. This paper focuses on the ability of Tornado to perform *a priori* explorations of the integration solver setting space (*i.e.*, the powerset of all possible combinations of solver settings such as accuracy, stepsize, ...) in order to find improved settings that may boost the efficiency of complex virtual experiments. The technique is an extension of the standard scenario analysis virtual experiment type that normally allows for evaluating the effects of changes in model parameters and initial conditions. It is shown that through *a priori* solver setting space exploration, the total execution time of computationally complex virtual experiments can be substantially reduced. Results from an application of the method to Monte Carlo simulation of a river system in Luxembourg are included.

INTRODUCTION

In water quality research, the biological and/or chemical quality of water in rivers, sewers and wastewater treatment plants (WWTP) is studied. Research in this domain is facilitated by a number of models that have received a formal or *de facto* standardization status. Most notable are River Water Quality Model No.1 (RWQM1) (Reichert et al., 2001) and the Activated Sludge Model (ASM) series (Henze et al., 2000).

Water quality models typically consist of large sets of non-linear Ordinary Differential Equations (ODE) and/or Differential-Algebraic Equations (DAE). These equations are mostly well-behaved, although discontinuities occur regularly. The complexity of water quality models is therefore not in the nature of the equations, but in the sheer number. In WWTP, smaller models such as the well-known Benchmark Simulation Model (BSM) (Copp, 2002) consist of approximately 150 derived variables. Larger systems have up to 1,000 derived variables and over 10,000 (partly coupled) parameters.

Simulation of water quality models is a computationally intensive process. A continuous need for software tools that offer improved performance therefore exists. An example of a software tool that was recently developed is Tornado (Claeys et al., 2006b). It manages to substantially improve performance with respect to its predecessors. However, the overwhelming number of configuration options may be difficult to manage for users lacking the necessary background. More specifically, Tornado offers a wide variety of dynamically loadable solvers. Selecting an appropriate solver and configuring it properly is a non-trivial task, especially since not every modeller can be expected to be a numerical expert. Even for experts in this field, finding the most appropriate solver settings can be a cumbersome task since models tend to be very large and exhibit multiple features. Therefore, it is difficult to judge beforehand which feature will have the most influence on the solver settings. Whilst research into the automated, intelligent configuration of solvers is ongoing (Claeys et al., 2006c), Tornado already provides some more crude methods that help in the selection of appropriate solver settings, and hence to reduce the overall computation time.

The method discussed in this article is based on semi-automated exploration of the effects of a number of predetermined combinations of solver settings, through a modification of the standard scenario analysis experiment type available in Tornado. Since many Tornado experiments rely on the subsequent execution of a large number of simulations, investing time in searching for the most appropriate integration solver settings ultimately has a positive net effect on the overall computation time.

The first two sections of this article respectively give a short introduction to Tornado and its distributed execution capabilities. A more elaborate discussion can be found in Claeys et al., 2006b and Claeys et al., 2006a. The next sections discuss Tornado's standard scenario analysis and solver setting scenario analysis experiment types. Subsequently follows a discussion of an applica-

tion of the suggested method and finally, some conclusions.

THE TORNADO KERNEL

The Tornado kernel for modelling and virtual experimentation attempts to offer a compromise between the computational efficiency of custom hard-coded (typically FORTRAN or C) model implementations and the flexibility of less computationally efficient generic tools such as MATLAB. In (Copp, 2002) it is argued that for large ODE systems, generic tools such as MATLAB/SIMULINK will only be sufficiently efficient when large portions of the model are implemented as procedural code (*e.g.*, C), which is detrimental to the overall readability and maintainability of the model. In Tornado, hierarchical models are specified in high-level, declarative, object-oriented modelling languages such as MSL (Vanhooren et al., 2003) and Modelica (Fritzson, 2004). From these high-level specifications, efficient executable code is generated by a model compiler. Using the executable models generated by the model compiler, Tornado allows for running a variety of so-called *virtual experiments*. Virtual experiments are the virtual-world counterpart of real-world experiments, similar to the way models relate to real-world systems.

The most common virtual experiment type in Tornado is dynamic simulation (ExpSimul). Other experiment types include optimization (ExpOptim), sensitivity analysis (ExpSens), scenario analysis (ExpScen), Monte Carlo (*i.e.*, LHS - Latin Hypercube Sampling) analysis (ExpMC), and steady-state analysis (ExpSS). Virtual experiments are hierarchical in nature, in the sense that one experiment can be composed of a number of other experiments. We therefore distinguish between *atomic* and *compound* experiments. Atomic experiments are not made up of constituents and therefore cannot be decomposed (examples are ExpSimul and ExpSS). Compound experiments on the other hand rely on one or more (atomic or compound) sub-experiments (examples are ExpOptim, ExpSens, ExpScen and ExpMC).

The Tornado kernel relies on a flexible input provider and output acceptor mechanism to deal with I/O for virtual experiments. In order to allow for the kernel to be deployed in a diverse array of applications, it has been equipped with multiple interfaces. Next to its native C++ interface, Tornado currently also has a C, .NET and MATLAB MEX interface.

Tornado is portable across platforms and was designed according to the three-tier principle. Most persistent representations of information types are XML-based. The grammar of these representations is expressed in XSD (XML Schema Definition) format and mimics very closely the internal representation of the respective types of information.

Several applications (graphical and other) can be built on top of Tornado. One example includes the next generation of the WEST® (Vanhooren et al., 2003) modelling and simulation tool for WWTP's. However, the most direct way of using the kernel is through the Tornado CUI (Command-line User Interface) suite, which is a comprehensive set of tools that is included with the kernel distribution. The most commonly used CUI tools are the following:

- `tmsl`: Compiles a high-level model to executable model code
- `tbuild`: Compiles and links executable model code to a dynamically loadable binary object
- `tcreate`: Creates an empty XML description of a virtual experiment (to be further completed manually)
- `texec`: Executes virtual experiments described in XML. Dynamically loads the necessary solvers and executable models.

The results discussed in this paper were obtained through the Tornado CUI suite.

DISTRIBUTED EXECUTION USING TYPHOON AND LCG-2

Since compound virtual experiments in Tornado are computationally complex, and often consist of a number of mutually independent sub-experiments, there is in many cases a possibility for coarse-grained gridification, *i.e.*, parallellization at the level of sub-experiments. Fine-grained gridification would entail splitting up atomic sub-experiments into constituents, which is a complex task that is believed to cause severe overhead and is therefore not considered.

In order to allow for a certain degree of freedom with respect to the selection of distributed execution environments, a generic XML-based job description format has been defined within the scope of Tornado. From this generic description, jobs for two distinct distributed execution environments can currently be generated: Typhoon and LCG-2.

Typhoon (Claeys et al., 2006a) is a light-weight cluster-oriented distributed execution environment that was developed at BIOMATH. The Typhoon job dispatcher is capable of directly interpreting Tornado-generated generic jobs descriptions. Jobs are then transferred from the dispatcher to a number of registered worker nodes for execution.

LCG-2 (http://public.eu-egee.org) is the software that was developed in the scope of the Large Hadron Collider Grid project. It is deployed in the European EGEE project, which represents the largest European grid to

date. In order to convert generic Tornado job descriptions to LCG-2, a conversion tool named `t2jdl` is available in the Tornado CUI suite. This tool converts a XML description of a batch of N jobs to N individual LCG-2 job description files and a number of convenience shell scripts.

Tornado is able to generate generic job descriptions for simulation sub-experiments that are to be run from scenario analysis and Monte Carlo analysis compound experiments. In Claeys et al., 2006b it is demonstrated that execution of gridified scenario analysis experiments on the UGent Grid (a 41-node LCG-2 infrastructure) leads to a speedup factor of approximately 30.

STANDARD SCENARIO ANALYSIS

One of the most commonly used compound virtual experiment types in Tornado is scenario analysis. Scenario analysis in principle allows for running a number of simulations using different parameter values and/or initial conditions. During this process a number of aggregation functions can be computed from the simulated trajectories of a selected set of variables. Scenario analysis forms the basis of the solver setting exploration method that is discussed later. It is therefore described in this section in some more detail.

In order to clarify the way a scenario analysis experiment is structured, an excerpt from the graphical representation of the ExpScen XSD definition is presented in Figure 1. Each of the entities in this figure additionally has a number of simple or structured attributes that are not represented in the figure.

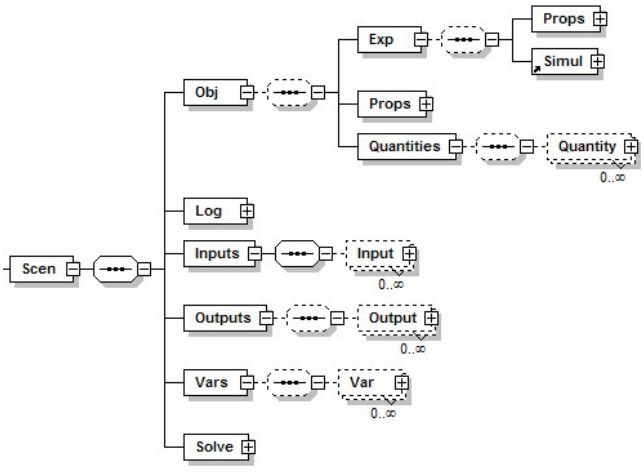

Figure 1: Grammar of Scenario Analysis XML Description

From the representation follows that the following entities play an important role in scenario analysis experiments:

- *Variables*: Model parameters and/or initial conditions that vary from one objective evaluation to the next. Values can either be supplied manually, sampled from a number of distributions (uniform, truncated normal, triangular), or evenly spaced (linearly or logarithmically) between a lowerbound and upperbound. In order to refer to the object that needs to be varied, all that is required is a fully-qualified object name (*e.g.*, *.a.b.c*, where *a* and *b* are submodels). Since the types of all entities are stored by the system, it can be determined automatically on the basis of an object name whether the value that needs to be varied is actually a parameter or an initial condition.
- *Objective*: Description of the evaluation that needs to be performed during each run. In this case the objective is defined as a simulation run, for which trajectories of a selected set of model variables are stored and later used for the computation of a number of aggregation functions.
- *Quantities*: Represent model variables for which trajectories need to be stored. For each quantity it can be specified which aggregation functions (such as Minimum, Maximum, Average, Integral, ...) are to be computed.
- *Solve*: Describes the sequence of evaluation of the powerset of variable values, *i.e.*, the set of variable value combinations. Alternatives are evaluation in plain sequential order, random order or according to the distance to a reference point in the scenario analysis variable space. Another possibility is only to evaluate a fixed subset of variable value combinations. Each of these alternatives is illustrated in Figure 2, for a simple case of 2 variables with 4 values each.
- *Inputs*: Represents input that is required for the computation of certain objective values.
- *Outputs*: Describes the output that is desired from the scenario analysis.
- *Log*: Describes the logging information that is desired.

As can be seen from Figure 1, the description of a scenario analysis objective includes a description of the simulation experiment to be executed. In fact, this description can either be embedded within the scenario analysis XML description itself, or reside in an external XML file. In either case, the structure of the simulation experiment description is similar to the scenario analysis description, as can been seen from the graphical XSD excerpt in Figure 3.

SOLVER SETTING SCENARIO ANALYSIS

As mentioned before, finding appropriate integrator solver settings can be a tedious task. It is typically a

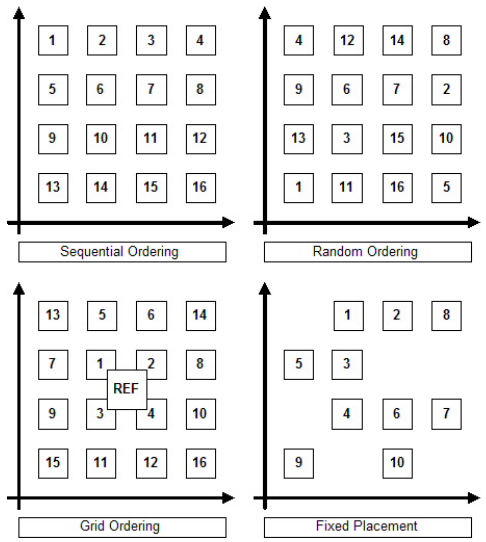

Figure 2: Scenario Analysis Evaluation Schemes

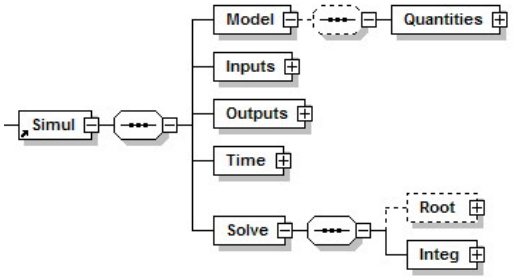

Figure 3: Grammar of Simulation XML Description

manual trial-and-error process in which iteratively solver settings are modified and simulations are run. The process continues until settings are found that solve the set of equations at hand with a sufficient level of accuracy, and in a timely fashion. Actually, this process is very similar to the scenario analysis process described above. It was therefore suggested to extend the standard scenario analysis functionality in Tornado so that not only model parameters and initial conditions can be varied, but also solver settings.

Thanks to the object-oriented and layered design of Tornado (Claeys et al., 2006b), implementation of solver setting scenario analysis was straightforward. In fact, the scenario analysis experiment only interacts with its embedded simulation experiment through a high-level abstract interface. The most relevant methods in this interface with respect to scenario analysis are the following:

- ```
  void
  SetValue(const std::wstring& FullName,
           double Value);
  ```

- ```
  double
  GetValue(const std::wstring& FullName);
  ```

- ```
  void Run();
  ```

In standard scenario analysis, the FullName argument in the Set/GetValue methods is a fully-qualified model object name. In order to allow for solver setting scenario analysis, the implementation of these methods was modified so that in case the FullName argument is structured differently, it is used to refer to a solver setting instead of a model object. More specifically, when a pattern such as *Solve.<Type>.<PropName>* instead of *[.<ModelName>]*.<ObjName>* is discovered as a FullName, the property of name *<PropName>* in the currently configured solver of type *<Type>* is modified. In the case of integration, the solver type is *Integ*. Other types of solvers are possible as well (such as optimizers, root finders, ...), but are not relevant in the scope of the current discussion.

The different methods that can be used for varying variables in scenario analysis (manual placement, sampling from distributions and linear/logarithmic spacing) turn out to be very practical in case solver settings need to be varied (*e.g.*, the use of logarithmic spacing for accuracies that are represented by negative powers of 10) . Also, the different methods for sorting value combinations (plain sequential, random, grid, fixed) are equally useful in solver setting scenario analysis as in standard scenario analysis. In case one only wishes to evaluate a limited set of predetermined combinations of settings, the fixed method is appropriate. In case one wants to evaluate an entire grid of combinations, without any preferences with regard to sequence, the plain sequential method seems most appropriate. The random method (or possibly grid method) could be used to walk through the solver setting space in order to get some initial understanding of how it behaves. Processing can be stopped as soon as this insight has been gained.

Another feature of standard scenario analysis that proves to be useful in the case of solver setting scenario analysis, is distributed execution. Indeed, as in standard scenario analysis, the number of runs to execute can become very large. Distributed execution therefore allows for faster evaluation of a large number of alternatives. Evidently, since in distributed execution there is no sequential execution of runs, the sorting methods of value combinations are not relevant here.

One must take into account that in case appropriate solver settings are sought for a certain simulation experiment through scenario analysis, these settings may only apply within a certain neighborhood of the initial values of the model. Strongly differing values may require a new analysis to be performed. If solver settings are sought that are to be applied within the scope of a compound experiment, one must make sure that the initial values of the simulation experiment that is iteratively run within the solver setting scenario analysis, are sufficiently representative for the simulations that are to be run in the standard compound experiment afterwards.
  ```

In solver setting scenario analysis, objective functions that are typically applied in standard scenario analysis are not very useful. More interesting in this case is to know the speed and accuracy of simulation. Speed of simulation could in principle be measured through the total simulation time. However, load variations of the machine on which the simulation is run render this metric unreliable in practice. A better approach is to relate simulation speed to the computational complexity of the simulation. A good measure for this is the number of state evaluations of the model. Indeed, the fewer times the state of a model needs to be computed during a simulation, the faster the simulation will run. Since Tornado by default internally computes the number of state evaluations, it was straightforward to make this metric available as an objective for scenario analysis. It should be noted that next to the model, the solver itself also introduces some computational complexity. In the case of small models, this solver-incurred complexity is non-negligible. However, for the elaborate models that are typically dealt with in the scope of Tornado, the complexity of the model largely outweighs the solver complexity.

For measuring the simulation accuracy, Tornado also has some useful functionality available: it allows for computing measures of similarity between a simulated trajectory and a reference trajectory (either by computing the sum of squared, relative or absolute differences (DiffSum), or by the computing the maximum difference (DiffMax)). This functionality has also been made available as an objective, so that trajectories that are simulated through solver settings scenario analysis can be compared with a reference trajectory with the desired accuracy, established beforehand.

Figure 4 shows the overall flow of the suggested procedure. First, a solver setting scenario analysis is performed for a simulation experiment with initial values that are representative for the compound experiment that follows. From the results obtained (number of state evaluations, and difference with regard to reference trajectories), an appropriate solver configuration is selected, which is then applied to the simulation experiment. Next follows the compound experiment (typically a standard scenario analysis or Monte Carlo analysis) that will iteratively run this simulation experiment, and should benefit from the improved solver settings that were found through the first step of the procedure. During both steps of the procedure, simulation sub-experiments can either be run sequentially or concurrently, in case a distributed execution environment is available. In order to further clarify both steps of the procedure, a functional comparison of standard and solver setting scenario analysis is presented in Table 1.

APPLICATION TO THE LUXEMBOURG CASE

Recently, an integrated model was built for the Sûre river in Luxembourg, and two of its tributaries (Solvi, 2006).

Table 1: Comparison between Standard and Solver Setting Scenario Analysis

Item	Standard	Solver Setting
Variables	Model parameters	Solver settings
	Initial conditions	
Objective	Aggregation functions	#StateEvals
	DiffSum, DiffMax	DiffSum, DiffMax
Solver	Plain Sequential	Plain Sequential
	Random, Grid	Random, Grid
	Fixed	Fixed
Gridification	Available	Available

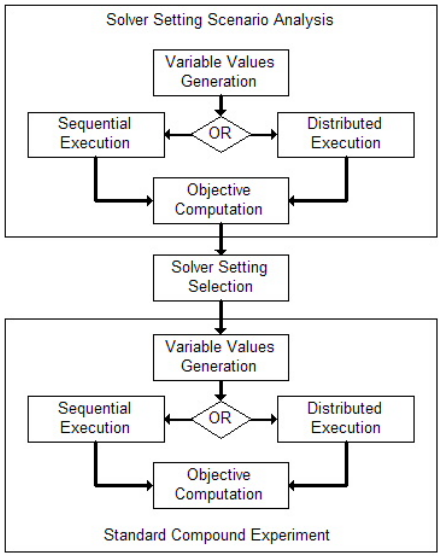

Figure 4: Solver Setting Scenario Analysis Procedure

Flow in the receiving river(s) is modelled as a tank cascade, and water quality is represented by a simplified version of the IWA river water quality model RWQM1 (Reichert et al., 2001). It contains processes for oxygen, biodegradable organic matter, nitrogen and phosphorus cycles, pH and algae growth. The model can be reduced to adapt to local circumstances and was developed to be compatible with the IWA standard activated sludge models (ASM1, ASM2, ASM3 (Henze et al., 2000)) for modelling WWTP's. For urban drainage and sewer transport, an adapted version of the German KOSIM model was implemented into the Tornado model library (Solvi, 2006).

The overall model consists of some 12,000 (mainly coupled) parameters, 4,400 algebraic variables and 414 ODE's. For this model, a Monte Carlo analysis was set up that takes 1,000 shots from a 32-dimensional parameter space. Of the 32 parameters involved, 30 vary according to a uniform distribution; the remaining 2 are triangularly distributed.

For simulating the model, an integration solver had to be chosen amongst the wide variety of solvers available in

Tornado (approx. 20 solvers available at this moment). In a first instance, a conservative approach was followed which lead to the selection of a variable stepsize Runge-Kutta 4 solver. The solver was shown to work well for all 1,000 runs of the Monte Carlo analysis. With this solver, a simulation run takes on average 53 s on a reference machine (HP DL145 Dual Opteron 242 1.6GHz, 4GB RAM, 40GB HD) and results in an average total of 392,356 state evaluations per run.

In general, advanced solvers such as CVODE, which is part of the SUNDIALS suite (http://www.llnl.gov/CASC/sundials), often yield better performance than the more basic Runge-Kutta solver that was chosen at first. Unfortunately, a number of simulation runs with manually configured CVODE settings initially lead to the belief that in this case CVODE would not be appropriate. However, CVODE has many combinations of solver settings, and it is very unpractical to test each of these manually. Therefore, the above-mentioned approach based on solver setting scenario analysis was applied. The CVODE solver settings and values that were chosen as variables for the analysis are listed in Table 2:

Table 2: Solver Settings for the Luxembourg Case

Name	Values
AbsoluteTolerance	1e-3, 1e-4, 1e-5, 1e-6, 1e-7, 1e-8
RelativeTolerance	1e-3, 1e-4, 1e-5, 1e-6, 1e-7, 1e-8
IterationMethod	Functional (0), Newton (1)
LinearMultistepMethod	Adams (0), BDF (1)
LinearSolver	Dense (0), Diag (2), SPGMR (3)

It should be noted that IterationMethod, LinearMultistepMethod and LinearSolver are not represented by a real value, but by an enumerated value (a string in Tornado). Since scenario analysis cannot deal with enumerated values, they have to be mapped to an integer value (as is illustrated by the values between parenthesis in Table 2.

The values for Absolute & Relative Tolerance could in principle be specified through the manual placement method, but in this case they were specified through logarithmic spacing, as illustrated in Table 3. Since for IterationMethod and LinearMultistepMethod only 2 choices exist, these were specified through the manual placement method. This also applies to LinearSolver, for in this case the values (0, 2 and 3) are unrelated.

As can easily be seen from Table 2, the total number of solver setting combinations and hence the number of simulations to be run is 6 x 6 x 2 x 2 x 3 = 432. A number of simulations as high as this evidently is only advisable in case the standard compound experiment that follows also consists of a high number of simulation runs, and in addition the expected speed-up for one simulation is substantial. In case these conditions are not fulfilled, the number of *a priori* simulations may have to be reduced in order to avoid overall performance degradation instead of the improvement that is hoped for.

Table 3: Properties of Absolute & Relative Tolerance Scenario Analysis Variables

Name	Value
DistributionMethod	Logarithmic
LowerBound	1e-8
UpperBound	1e-3
Spacing	10

During the scenario analysis, the average number of state evaluations as well as the total number of failures of the CVODE algorithm (due to stepsize underflow) were determined for each IterationMethod / LinearMultistepMethod / LinearSolver pattern, as is shown in Table 4. From this table follows that 1-0-0, 1-0-2, 1-0-3, 1-1-0 and 1-1-2 are unreliable since these settings sometimes result in CVODE failures. For the remaining settings, the difference with respect to reference trajectories was studied. From this, is was concluded that the 0-1-* patterns lead to appropriate solver settings for the simulation at hand, for all Absolute and Relative Tolerances considered. The 0-0-* patterns (which have a more or less comparable number of state evaluations) are less favorable since in this case the difference with respect to the reference trajectories is higher. On the basis of the number of state evaluations, one might be lead to believe that also the 1-1-3 pattern is a good option. However, in this case the resulting trajectories differ considerably from the desired reference trajectories. This indicates that this pattern is actually unstable, although no explicit CVODE failures have occurred. In view of the preceding analysis, it was decided for this application to retain the 0-1-0 pattern in combination with a Relative and Absolute Tolerance set to 1e-6.

Table 4: Total Number of Failures and Average Number of State Evaluations per Pattern (average over all selected tolerance settings)

Pattern	#Failures	Avg(#StateEvals)
0-0-0	0	150,128
0-0-2	0	150,128
0-0-3	0	150,128
0-1-0	0	142,943
0-1-2	0	142,943
0-1-3	0	142,943
1-0-0	4	55,879
1-0-2	26	29,835
1-0-3	2	167,988
1-1-0	6	6,052
1-1-2	27	8,782
1-1-3	0	1,594

Using the new solver settings, the average simulation time per run could be reduced from 53 to 33 s. The number of state evaluations could be more than halved (from 392,356 to 153,490). This implies that for sequential execution, the total simulation time for 1,000 runs could be reduced by 1,000 x 20 s, or approximately 5.5 hours. Compared to the approximate 2.5 hours that was spent on the *a priori* solver setting exploration, this results in a net performance gain of 3 hours. This clearly shows that because of the amplification of performance improvements (or degradations) that result from compound virtual experiments, it is beneficial to invest time into the exploration of the solver setting space *a priori*.

CONCLUSION

This paper touches on the application of scenario analysis to the problem of finding appropriate integration solver settings for a certain simulation. It shows that in case solver settings are to be applied to a simulation sub-experiment executed iteratively within the scope of a compound experiment, investing time into the *a priori* exploration of the solver setting space can substantially reduce the overall simulation time. Evidently, the scale of the *a priori* exploration and the potential performance benefit need to be well-balanced.

Thanks to the object-oriented design of the tool used for this study, implementation of solver setting scenario analysis on the basis of standard scenario analysis was straightforward.

In the study that was reported on in this article, a crude scenario analysis technique for finding good solver settings was used. It was shown that this technique allows for the reduction of the overall computation time of large-scale problems, even though it does not exhibit any form of intelligence. In future studies, heuristics and additional knowledge on model features may be introduced, which should allow for a reduction of the required number of *a priori* simulation runs.

REFERENCES

F. Claeys, M. Chtepen, L. Benedetti, B. Dhoedt, and P.A. Vanrolleghem. Distributed virtual experiments in water quality management. *Water Science and Technology*, 53(1):297–305, 2006a.

F. Claeys, D. De Pauw, L. Benedetti, I. Nopens, and P.A. Vanrolleghem. Tornado: A versatile efficient modelling & virtual experimentation kernel for water quality systems. In *Proceedings of the iEMSs 2006 Conference*, Burlington, VT, 2006b, *Accepted*.

P. Claeys, F. Claeys, and P.A. Vanrolleghem. Intelligent configuration of numerical solvers of environmental ODE/DAE models using machine-learning techniques. In *Proceedings of the iEMSs 2006 Conference*, Burlington, VT, 2006c, *Accepted*.

J.B. Copp, editor. *The COST simulation benchmark*. European Commission, 2002.

P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Press, February 2004. ISBN 0-471-47163-1.

M. Henze, W. Gujer, T. Mino, and M. van Loosdrecht. *Activated Sludge Models ASM1, ASM2, ASM2d, and ASM3*. Scientific and Technical Report No.9. IWA Publishing, London, UK, 2000.

P. Reichert, Borchardt D., Henze M., Rauch W., Shanahan P., Somlyódy L., and P.A. Vanrolleghem. *River Water Quality Model No.1*. Scientific and Technical Report No.12. IWA Publishing, London, UK, 2001.

A.-M. Solvi. Construction and calibration of an integrated model for catchment, sewer, treatment plant and river. In *Proceedings of the 7^{th} International Conference on Hydroinformatics 2006*, Nice, France, 2006.

H. Vanhooren, J. Meirlaen, Y. Amerlinck, F. Claeys, H. Vangheluwe, and P.A. Vanrolleghem. WEST: modelling biological wastewater treatment. *Journal of Hydroinformatics*, 5(1):27–50, 2003.

AUTHOR BIOGRAPHIES

FILIP H.A. CLAEYS was born in Ghent, Belgium. He received a MSc in Computer Science from Ghent University and a Master's in Artificial Intelligence from K.U.Leuven. He currently works as a senior software engineer for HEMMIS N.V. and leads a research group in the field of modelling and simulation software tools at Ghent University.

PETER A. VANROLLEGHEM, bio-engineer, PhD, heads the model*EAU* research team at Université Laval (Québec) and has ample experience with modelling, monitoring and control of wastewater treatment systems. He has over 175 peer-reviewed papers and is very active within the International Water Association.

PETER FRITZSON is head of the Programming Environment Laboratory at Linköping University, Sweden. He holds the positions of research manager at MathCore Engineering AB, chairman of the Scandinavian Simulation Society and vice-chairman of the Modelica Association. He has published 10 books and over 100 scientific papers.