Automatic numerical solver selection from a repository of pre-run simulations

Petra Claeys, Peter A. Vanrolleghem and Bernard De Baets

ABSTRACT

Numerical solver uncertainty is high when the solutions of the differential equations of a model, computed with different numerical solvers, deviate from each other. Numerical solver uncertainty is a serious limiting factor of the simulation process and can lead to incorrect model predictions. This problem is especially critical because the correct solution trajectory of environmental models, often consisting of large systems of ODEs, is almost always unknown. The selection of the most appropriate solver, according to speed and correctness, is not a straightforward task and cannot be based on, for instance, literature. Moreover, with the advent of distributed computing, large amounts of data on previously run simulations are readily available. Analyzing these data can help automating the selection of the most appropriate solver. A new methodology for this automatic selection, based on the correctness of the solution from a repository of simulations, was developed and tested on a set of 16 models with different levels of complexity. This methodology is capable of finding deviating solutions when the model is computed with different solvers and settings, and shows that numerical solver uncertainty is quite common. A cluster of appropriate solvers, which are able to solve the model correctly, can be identified and the most efficient solver can be selected among them. This results in a reduction of the numerical solver uncertainty. On top of that, it was also possible to achieve a reduction of the computation time by a factor of 10^6, compared to slow, but undoubtedly correct solvers.

Key words | automatic solver selection, clustering, correct solution trajectory, differential equations, numerical solver uncertainty, simulation time

Petra Claeys
Peter A. Vanrolleghem
BIOMATH, Department of Applied Mathematics,
Biometrics and Process Control,
Ghent University, Coupure links 653,
Ghent B-9000,
Belgium
E-mail: *pclaeys@biomath.ugent.be*

Peter A. Vanrolleghem
modelEAU, Département de génie civil,
Pavillon Pouliot, Université Laval,
Québec G1K 7P4,
QC,
Canada

Bernard De Baets
KERMIT, Department of Applied Mathematics,
Biometrics and Process Control,
Ghent University, Coupure links 653,
Ghent B-9000,
Belgium

INTRODUCTION

Numerical solver uncertainty is a source of uncertainty that considers all aspects of uncertainty related to the numerical solver that is used to integrate the differential equations of a model (Claeys *et al.* 2008). A numerical solver is implemented in a programming language, and has a range of simulation problems to which it is applicable. It is a well known fact that a given numerical solver, although suitable for a certain type of differential Equation (PDE, ODE, DAE, etc.), cannot correctly simulate every possible model constructed with that type of differential Equation (Seppelt & Richter 2005). When the chosen solver and its settings are appropriate, they will not influence the behavior of the solution significantly, but when they are incorrect the influence becomes stronger, and so does the total uncertainty of the simulation. Sometimes solvers fail and generate an error, and computation stops without the generation of solution trajectories. Choosing the appropriate numerical solver for a certain initialized model is not a straightforward task (Claeys 2008*b*). In order to make this choice, a model user must combine the knowledge he/she has of the conceptual model, and its initial conditions, with the mathematical and practical knowledge of certain numerical solvers and their settings. Unfortunately, most model users do not have the opportunity to gain experience as a

doi: 10.2166/wst.2009.034

modeller/mathematician. Most of the time they rely on the default choices, provided with the modelling software, which can give rise to wrong results or unnecessarily long computation times.

Software tools that help in the selection of the most appropriate solver can further enhance the performance of simulations, and diminish numerical solver uncertainty. Some stand-alone systems that give advice on the intelligent configuration of a solver have already been developed: ODEXPERT (Kamel *et al.* 1993), EPODE (Kamel *et al.* 1993), PYTHIA (Weerawarana *et al.* 1996) and PYTHIA-II (Houstis *et al.* 2000).

ODEXPERT uses textual parsing to determine some properties of the ODEs such as the number of additions, subtractions, multiplications, divisions, exponentiations and function evaluations. It also determines the linearity of the system, but when no external symbolic manipulation module is available, misclassification is possible. Stiffness tests can also take place but again require an external symbolic manipulation module. The system recommends a solver, based on a rule-based knowledge base that uses the characteristics of the equations to determine a suitable solver.

EPODE automatically detects the characteristics of the problems, using symbolic manipulations, and uses these characteristics to select the most appropriate solver (Petcu & Dragan 2000).

PYTHIA assists in the choice of partial differential equation solvers for boundary value problems. It uses information on the characteristics of the equations, and uses this knowledge together with previously seen problems, to make a decision on which solution algorithm to use. Determining the characteristics of the equations is a difficult problem and PYTHIA requests this detailed information from the user.

PYTHIA-II uses a database with previous cases to select a partial differential equation solver, using objectives that take speed (performance) and errors for each of the given solvers using various grid sizes into account. The rules that are extracted to select solvers are based on the characteristics of the equations and the objectives.

Another system was proposed by (Bunus 2007). It generates a domain-specific solver configuration model through the use of decision trees, after detecting the nature of the equations and performing symbolic manipulations on them. The domain-specific solver configuration model contains the appropriate numerical solver to compute the model, together with the model that is transformed into a form required by that solver.

Numerical analysis literature on this subject exists only for particular (smaller) models. Stiffness tests exist, as described in the work of (Ascher & Petzold 1998) and (Cameron & Hangos 2001), but these are based on the Jacobian matrix, which depends on the state of the model for non-linear systems of ODEs.

Most of these methods need symbolic manipulations to detect the characteristics of the equations and use expert knowledge, based on these characteristics, to deduce the most appropriate solver. PHYTIA also uses a database of previous cases to help in the selection of an appropriate solver. In all these systems the correctness of the results is not evaluated automatically. Moreover, a system that helps users with the choice of the solver for complete environmental models, also taking correctness of the solution trajectories into account, is not yet available.

The aim of the reported research is to develop a methodology that helps to find the most appropriate solver for an initialized model, based on the correctness of the solution trajectory and on the speed of the computation. It is a striking fact that the correct solution of environmental models is almost always unknown. The initialized model is defined as a computerized environmental model, together with its dynamic input data, its initial values, its parameter values, and its simulated period (or time interval) (Claeys 2008*b*). With the advent of distributed computing, large amounts of data on previously run simulations is readily available, and one can therefore rely on historical data to achieve the above aim, by interpreting the plethora of historical data. In this manner, symbolic manipulations and additional complex computations are avoided, since these are impractical on large systems of ODEs.

PREVIOUS WORK

To realize the use of historical data, an efficient mechanism was needed to archive all important information on previously completed simulations. The modelling framework

used in this work is Tornado© (Claeys 2008*a*), and a command line software module that makes all required information persistent was developed for this framework. Information on the model under consideration, the input values, the parameter values, the initial values, the time interval, the success or failure of a simulation, the computation time, and the resulting trajectories is saved into a MySQL database (Claeys 2008*b*). In this manner, elapsed simulations can be quickly queried from the MySQL database. In the remainder of this work, we use the broader term 'repository' to indicate the MySQL database, that is filled with previously run simulations. Note that a tool to make simulations persistent can be developed for any modelling framework.

The computation time is quantified by the number of state evaluations of the model (Claeys 2008*b*). This quantity stands for the number of times the state of the model is computed and is referred to as 'NrOfCS' in this contribution.

KEY IDEAS

In order to use pre-run simulations to guide in the selection of the most appropriate solver, a broad range of solvers must be available. Unfortunately, it is impossible to acquire experiments, solved with every possible integration solver that exists, so one must use 'a sampling frame' of solvers. Tornado© offers a wide range of dynamically loadable integration solvers, and it was therefore decided to let the sampling frame select among these integration solvers (Claeys 2008*a*). Again, any other modelling framework that supports many numerical solvers can be used. A collection of simulations of a certain initialized model, computed using these integration solvers, is considered to be a good representation of the possible outcomes a model can have depending on the choice of the solver.

The correct solution trajectory of the models that we use is almost always unknown and one can therefore not decide whether a solver has worked properly. In this work a methodology to automatically find a possibly correct solution trajectory was developed. It is assumed that the correct solution will recurrently show up in the collection of samples, obtained with different numerical solver settings. In other words, it is assumed that correct trajectories are the rule while erroneous ones are the exception. Furthermore it is believed that erroneous trajectories will not resemble each other while correct ones will almost be identical. However, one must remain perceptive because exceptions can be expected. For these exceptions, verification using experimental data is a possible approach, and future research can be necessary to solve the issues that result.

In order to find the correct solution trajectory it was decided not to rely on experimental data because they typically are not available. Instead, an unsupervised agglomerative classification technique is used, using the fast Fourier coefficients of the solution trajectories. A clustering algorithm, namely UPGMA (Dawyndt *et al.* 2006), is used to group all samples hierarchically, according to a metric based on their power spectra. Solution trajectories that closely resemble each other are collected into a same cluster at an early stage of the clustering procedure, while solutions that are significantly different, according to shape and behavior, will remain in separate clusters during several stages of the clustering procedure. Distances between clusters are recorded and are analyzed in order to identify the cluster that contains numerous solutions that are nearest to each other.

FAST FOURIER TRANSFORM

When $h(n)$ is a discrete signal that contains N samples: $[h(0), \ldots, h(N-1)]$, its fast Fourier transform is given by:

$$H(k) = \sum_{n=0}^{N-1} h(n) e^{(2\pi kn/N)i} \quad \text{for } k = 0, \ldots, N-1 \tag{1}$$

The power spectral density of a continuous signal describes how the power of a signal is distributed with frequency. The estimator of the power spectral density of a random signal from a sequence of discrete time samples $h(t)$ is called a periodogram. Spectral leakage is a phenomenon that appears as if some energy has 'leaked' out of the original signal spectrum into other frequencies. This happens because the frequencies are discrete values. To diminish spectral leakage data windowing is applied. It multiplies the original data $[h(0), \ldots, h(N-1)]$ in the time

domain by a specific window function $w(n)$. The resulting Fourier transform is given by (Press *et al.* 1992):

$$D(k) = \sum_{n=0}^{N-1} h(n)\, w(n) e^{(2\pi kn/N)i} \quad \text{for } k = 0, \ldots, N-1 \tag{2}$$

A window function should be left-right symmetric: it should rise from zero to a peak, and then fall again. For our computations the Bartlett window was used, given by (Press *et al.* 1992):

$$w(n) = 1 - \left| \frac{n - \frac{1}{2}N}{\frac{1}{2}N} \right| \tag{3}$$

The estimator for the power spectrum that was used, is given by (Press *et al.* 1992):

$$\begin{aligned} P(k) &= \frac{1}{W_{ss}} |D(k)|^2 \quad \text{for } k \in \left\{0, \frac{N}{2}\right\} \\ P(k) &= \frac{1}{W_{ss}} \left[|D(k)|^2 + |D(N-k)|^2 \right] \quad \text{for } k = 1, \ldots, \frac{N}{2} - 1 \end{aligned} \tag{4}$$

where W_{ss} is given by:

$$W_{ss} = N \sum_{n=0}^{N-1} w(n)^2 \tag{5}$$

SIMILARITY IN FUZZY SET THEORY

Similarity and dissimilarity are inverse relations. The first indicates the degree to which two objects are similar, while the latter indicates the degree to which two objects are different. When the similarity between two objects increases, the dissimilarity between these objects decreases. Similarity has been studied in different disciplines, such as taxonomy, psychology and statistics. For the problem under study the feature-based approach used in taxonomy appears very interesting, because a collection of $P(k)$ can be seen as a set of distinct objects, and the values of the $P(k)$ can be viewed as the presence or absence of a fictitious attribute. Unfortunately, this approach is inappropriate, since in classical set theory information on the presence of attributes is always represented using {0,1}-vectors. As an alternative, fuzzy set theory permits the gradual assessment of the membership of elements in a set. This membership is expressed through a membership function, that takes values in the real unit interval [0, 1].

Fuzzy sets and fuzzy relations

A fuzzy set Aon a finite universe $U = \{u_1, u_2, \ldots, u_n\}$ is a function from $U$to [0,1], which specifies the degree $A(ui)$ to which each ui belongs to A. Membership functions should not be confused with probability distributions: they do not represent the likelihood of an event, but a degree of truth that is situated between 0 (false) and 1 (true).

A binary fuzzy relation R on a finite universe $U = \{u_1, u_2, \ldots, u_n\}$ is a fuzzy set on the Cartesian product $U \times U$. $R(u_i, u_j)$ indicates the degree to which u_i is related to u_j. This fuzzy relation can also be represented as a matrix A_R, with elements $a_{ij} = R(u_i, u_j)$ for any $i, j = 1, \ldots, n$.

Similarity measures in set theory

A large family of similarity measures exists for ordinary subsets A and B of a finite universe $U = \{u_1, u_2, \ldots, u_n\}$ (De Baets *et al.* 2002):

$$\begin{aligned} S(A,B) &= \frac{x\,\Delta_{A,B} + y\,\delta_{A,B} + z\,\upsilon_{A,B}}{x'\,\Delta_{A,B} + y\,\delta_{A,B} + z\,\upsilon_{A,B}} \\ &\text{with } A,\ B \text{ in the power set } P(X) = \{0,1\}^X, \\ &\text{where } \Delta_{A,B} = |A \Delta B| \\ &\delta_{A,B} = |A \cap B| \upsilon_{A,B} = |(A \cup B)^c| \\ &\text{and } x, x', y, z \text{ are positive reals such that } 0 \le x \le x', \\ &\text{and } |A| \text{ denotes the cardinality of a set } A \end{aligned} \tag{6}$$

Similarity measures for fuzzy sets

Fuzzification schemes exist to translate the family of similarity measures for crisp sets into their fuzzy counterpart (De Baets & De Meyer 2005). These schemes rely on the fuzzification of the cardinality of an ordinary set, and on the fuzzification of the basic classical set operations, such as intersection, union and (symmetric) difference. The family of similarity measures for crisp sets can be translated into a family of similarity measures between two fuzzy sets

A and B in a finite universe $U = \{u_1, u2, \ldots, u_n\}$:

$$S(A,B) = \frac{x(a+b-2u) + yu + z(n-a-b+u)}{x'(a+b-2u) + yu + z(n-a-b+u)},$$

$$\text{where } a = \sum_{i=1}^{n} A(u_i)$$

$$b = \sum_{i=1}^{n} B(u_i) \tag{7}$$

$$u = \sum_{i=1}^{n} Q(A(u_i), B(u_i))$$

and Q denotes a commutative quasi − copula.

A quasi-copulas is a function Q: $[0,1]^2 \mapsto [0,1]$ which satisfies (Genest *et al.* 1999):

1. Neutral element 1 and absorbing element 0.
2. Monotonicity: Q is increasing in each variable:

   $Q(x_1, y_1) \leq Q(x_2, y_2)$ if $x_1 \leq x_2$ and $y_1 \leq y_2$

3. 1-Lipschitz property: for any $(x_1, x_2, y_1, y_2) \in [0,1]^4$ it holds that: $|Q(x_1, y_1) - Q(x_2, y_2)| \leq |x_1 - x_2| + |y_1 - y_2|$

Quasi-copulas are used to generalize classical set intersection to fuzzy sets. The mapping $d(A,B) = 1 - S(A,B)$ is a pseudo-metric when $x' \geq \max(y,z)$. In this paper, we have restricted our attention to the subfamily of similarity measures that results when $x' = y = 1$ and $x = z = 0$, or when $x' = y = z = 1$ and $x = 0$, which all yield pseudo-metrics. These similarity measures can be seen as generalizations of the well known Jaccard similarity coefficient, and the simple matching coefficient, respectively.

METHODOLOGY

In general, when the step size is sufficiently small, solvers are able to compute the correct solution trajectory. Especially for stiff problems, when the step size becomes too big, the numerical solution becomes unstable. Unstable solutions diverge from the real solution trajectory of the system of differential equations under consideration. Stability regions define a threshold on the step size that still produces a stable solution.

Explicit (Euler, AB2, AB3, AB4) and Runge Kutta methods (RK2b, Midpoint, RK2a, RK4, RK4ASC) in combination with a moderate step size or accuracy, are very slow, and are in the remainder of this work denoted as 'reference solvers'. The stability regions of the reference solvers considered in this work are not identical, but overlap. Reference solvers and other solvers that fail for a certain model are not considered anymore for that model.

Comparison of solution trajectories is done on the basis of their power spectra. The fast Fourier method and the periodogram method were used. According to the recommendations of Press *et al.* (1992) the choice of the window function is the Bartlett window.

It is assumed that in the repository a range of solution trajectories for a certain (environmental) model reside, computed using different solvers and settings. The vector of all frequencies of the power spectrum, for the different simulations (L) and the different model outputs (P) are the elements of a matrix as shown in Figure 1. The number of outputs considered is set by the user.

The methodology for automatic solver selection consists of 4 steps:

1. Frequency selection.
2. Normalization of the values.
3. Similarity and distance computation.
4. Clustering.

Frequency selection

The process of comparing solution trajectories on the basis of all available frequencies can be very time-consuming.

	$Output_1$	$Output_2$	$\cdots$	$Output_P$
Sim_1	$Freq_{11}$	$Freq_{12}$	$\cdots$	$Freq_{1P}$
Sim_2	$Freq_{21}$	$Freq_{22}$	$\cdots$	$Freq_{2P}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	
Sim_L	$Freq_{L1}$	$Freq_{L2}$	$\cdots$	$Freq_{LP}$
Result	F_1	F_2	$\cdots$	F_P

Figure 1 | The elements ($Freq_{lp}$) of this matrix are vectors that contain all frequencies of the power spectrum, per simulation (l) and per model output (p). The result after frequency selection is one vector of frequencies per output for all simulations (F_p).

Moreover, in general the higher the frequencies, the more the values of the power tend to zero. Including all these small values would unrealistically dominate the comparison. For this reason it was decided to compare the trajectories on the basis of parts (not all frequencies) of their power spectra. Hence, the first step of this methodology is the selection of these frequencies.

Frequencies ($Freq_{lp}$) are selected per output variable (p) for all simulations. For a certain output (p) and a certain simulation (l) all frequencies with a power value ($Val_{lp}(x)$) above 0.1% of the maximum power value of that simulation (m) are kept. These frequencies are selected in this way for all simulations (L) and are gathered per output, by taking the union over all simulations, but not over all outputs. This yields for each output a vector of frequencies gathered from all simulations (l) with that output (F_p).

At the end, for every simulation, the power values that correspond to the frequencies of every F_p are taken for each output, and the vectors V_{lp} are produced.

The selection of frequencies can vary depending on the specific objectives of the analysis. One can perform a selection based on the values of the frequencies in combination with their power values. For instance, low-amplitude instabilities in the outputs can be caused by environmental conditions that are temporally near zero. When these instabilities must be computed correctly, certain frequencies with low power values become important. In this case, one could augment the previously described selection with the selection of high frequencies that still have power values above and below a minimum and maximum threshold.

Normalization of the values

During the second step, the power values of V_{lp} are normalized, so that they become elements of the interval [0,1]. We have chosen to perform a pairwise normalization. For all possible pairs, and every time a new pair of simulations (V_{lp} and $V_{l'p}$) is formed, normalization is performed.

The normalization is relative to the largest power value of a pair of values, thus when $Vlp[i] > Vl'p[i]$ the normalized values become $vlp[i] = 1$ and $vl'p = Vl'p[i]/Vlp[i]$, otherwise they become $vlp = Vlp[i]/Vl'p[i]$ and $vl'p = 1$.

Because of this relativeness very small values will be regarded as being completely dissimilar. To avoid this, the similarity between two power values both below 10^{-3} is artificially set to 1, thus they are regarded as equal. The final result is that each pair of vectors of power values $(V_{lp}, V_{l'p})$ becomes a pair of fuzzy sets $(\nu_{lp}, \nu_{l'p})$, since all elements of $(\nu_{lp}, \nu_{l'p})$ have values between 0 and 1.

Similarity and distance computation

The similarity (S) between the fuzzy sets is computed using the family of similarity measures defined in Equation (7), with $x' = y = 1$ and $x = z = 0$, or with $x' = y = z = 1$ and $x = 0$. Since 1 is the neutral element of any commutative quasi-copula, and because the normalization always sets one value equal to one, any commutative quasi-copula results in the same expression:

$$S(\nu_{lp}, \nu_{l'p}) = \frac{\sum_{i=1}^{N} \min(\nu_{lp}(i), \nu_{l'p}(i))}{N} \tag{8}$$

The shorthand $S^p_{ll'}$ stands for $S(\nu_{lp}, \nu_{l'p})$, and after complementataion the final result is a distance matrix as shown in Figure 2. In order to come to a distance matrix for all outputs, the above distance matrix needs to be aggregated. To this end the largest dissimilarity of all outputs per pair of simulations is taken: $d_{ll'} = \max(d^1_{ll'}, \ldots, d^P_{ll'})$ (see Figure 3). Taking the maximum preserves the properties of a metric.

Clustering

Using the final distance matrix an UPGMA clustering with average linkage is performed. Figure 5 shows an example of

$Output_p$	Sim^p_1	Sim^p_2	$\cdots$	Sim^p_L
Sim^p_1	0	$1 - S^p_{12}$	$\cdots$	$1 - S^p_{1L}$
Sim^p_2	$1 - S^p_{12}$	0	$\cdots$	$1 - S^p_{2L}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Sim^p_L	$1 - S^p_{1L}$	$1 - S^p_{2L}$	$\cdots$	0

Figure 2 | The distance matrix for one $Output_p$: the smaller $d^p_{ll'} = 1 - S^p_{ll'}$, the more similar the solution trajectories of the outputs are.

	Sim_1	Sim_2	Sim_3	$\cdots$	Sim_L
Sim_1	0	$\max(d_{12}^1,\cdots,d_{12}^P)$	$\max(d_{13}^1,\cdots,d_{13}^P)$	$\cdots$	$\max(d_{1L}^1,\cdots,d_{1L}^P)$
Sim_2	$\max(d_{12}^1,\cdots,d_{12}^P)$	0	$\max(d_{23}^1,\cdots,d_{23}^P)$	$\cdots$	$\max(d_{2L}^1,\cdots,d_{2L}^P)$
Sim_3	$\max(d_{13}^1,\cdots,d_{13}^P)$	$\max(d_{23}^1,\cdots,d_{23}^P)$	0	$\cdots$	$\max(d_{3L}^1,\cdots,d_{3L}^P)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Sim_L	$\cdots$	$\cdots$	$\cdots$	$\cdots$	0

Figure 3 | The final distance matrix used to compare simulation trajectories.

a dendrogram for the Circle model, computed with a range of 30 different solvers and settings, and with moderate accuracies (step size: 10^{-5} or accuracies of 10^{-5}). More details on this example are given below.

EXPERIMENTS

We have tested this new methodology on 16 models: Foubert (Foubert *et al.* 2006), Hindmarsh (Zhabotinsky 2007), Influenza (Cellier 1991), Krogh (Shampine & Gordon 1975), PredatorPrey (Seppelt & Richter 2005), VanDerPol (van der Pol 1920), Circle, Damper, and on environmental models: Lambro (Benedetti & Sforzi 1999), Benchmark (Copp 2002), BioReactor (Dochain & Vanrolleghem 2001), Oxygen Uptake Rate (OUR), LLAS PS (Benedetti 2007), ASU, TwoASU, Galindo (Ayesa *et al.* 2006). A description of these models is beyond the scope of this article, and is presented in the work of Claeys (2008*b*). Environmental models are large systems that can have more than 500 derived variables and over 5000 (partly coupled) parameters.

The repository is filled with simulations that were computed with 30 different solvers and settings Adams-Bashforth 2 (Hairer *et al.* 1993), Adams-Bashforth 3 (Hairer *et al.* 1993), Adams-Bashforth 4 (Hairer *et al.* 1993), Rosenbrock (Shampine 1982), RK2a (Abramowitz & Stegun 1965), Midpoint (Kloeden & Platen 2000), Euler (Flowers 2000), Milne (Abramowitz & Stegun 1965), RK2b (Butcher 2003), Runge Kutta 4 (Kloeden & Platen 2000), Runge-Kutta-Fehlberg (Press *et al.* 1992), 10 variations of CVODE (Hindmarsh *et al.* 2005) (see Table 1), 6 variations of LSODE (Hindmarsh 1983) (see Table 1), DASSL (Petzold 1982), DASRT (Brenan *et al.* 1989), LSODA (Petzold 1983)), and with a moderate accuracy (10 − 5). The accuracy is set at 10 − 5 so that the overall simulation time remains acceptable, for the entire range of simulations. A description on the use of these solvers is beyond the scope of this article, and can be found in the Table 1 and 2.

Easy retrieval of the information needed to apply the methodology was possible because, after the simulation ended, each experiment was automatically saved in the repository.

Table 1 | List of abbreviations for solvers, used in the dendrograms. Short descriptions on the usage and meaning of these terms is given by Claeys (2008*b*)

Abbreviation	Full Name
Rosen	Rosenbrock
Midpo	Midpoint
Modif	Modified Euler
CVODE/A/F	CVODE/Adams/Functional
CVODE/A/N/D	CVODE/Adams/Newton/Dense
CVODE/B/F	CVODE/BDF/Functional
CVODE/B/N/D	CVODE/BDF/Newton/Dense
CVODE/A/N/B	CVODE/Adams/Newton/Band
CVODE/B/N/B	CVODE/BDF/Newton/Band
CVODE/B/N/S/M	CVODE/BDF/Newton/SPGMR/ ModifiedGS
CVODE/B/N/S/C	CVODE/BDF/Newton/SPGMR/ ClassicalGS
CVODE/A/N/S/M	CVODE/Adams/Newton/SPGMR/ ModifiedGS
CVODE/A/N/S/C	CVODE/Adams/Newton/SPGMR/ ClassicalGS
LSODE/A/F	LSODE/Adams/Functional
LSODE/B/F	LSODE/BDF/Functional
LSODE/A/N	LSODE/Adams/Newton/Dense
LSODE/B/N	LSODE/BDF/Newton/Dense
LSODE/A/D	LSODE/Adams/Newton/Diag
LSODE/B/D	LSODE/BDF/Newton/Diag

Table 2 | List of abbrviations for numerical methods, used in the dendrograms. Short descriptions on the usage and meaning of these terms is given by Claeys (2008*b*)

Abbreviation	Full Name
BDF	Backward Differentiation Formulas (integration)
Adams	Adams-Moulton (integration)
Functional	Functional iteration
Newton	Newton iteration
Dense	Full Jacobian matrix
Band	Diagonal treatment of Jacobian matrix
Diag	Diagonal treatment of Jacobian matrix
SPGMR	Preconditioned GMRES (a Krylov method)
GMRES	Generalized Minimal RESidual method
ClassicalGS	Classical Gram-Schmidt procedure
ModifiedGS	Modified Gram-Schmidt procedure

RESULTS

A special case: the circle model

Let us consider the Circle model, which is a special model with a simple analytical solution. Circle describes an undamped spring-mass system or a harmonic oscillator:

$$F = ma = m\frac{dx^2}{dt} = -kx \tag{9}$$

The system of differential equations of the model, with m and $k = 1$, is given by:

$$\frac{dx}{dt} = y \text{ and } \frac{dy}{dt} = -x \tag{10}$$

The analytical solution of the model is illustrated in Figure 4, for $x(0) = 0$ and $y(0) = 0$, and is given by:

$$x(t) = cos(t) \tag{11}$$

The final result of the methodology is a dendrogram, which represents the clustering information of the trajectories computed by the 30 solvers and their settings. As an illustration, the dendrogram for the Circle model is shown in Figure 5. From this figure it can be quickly concluded that, because the dissimilarity between the reference solvers is low (≤ 0.005), we can assume that these solvers have computed the same solution trajectories. In the remainder of this work, for each experiment, the cluster that contains all reference solvers that did not fail, is denoted as the reference cluster.

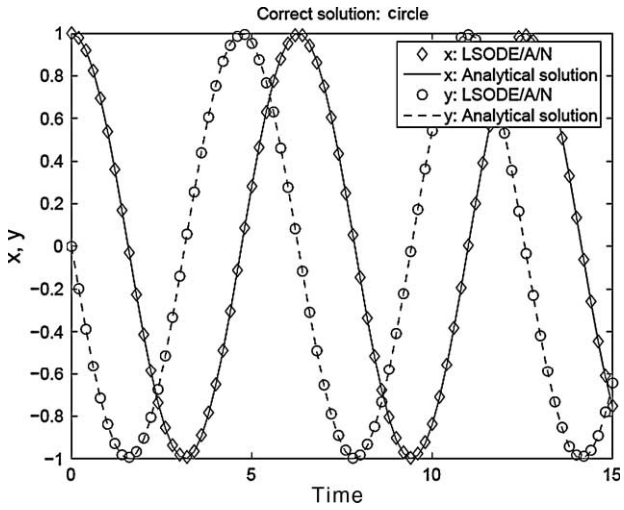

Figure 4 | The correct analytical solution of Circle compared to the solution of the LSODE/A/N solver, where x is defined as in Equation (10).

On the other hand, in Figure 5, the solvers CVODE/A/N/B and CVODE/B/N/B have a dissimilarity of 1, as compared to the rest of the solvers, and therefore it is concluded that they produce solution trajectories that deviate from all the other trajectories. Solvers that produce these diverging solutions are denoted as 'anomalous solvers' in the remainder of this article.

The anomalous solvers of this model detected by the methodology are shown in Figure 6. These are indeed strange combinations of solver settings, and detection of the anomalous solvers by inspection is easy, but this is not always the case for other models. What is important is that they were detected automatically, without having to look at all solution trajectories.

In order to arrive at a general interpretation of the dendrograms for all experiments, it was decided to use two threshold values. The first threshold helps to identify the anomalous solvers. This threshold was set to a dissimilarity of 0.2, which means that all solvers that group with the cluster that contains the reference solvers, at a distance above 0.2, are categorized as anomalous solvers (see Figure 5). A second threshold was used to help select the most appropriate solver, and was set to a rather small value of 0.05. All solvers that group with the cluster that contains the reference solvers below a distance of 0.05

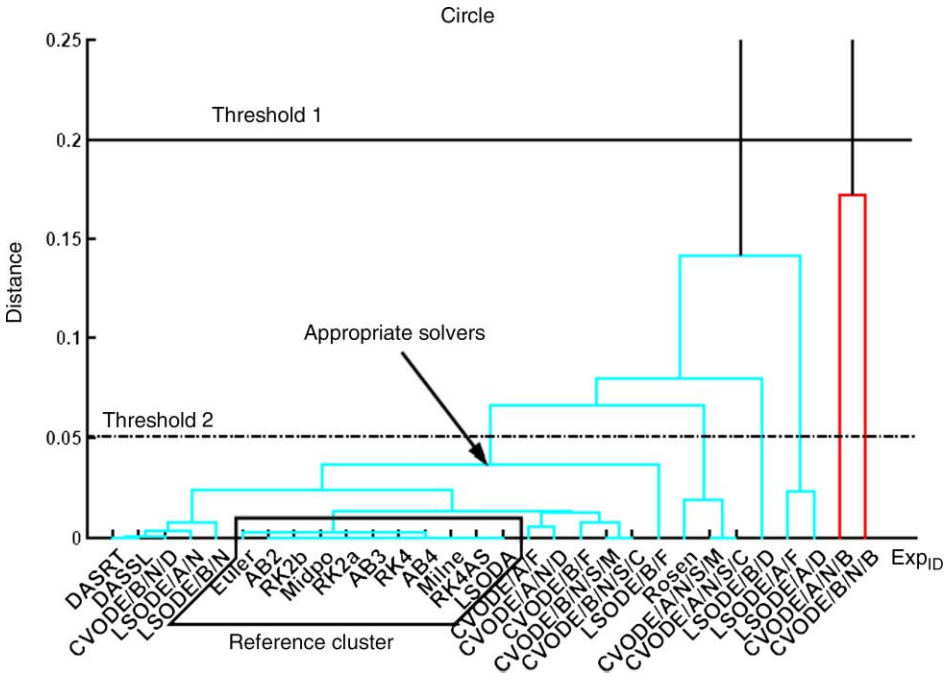

Figure 5 | Dendrogram of the Circle model, computed with a range of 30 different solvers. The unabbreviated terms of the solvers are shown in Tables 1 and 2. In this case Milne and LSODA fall into reference cluster, but these are not reference solvers.

are labelled as 'appropriate' for solving the model under consideration (see Figure 5). Within this group the fastest solver, i.e. the one with the lowest NrOfCS, is taken as the most suitable one. It is the most efficient solver that is able to compute the correct solution, and is denoted as the 'selected' solver in the remainder of this article.

In this manner LSODE/A/N was detected as the fastest solver for the Circle model, and we found indeed that is was appropriate for solving the model. The solution trajectory of LSODE/A/N, compared to the analytical solution is shown in Figure 4.

Results for the other models

At this stage of the study the interpretation of the dendrograms was done by hand and was not automated yet. However, complete automation of the methodology is possible and fine-tuning can be performed at that stage.

For the remaining models, analytical solutions were not available, but the correctness of the trajectories of the selected solver was confirmed by domain experts, while the anomalous trajectories were also confirmed. The detailed results of these confirmations are beyond the scope of this article and is available on request or in the work of (Claeys 2008*b*).

Next to the identification of the anomalous solvers, and failures, we also selected the most appropriate solver for each model. Figure 7 gives an overview of the 'fastest',

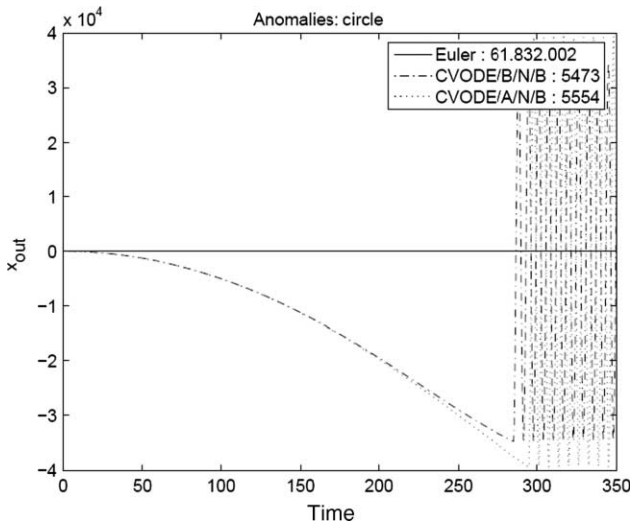

Figure 6 | The anomalous solvers of Circle compared to the solution of the Euler solver, which is an oscillation between −1 and +1 (see Figure 4). The numbers indicate the simulation time in NrOfCS (the higher the slower).

ModelName	Slowest	NrOfCS	Fastest	NrOfCS	Selected	NrOfCS	Strictest	NrOfCS
ASU	RK4	$2.8\ 10^6$	CVODE/A/N/D	101.984	CVODE/A/N/D	101.984	CVODE/B/F	203.615
Circle	RK4	$1.24\ 10^8$	LSODE/A/N	2740	LSODE/A/N	2740	LSODA	1.955.311
Foubert	RK4	$1.6\ 10^6$	LSODE/A/F	92	CVODE/B/N/B	122	LSODA	25.307
Galindo_OL	LSODA	$29.6\ 10^6$	LSODE/A/D	178.531	LSODE/A/D	178.531	LSODE/A/F	378.479
TwoASU	Rosen	$6.28\ 10^6$	LSODE/A/D	138.036	LSODE/A/D	138.036	LSODE/A/F	205.658
Damper	RK4	$12\ 10^6$	CVODE/A/N/D	439	CVODE/A/N/D	439	LSODA	48.980
Hindmarsh	RK4	$4\ 10^8$	LSODE/B/N	5009	LSODE/B/N	5009	LSODA	4.452.129
Influenza	RK4	$20.8\ 10^6$	Rosen	10.846	Rosen	10.846	LSODA	145.366
Krogh	RK4	$24\ 10^6$	CVODE/A/N/D	374	CVODE/A/F	435	LSODA	331.033
Lambro	RK4	$3.2\ 10^6$	CVODE/A/N/S/C	3042	CVODE/A/N/S/C	3042	LSODA	1.679.793
Benchmark	Rosen	$34.7\ 10^6$	LSODE/A/D	299.674	LSODE/A/D	299.674	CVODE/B/N/B	1.960.582
OUR	RK4	$24\ 10^6$	LSODE/A/N	170	LSODE/B/F	275	LSODA	239.158
BioReactor	RK4	$1.2\ 10^8$	LSODE/A/D	36	LSODE/A/D	36	LSODA	948.689
LLAS_PS	RK4	$4\ 10^7$	Euler	$1\ 10^7$	Euler	$1\ 10^7$	–	
PredatorPrey	RK4	$8\ 10^8$	LSODE/B/D	3344	Rosen	143.116	LSODA	6.350.864
VanderPol	RK4	$2\ 10^9$	LSODE/A/D	2993	LSODE/A/N	4370	–	

Figure 7 | The most appropriate solver ('selected') for the different test cases. The 'slowest', 'fastest', and 'strictest' solver are also shown. The NrOfCS is a measure for the computation time, the higher the slower. When the reference solvers do not cluster in an early stage of the clustering process, the strictest solver cannot be defined (–).

the 'slowest', the 'selected' and the 'strictest' solver. The 'strictest' solver is the first solver that joins the cluster of reference solvers. This means that the solution of the strictest solver lies the closest to the references, much closer than the solution of the selected solver. However, mostly the strictest solver is much slower than the selected solver. As an overview, Figure 8 illustrates the selected, the anomalous and failed solvers for each experiment.

DISCUSSION

From Figure 8 it is not possible to detect very strong trends in solver suitability. None of the solver's rows are completely or mainly black, which means that no perfect solver exists. Milne, CVODE/B/N/S/C, CVODE/B/N/S/M, CVODE/A/N/S/M, LSODE/B/D, CVODE/B/N/D, CVODE/A/N/B, LSODE/B/D, DASRT, and DASSL appear not to be good candidates since they failed and were anomalous in more than 25% of the test cases. Moreover, they were never selected.

A remark concerning the anomalous solvers is that many represent solvers with unusual combinations, such as Adams-Moulton with variations on a Newton iteration. Moreover, a BDF method combined with a banded or a preconditioned Newton iteration also appears frequently among the anomalous solvers. Nevertheless also solvers that have recommended combinations such as Adams-Moulton together with Functional iteration, or BDF with Newton iteration, are also part of the anomalous solvers. Even more remarkable is that all of these combinations have representatives among the selected solvers (see Figure 7). Thus a general rule that favors logical combinations of integration and iteration methods over unusual ones is not useful.

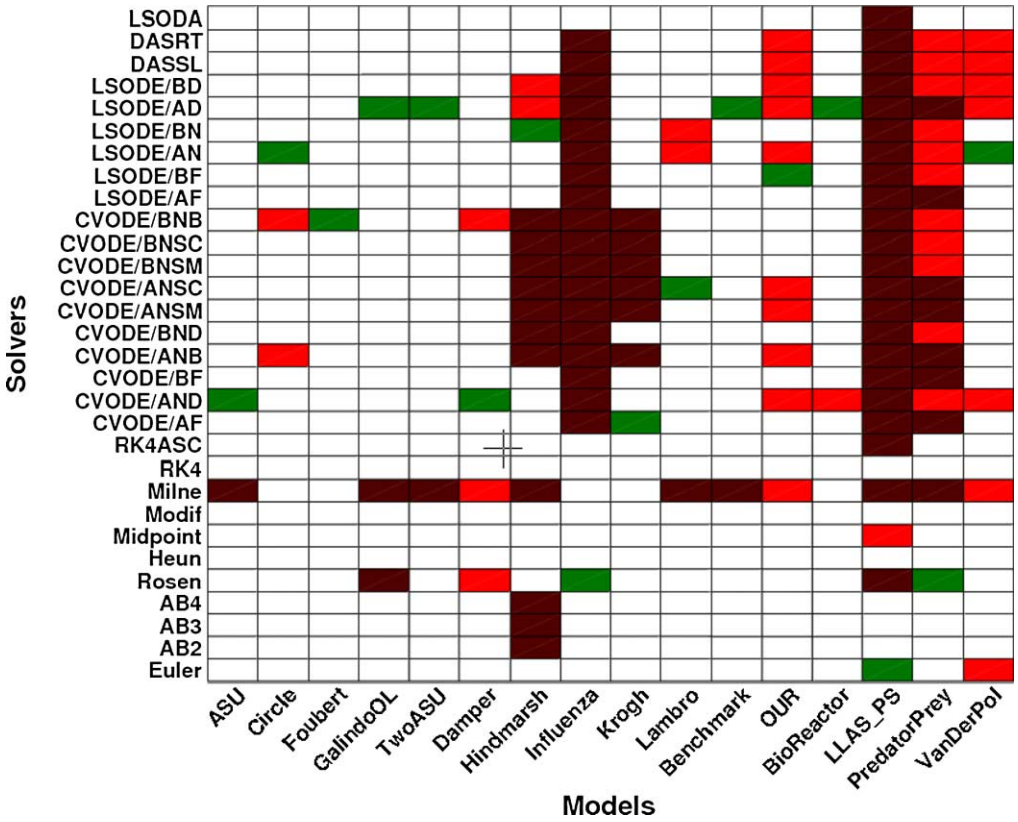

Figure 8 | An overview of the results. Red indicates failure: the solver fails without the generation of solution trajectories. Brown indicates an anomaly: an erroneous solution is generated. Green indicates the selected solver: the most appropriate solver.

Figure 8 shows that, among the non-reference solvers, only CVODE/A/F, CVODE/B/F, LSODE/A/F and LSODA never appear as anomalies, while they do become the selected or strictest solver. CVODE/A/F is fast, but it is selected only once, while CVODE/B/F is never selected and appears as the strictest solver only once. LSODA appears frequently as the strictest solver, but never as the selected solver, because it is too slow. LSODE/A/F appears as the strictest solver 3 times, but is never selected. From this we cannot conclude that these solvers are in general the best choices, and are perfect candidates for solving all test cases. Choosing a solver, like LSODA, that produces the most correct result most of the time, without taking into account its speed is not a good approach. Moreover, the number of models in the intersection between the fastest and the anomalies often is not zero, thus choosing the fastest solvers without taking into account the correctness of the solution is not a good approach either. The solution lies somewhere in the middle between very fast solvers and correct, but slow ones.

Another conclusion is that LSODE/A/N, LSODE/A/D, CVODE/A/N/D, and Rosenbrock are good choices, but we must not neglect that these solvers also appear frequently among the anomalous solvers. This clearly indicates that we are exploring the edge between fast solvers that still give satisfactory results, and fast solvers that are deficient. It is also noteworthy that frequently anomalous solvers are slower than the selected solver. Surprisingly, unusual combinations of solver settings can produce very good and fast simulation results.

Solvers like CVODE/A/N/B, CVODE/A/N/S/M, CVODE/B/N/S/M, CVODE/B/N/S/C, CVODE/B/F, CVODE/B/N/D, LSODE/B/D, LSODE/A/F, DASSL, DASRT and even LSODA are never selected. We should not discourage the use of these solvers, because we should keep in mind that when more test cases are used these

solvers could become selected in the future. Although most of these solvers fail a lot too, they are found to produce correct results in some cases, but they are never selected as the fastest one.

We should also mention that Milne is a solver that failed frequently, and when it was successful it was very slow or was anomalous. Therefore, based on these experiments, we can undoubtedly discourage the use of Milne. Milne might be anomalous because of implementation errors, thus because of high numerical coding uncertainty.

These findings show that deducing general rules on the use of solvers, based only on the characteristics of the solvers themselves is not possible. The most appropriate solver also depends on the model (experiment) under consideration. Therefore it is confirmed that the proposed methodology is important for practice.

Although we cannot derive general rules on the basis of solvers only, it is more than clear that RK4 is the most sluggish solver among all solvers tested. Choosing RK4, together with a moderate step size, will probably give a correct solution, but one must be very patient. Therefore we can certainly advise modellers not to use this rather popular solver in future research projects.

The trends based on the test cases are more significant, computations for Hindmarsh, Influenza, OUR, LLASP PS, PredatorPrey and VanDerPol fail or are incorrect for more than 20% of the solvers. For two of these models, the slow solver Rosenbrock is selected, and the solver LSODE/A/D is always anomalous. On the contrary, for the other test cases (ASU, Circle, Foubert, GalindoOL, TwoASU, Damper, Benchmark, BioReactor, Krogh, Lambro) LSODE/A/D is a good choice, since it appears as the selected solver four times, and is not anomalous for any of these experiments.

ASU, TwoASU, Benchmark and GalindoOL are bioprocess models composed of the same building blocks, thus they have several similar properties. It is noteworthy that their anomalies are rather similar, and that LSODE/A/D appears as the selected solver for three of these models, while it is a good choice for ASU.

Similarities between experimental specifications that influence the outputs of the model can also play an important role in the selection of the most appropriate solver. Dynamic input data is such an experimental specification and in the future similarities between these data could be computed using the family of similarity measures proposed in this work (Equation (8)).

The time it took to apply this methodology per test case was an average of 10 seconds. Thus, when this methodology is implemented in an efficient manner, it can be run every night using the cases available in the repository. Test cases with different accuracies could become available and predictions on the choice of the best solver for each test case are then refreshed daily. This updating process is very useful, because the results depend on the test cases that reside in the repository, that is continuously extended.

Finally, Figure 7 shows that choosing the most appropriate solver can indeed speed up computation time with a factor that ranges from 10 up to 10^6. This factor of improvement can serve as an encouragement to support further research into this topic.

NUMERICAL UNCERTAINTY SCORE

The dendrograms, presented above, could help identifying the degree of numerical solver uncertainty that characterizes an experiment. The lower the distances at which clusters are formed, the lower the numerical solver uncertainty. The higher these distances are, the higher the numerical solver uncertainty. It can therefore be stated that automatic solver selection can reduce numerical solver uncertainty in future simulations.

A simple method to assess the numerical uncertainty related to a mathematical method, that an experiment exhibits, is to compute 1 minus the fraction of solvers that correctly computed the model for a certain accuracy. Table 3 lists the numerical uncertainty score for the test cases in ascending order. The lower the score, the lower the numerical uncertainty the experiment is exposed to.

If the accuracy is satisfactory, then the (%) of numerical solvers that fail to compute the correct solution can be an indication of the numerical uncertainty related to coding errors in the numerical solvers, or in the model. Again, when the accuracy is satisfactory, the number of anomalous solvers (see 2nd column of Table 3) is an indication of how sensitive the initialized model is to the suitability of the mathematical methods of the numerical solvers. The lower

Table 3 | A simple manner to quantify uncertainty related to the numerical solver, on a per model basis. The total number of solvers is 30, the number of solvers that failed or were anomalous are shown, and 1 minus the (%) of solvers that was able to compute the model correctly is the Numerical Uncertainty Score (NUS)

Model Name	Failed	Anomalous solvers	NUS (%)
Foubert	0	0	0
ASU	1	0	3.3
BioReactor	0	1	3.3
Benchmark	1	0	3.3
TwoASU	1	0	3.3
Galindo	2	0	6.7
Circle	0	2	6.7
Damper	0	3	10
Lambro	1	2	10
Krogh	6	0	20
VanderPol	0	7	23
OUR	0	10	33
Hindmarsh	11	2	43
Influenza	18	0	60
PredatorPrey	8	11	63
LLAS_PS	22	1	76

the number of anomalous solvers, the lower this sensitivity, and the lower the numerical uncertainty related to the suitability of the solver for that model.

CONCLUSIONS AND FUTURE WORK

In this study it has been observed that several initialized models have largely dissimilar solution trajectories, when computed with different solvers and settings. Hence, numerical solver uncertainty is not a rare phenomenon. Moreover, proper solver selection can shorten simulation time with a factor up to 10^6, and it is not a straightforward task for the unexperienced modeller to find this selection. It can therefore be concluded that automatic solver selection can reduce numerical solver uncertainty in future simulations, and in this work it was shown that repositories, containing numerous simulations, are suitable for the process of automatic solver selection.

The methodology proposed in this article is capable of detecting deviations from the reference solutions, and could be run during idling time of servers. In this way the solver selection process would be updated daily, according to the available test cases.

It is known since long that the fastest solvers are not always the best candidates, since they can produce wrong results. Taking only correctness into account while evaluating solvers, by selecting the strictest solver as the most appropriate one, is not a good approach either, since mostly this solver is very slow. The most appropriate solver can be found in between solvers that exhibit fast computations and solvers that produce correct results.

Deriving general rules solely on the basis of the solver properties is not possible, as it was shown that none of the solvers is perfect for solving all models. One cannot discourage the use of certain solvers. Similarities between models seem however to be important in the selection of the most appropriate solver. In the future it may therefore be possible to extract general rules based on the properties of models and experimental specifications in combination with the properties of solvers. Certainly, the following holds: the more test cases are available, the more we are able to derive general rules, based on solvers and models.

ACKNOWLEDGEMENTS

Peter Vanrolleghem holds the Canada Research Chair in Water Quality Modelling.

REFERENCES

Abramowitz, M. & Stegun, I. A. 1965 *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Courier Dover Publications, New York, USA.

Ascher, U. M. & Petzold, L. R. 1998 *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, Philadelphia, USA.

Ayesa, E., Grau, P., Sagarna, J. M., Salterain, A., de la Sota, A. & Suescun, J. 2006 Supervisory control strategies for the new WWTP of Galindo-Bilbao: the long run from the conceptual design to the full-scale experimental validation. *Water Sci. Technol.* **53**(4–5), 193–201.

Benedetti, L. 2007 *Probabilistic Design and Upgrade of Wastewater Treatment Plants in the EU Water Framework Directive Context*. PhD thesis, Ghent University, Ghent, Belgium.

Benedetti, L. & Sforzi, F. 1999 *Dynamic Integrated Modelling: A case Study on the Lambro Catchment*. Msc. thesis, Ghent University, Ghent, Belgium.

Brenan, K., Campbell, S. & Petzold, L. 1989 *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Society for Industrial Mathematics, New York, USA.

Bunus, P. 2007 Automating the numerical solver selection. *Journal Simulation News Europe* **17**(1).

Butcher, J. C. 2003 *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, Queensland.

Cameron, I. & Hangos, K. 2001 *Process Modelling and Model Analysis*. Academic Press, London.

Cellier, F. E. 1991 *Continuous System Modeling*. Springer-Verlag New York, Inc, New Jersey, USA.

Claeys, F. 2008*a A Generic Software Framework for Modelling and Virtual Experimentation with Complex Environmental Systems*. PhD thesis, Ghent University, Ghent, Belgium.

Claeys, P. 2008*b Towards Automatic Numerical Solver Selection for Hierarchical Bioprocess Models*. PhD thesis, Ghent University, Ghent, Belgium.

Claeys, P., De Baets, B. & Vanrolleghem, P.A. 2008 On numerical solver selection and related uncertainty terminology. Submitted to *Journal of Hydroinformatics*.

Copp, J. B. 2002 *The COST Simulation Benchmark: Description and Simulator Manual*. Office for Official Publications of the European Communities, Luxembourg.

Dawyndt, P., De Meyer, H. & De Baets, B. 2006 UPGMA clustering revisited: a weight-driven approach to transitive approximation. *Int. J. Approx. Reason.* **42**(3), 174–191.

De Baets, B. & De Meyer, H. 2005 Transitivity-preserving fuzzification schemes for cardinality-based similarity measures. *Eur. J. Oper. Res.* **160**(3), 726–740.

De Baets, B., De Meyer, H. & Naessens, H. 2002 A class of rational cardinality-based similarity measures. *J. Comput. Appl. Math.* **132**(1), 51–69.

Dochain, D. & Vanrolleghem, P. A. 2001 *Dynamical Modelling and Estimation in Wastewater Treatment Processes*. IWA Publishing, London.

Flowers, B. H. 2000 *An Introduction to Numerical Methods in C++*, Revised Edition. Oxford University Press, Inc, New York, USA.

Foubert, I., Dewettinck, K., Janssen, G. & Vanrolleghem, P. A. 2006 Modelling two-step isothermal fat crystallization. *J. Food Eng.* **75**(4), 551–559.

Genest, C., Molina, J. J. Q., Lallena, J. A. R. & Sempi, C. 1999 A characterization of quasi-copulas. *J. Multivar. Anal.* **69**(2), 193–205.

Hairer, E., Nørsett, S. P. & Wanner, G. 1993 *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer Verlag, New York, USA.

Hindmarsh, A. C. 1983 Odepack: a systematized collection of ode solvers. *IMACS Trans. Sci. Comput.* **1**, 55–64.

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E. & Woodward, C. S. 2005 Sundials: suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Softw.* **31**(3), 363–396.

Houstis, E. N., Catlin, A. C., Rice, J. R., Verykios, V. S., Ramakrishnan, N. & Houstis, C. E. 2000 Pythia-II: a knowledge/database system for managing performance data and recommending scientific software. *ACM Trans. Math. Softw.* **26**(2), 227–253.

Kamel, M. S., Enright, W. H. & Ma, K. S. 1993 Odexpert: an expert system to select numerical solvers for initial value ode systems. *ACM Trans. Math. Softw.* **19**(1), 44–62.

Kloeden, P. E. & Platen, E. 2000 *Numerical Solution of Stochastic Differential Equations*. Springer, Germany.

Petcu, D. & Dragan, M. 2000 *Designing an ODE Solving Environment*. Springer, Verlag, Berlin.

Petzold, L. R. 1982 *A description of dassl: A differential/algebraic system solver*. Technical Report SAND82-8637, Sandia National Laboratories, Livermore.

Petzold, L. R. 1983 Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *J. Sci. Stat. Comput. (SIAM)*, 136–148.

Press, W. H., Teukolsky, S. A., Vetterling, T. & Flannery, B. 1992 *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edition. Cambridge University Press, New York, NY, USA.

Seppelt, R. & Richter, O. 2005 It was an artefact not the results–a note on system dynamic model development. *Environ. Model. Softw.* **20**, 1543–1548.

Shampine, L. F. 1982 Implementation of Rosenbrock methods. *ACM Trans. Math. Softw.* **8**(2), 93–113.

Shampine, L. F. & Gordon, M. K. 1975 *Computer Solution of Ordinary Differential Equations*. W.H. Freeman & Co Ltd, San Francisco, USA.

van der Pol, B. 1920 A theory of the amplitude of free and forced triode vibrations. *Radio Rev.* **1**, 701–710.

Weerawarana, S., Houstis, E. N., Rice, J. R., Joshi, A. & Houstis, C. E. 1996 Pythia: a knowledge-based system to select scientific algorithms. *ACM Trans. Math. Softw.* **22**(4), 447–468.

Zhabotinsky, A. 2007 Belousov-Zhabotinsky reaction. *Scholarpedia* **2**(9), 1435.