The Journal of Systems and Software 163 (2020) 110517

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Cyber-physical modelling in Modelica with model-reduction techniques

Anton Sodja, Igor Škrjanc, Borut Zupančič*

University of Ljubljana, Faculty of Electrical Engineering, Tržaška 25, 1000 Ljubljana, Slovenia

ARTICLE INFO

Article history:
Received 31 August 2018
Revised 27 September 2019
Accepted 31 December 2019
Available online 17 January 2020

Keywords:
Continuous systems modelling
Object oriented modelling
Realisation preserving modelling
Model reduction
Ranking metrics
Modelica

ABSTRACT

Object-oriented modelling of cyber-physical systems with Modelica and similar environments has brought many advantages, especially the efficient re-use of models and thus the possibility of creating powerful multi-domain libraries. Unfortunately, the models have become highly complex, which causes serious problems during processing and execution. Consequently, verification and debugging is becoming an increasingly challenging task. The continuous investigation of simplifications and reductions in all phases of model developments is thus urgent.

The present paper deals with reduction methods based on metric ranking and preserve realisation, which means that the structure and the parameters of the model remain physically interpretable. Two model-reduction methods are described and implemented in Open Modelica. The first operates on a set of differential-algebraic equations, and the second is based on modified bond-graphs-reduction techniques. The latter approach is suitable for component-based models in Modelica that are usually represented graphically with object diagrams. The paper briefly describes the research area, the problems of the adoption of the developed model reduction techniques to the Modelica environments, and the final implementation. Both proposed approaches are tested on the model of a car suspension system and briefly discussed.

1. Introduction

Modelling is an iterative process. In the early stages of model development, relatively simple conceptual models are created, which can answer only a few general design questions and usually have low accuracy (Murray-Smith, 2009; Cellier, 1991; Matko et al., 1992). Later, as more data and more knowledge about the modelled system become available, the accuracy of the model is frequently improved by including more details in it. However, a detailed model leads to complexity, which can cause serious problems during processing and/or execution. Therefore, models should regularly be simplified in order to remain useful and usable.

This latter requirement is more difficult to achieve in a contemporary component-based approach to modelling, in which a model is built up of previously prepared building blocks (i.e., components or submodels). Such an approach is especially efficient when using components from different fields (e.g., multi-domain, cyber-physical models) (Modelica Association, 2010; Modelica Association, 2012; Open Source Modelica Consortium, 2012; Fritzson, 2014; Sodja, 2012; Zupančič and Sodja, 2013).

The advantage of this approach is manifold:

- rapid model development – detailed models can be built from well-tested components; only the component interfaces need to be specified, and the detail design of each component can be abstracted away;
- models are more transparent and understandable - it is extremely important for an efficient collaboration inside an interdisciplinary group;
- it is easier to verify and validate the overall model because components have been tested previously.

However, component-based modelling can also bring some disadvantages:

- it is more tempting to build an overly complex model, which can also make verification and validation more difficult;
- it is more difficult to solve numerical problems, which usually arise suddenly and unpredictably when an appropriate level of complexity is reached;

* Corresponding author.
E-mail address: borut.zupancic@fe.uni-lj.si (B. Zupančič).

https://doi.org/10.1016/j.jss.2019.110517
0164-1212/

- not all modelling issues can be addressed during the component-models' design; the modeler must often possess advanced knowledge about the modelling methodologies and suppositions used in the components' design, so that the components are not used in a way that the original developer never intended; otherwise, the model might be physically inconsistent or numerical difficulties may appear during the simulation.

It is thus very important that the component model library (i.e., the collection of component models) be adequately documented and that the most common pitfalls when using the components from the library be described. Nevertheless, the increasing complexity of models used in contemporary engineering applications requires more than just exhaustive documentation. Modelling environments should include various tools to help the user assess the quality of the model (e.g., visualisation tools).

Finally, a model should not be more complex than required for the given task. This goal is difficult to achieve in component-based modelling. One approach to identify overly-detailed parts of the model is to use model-reduction techniques that prune the components of the model that have negligible effects on the salient model dynamics. Although the model-reduction techniques of dynamic-systems' models are an active research topic, there are hardly any tools to be found in industrially relevant modelling environments, such as those supporting the modelling language Modelica. The major reason for this is the heterogeneity of the models (e.g., the different formulation of the sub-models, which impairs the creation of efficient automated model-reduction tools.

In the rest of this paper, we describe our approach to creating such a tool for model reduction that is applicable for a large number of complex cyber-physical models implemented in the Modelica modelling language. Section 2 classifies reduction methods into several categories. A special class of model-reduction methods that retain the formulation (i.e., realisation) of the original model are introduced. In the final part, some facts regarding Modelica models and model reduction techniques are presented.

Section 3 is the main contribution of the paper and describes the reduction of systems defined with differential-algebraic equations (DAE). Elementary reduction operations are introduced, and the problems with the solvability and numerical stability of reduced systems are discussed. The ranking procedure is the central part of the proposed approach. We present ranking with a one-step solver and ranking by statistical properties. The implementation in Open Modelica is then described. The section concludes with an example of a car suspension system using both ranking procedures. The results confirm the success of the proposed procedures.

As at least on higher hierarchical levels, the model components in Modelica are usually described using object diagrams Section 4 describes the reduction possibilities for such models. In this case, energy-based metrics are used. Unfortunately. energy flows are not explicitly available in Modelica components but can be calculated from the information available in component connectors. Therefore, we added the instrumentation phase in the model translation procedure. Here, the equations for energy flows calculations are added, which enables the metrics and ranking calculations after translation and simulation. The same example as in Section 3 was used.

Both approaches produced similar results.

2. Model reduction techniques

Engineers use experience and intuition to determine the important parts of a model that have the highest impact on a system's dominant dynamics or on the model's simulation response in a specific scenario. In an attempt to diminish the reliance on subjective factors, such as experience, numerous model-simplification and reduction methods have been developed (Ersal et al., 2008).

In this paper, a clear distinction is made between model simplification and model reduction. The simplification indicates that modifications to a model perform more concise representations but the behaviour of the original and the simplified models are exactly the same (e.g., the algebraic elimination of an algebraic loop or the simplification of a block scheme of a control system using the algebra of block schemes). In contrast, the reduction procedure omits some less interesting aspects of the behaviour of the original model. Hence, the reduction only represents a portion of the behavior of the original model (e.g., parts of the model can be exchanged with simpler ones, such as a lower order or a linearised submodel). Therefore, the resulting reduced model is thus valid only within a limited range.

2.1. Classification of model-reduction techniques

Most model-reduction techniques consist of running a series of simulations, ranking the individual terms, elements or components with the appropriate metric based on the results obtained by simulation and approximating (or removing) those parts that fall below a certain threshold (Chang et al., 2001). The choice of the ranking metric and the elementary reduction operations depends on how the model is formulated and may be limited by the modelling domains. Based on the principles on which the reduction methods operate, they can be classified into three categories (Ye, 2002):

- methods based on a mathematical manipulation of the relations in the model;
- methods based on a physical interpretation of the system's dynamics;
- methods that are a combination of the previous two items.

In contrast, Ersal et al. (2008) classify model-reduction techniques into frequency-, projection-, optimisation-, and energy-based categories. Frequency-based reduction techniques aim at pruning the dynamics outside of some frequency range, e.g., removing 'fast' dynamics. Similarly, projection-based techniques are used in an attempt to find a lower dimensional portion of the system's state space (a subspace) that describes the salient dynamics adequately. Optimisation-based techniques are a more formal approach to obtain an optimal reduced model subject to a complexity constraint. Finally, energy-based model-reduction techniques presume that the components associated with a small energy flow also have a small effect on the dominant system's dynamics. However, the classifications are not strict. In other words, a certain method may belong to several categories.

2.2. Preservation of realisation

A special class of model-reduction methods are those that retain the formulation (i.e., realisation) of the original model, which are so-called realisation-preserving model-reduction methods.

Preservation of realisation means that the structure and parameters of the reduced model must remain physically interpretable. This is an important property in determining overly-detailed parts of the model. The reduced models can be simulated using the same tool as the original model (which in the case of non-realisation-preserving methods is not always true (Sheehan, 1999)). Preservation of realisation is extremely useful for more efficient verification purposes.

However, most model-reduction techniques do not preserve realisation, because such a requirement hinders their performance. If the formulation of the model is allowed to change, a much greater reduction of the model's complexity can be achieved, so that the

dissimilarity of the full and reduced model behaviours falls within the prescribed error bounds.

2.3. Realisation-preserving model-reduction methods and Modelica

To the best of the authors' knowledge, there were no realisation-preserving model-reduction methods developed for models implemented in Modelica. Nevertheless, corresponding techniques were developed for a modelling formalism that supports the formulation of models using similar concepts as these in Modelica (Modelica Association, 2010; Modelica Association, 2012; Open Source Modelica Consortium, 2012; Fritzson, 2014).

Modelica is a language for modelling complex physical systems with (hybrid) differential-algebraic equations (DAE). For the purpose of structuring large models, many object-oriented concepts are supported by the language (e.g., inheritance, encapsulation, etc.). Especially notable are the *connections*, which are a convenient way to avoid explicitly stating the structural equations (e.g., Kirchhoff's laws) in order to describe the relations between submodels. A set of connected submodels can be presented graphically with a schematic diagram (also called an object diagram).

Large complex models in Modelica are normally hierarchically composed: at the lowest levels, the equations are entered directly, and at higher hierarchical levels, a graphical description with object diagrams is preferred. Therefore, two kinds of reduction techniques are needed:

- the reduction techniques for the differential-algebraic equations;
- the reduction techniques for object diagrams.

3. Reduction of differential-algebraic equations systems

The most general representation of a dynamical system is the differential algebraic equation (DAE)

$$\boldsymbol{F}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{y}, t) = 0 \tag{1}$$

where $\boldsymbol{x}$ is the vector of systems states, $\dot{\boldsymbol{x}}$ is the vector of corresponding derivatives, $\boldsymbol{y}$ is the vector of algebraic variables, and t is the independent variable.

Modelica is based on equations (models that can be implemented in it must be described by equation systems) and, therefore, all the Modelica language elements are mapped to differential, algebraic, and discrete equations. It is thus often useful to inspect the equations directly, for example, to verify the model. Nevertheless, the resulting derived equation system is often too large and intricate, even for relatively small models, to be readable by humans or manually investigated.

Another problem is that efforts to make the component models more modular and extendable often lead to an intricate implementation: the equation systems (i.e., its parts) are cluttered into many incomplete (partial) component models. These partial models are then combined into a complete component by either inheritance or replaceable submodels (i.e., submodels that are defined during the declaration of the model). The partition of an equation system into several partial (sub)models entails the use of many auxiliary variables that impair the clarity of the flattened model (i.e., the model translated into a single equation system).

For a manual investigation of the model's underlying equation systems, it would thus be preferable to post-process them before showing them to the user.

There are numerous equation-system reduction techniques that are routinely used in a manual derivation of system equations. However, most of them are not appropriate for automated model reduction due to their dependency on the context (i.e., knowledge of the modelled system's physics). Then there are also more formal, model (order) reduction methods that are applicable only to certain kinds of DAEs (e.g., Lagrange's equations Chang et al., 2001 or explicit space-state realisations Kokotovic and Sannuti, 1968). Regardless of the level of formalisation, the same strategies are used to reduce the equation system: elementary constituents of the equation set (in most cases, variables and/or equation terms) are either eliminated (i.e., set to zero) or replaced with simpler constituents (e.g., constants).

In our toolbox (Sodja and Zupančič, 2012; Sodja, 2012), we used an approach originally developed for the model reduction of nonlinear DAEs entitled *Behavioural model generation (BMod-Gen)*(Borchers, 1997; Wichmann, 2004; Sommer et al., 2008). Equations were generated from a higher-level description (i.e., netlist) of electrical circuits. The method refrains entirely from making any supposition about the physics of the modelled system. A decision about which elemental reduction operation should be performed depends entirely on an estimation of how much the considered operation will change the behaviour of the model in the specific simulation experiment(s). A clear disadvantage of this approach is that the model can be rendered nonphysical or even unsolvable (which indeed often happens). As will be shown in this section, the most complicated part of the method (besides the error estimation introduced to the model by elemental reduction operations) is the elimination of illegal reduction operations.

3.1. Elementary reduction operations

There are several ways of reducing a model while preserving its realisation; however, mostly they are either removing some components of the model or replacing them with simpler ones. In the equation system, these components can consist of state variables, state derivatives, and algebraic variables. The basic DAE expression (see Eq. (1)) is usually composed of several (n) terms (unless it is a trivial equation, e.g., $x = y$)

$$t_1(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{y}) + \cdots + t_j(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{y}) + \cdots + t_n(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{y}) = 0 \tag{2}$$

$t_j(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{y})$ designates expressions, i.e., terms, of the variables $\boldsymbol{x}$, $\dot{\boldsymbol{x}}$, and $\boldsymbol{y}$. An elementary reduction acting on term $t_j(\cdot)$ in Eq. (2) could thus delete it or replace it with a simple expression, i.e., a constant value or a linearised expression.

Because elementary-reduction operations can be mutually exclusive, a priority of operations must be established. In most cases, operations achieving better reduction of the model are preferred (e.g., the deletion of terms before a replacement with a constant value).

In our toolbox, we implemented only term deletion and replacement with constant values, due to their being identified in the literature as the most effective methods (Wichmann, 2004).

3.2. Solvability and numerical stability

Reduction operations that remove an equation term or substitute it with a constant cannot be applied on arbitrary equation term because they can cause the equation system to become unsolvable. This is prevented by checking the reduction operation's effect on the solvability of the system before reduction operation is applied. A further problem is assuring that it will be possible to initialise the reduced model. In Modelica, the initialisation problem differs from the simulation problem. It comprises the equations of the simulation problem and possibly additional initial equations. Derivatives are decoupled from the variables, i.e., they are considered as algebraic variables, and a variable may have a fixed initial value, i.e., it is considered a constant in initialisation problem, or only a guess value. Therefore, if a variable has a fixed non-zero initial value and is neglected by the reduction algorithm, i.e., it is set to zero in the simulation problem, the initialisation problem

becomes unsolvable. In contrast, if the initial value of the variable is also updated by the reduction algorithm, the solution of the initialisation problem might differ from the solution of the original substantially. An approach presented by Wichmann (2003) is to assert that the reduction operations do not increase the structural index of the DAE system. However, this degrades performance of the model reduction method because many reasonable model reductions increase the structural index of the system. Tools supporting Modelica are able to perform automatic DAE index reduction; therefore, an increase of the index should be admissible. Moreover, if the initialisation problem consists of additional equations that are not present in the simulation problem, the accordance of these equations with the reduced equation system of the simulation problem must be assured.

Again, the most reliable way of ensuring the solvability and stability of the systems is to perform the full simulation after applying each reduction operation. However, only heuristic tests are practically useful. For example, it may be required that the reduction does not change certain structural properties of the equation system, while those structural properties are characterised by various DAE indices (Wichmann, 2004).

In contrast, we implemented few restrictions in Modelica for the reduction operations used the same algorithms as used by the Modelica translator (Bunus, 2004) to verify whether the equation system is well posed (i.e., if there is the same number of equations and variables, whereby the latter can be uniquely assigned to the equations from which shall be solved).

3.3. Ranking procedures

3.3.1. Ranking by one-step solver

The exact error introduced by each reduction operation can be acquired by simulating the model that was previously reduced by a corresponding elementary operation. This yields a perfect ranking but, due to the enormous computational effort required, it does not have any practical value and can be used only for verification of more practical approaches.

However, a computationally non-demanding estimate of how much a solution $[\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}]$ of the reduced model $\tilde{\boldsymbol{F}}$ would diverge from the solution $[\boldsymbol{x}, \boldsymbol{y}]$ of the full model $\boldsymbol{F}$, i.e.:

$$\boldsymbol{F}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{y}, t) = 0, \tag{3}$$

can be obtained by inserting the values of the full-model variables (simulation results) at several time instants (e.g., selected randomly or equidistantly) into the set of reduced equations:

$$\tilde{\boldsymbol{F}}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{y}, t) = \boldsymbol{\delta} \tag{4}$$

This yields a residual vector $\boldsymbol{\delta}$ that indicates how similar the reduced and full models are: small changes to the original equation system result in a smaller residual vector.

However, the error ranking based on the residual vector is mostly of poor quality (Wichmann, 2003) and, therefore, a further improvement was introduced by Wichmann (2003), in which the exact solution vector $[\boldsymbol{x}, \boldsymbol{y}]$ is taken as an initial estimate for the Newton-Raphson numerical iteration that is used to estimate the solution vector $[\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}]$ of the reduced equation at the time instant t_k:

$$\begin{bmatrix} \boldsymbol{x}_k^* \\ \boldsymbol{y}_k^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{y}_k \end{bmatrix} - \mathbf{J}_{\tilde{F}}^{-1}(\boldsymbol{x}_k, \dot{\boldsymbol{x}}_k, \boldsymbol{y}_k) \cdot \tilde{\boldsymbol{F}}(\boldsymbol{x}_k, \dot{\boldsymbol{x}}_k, \boldsymbol{y}_k) \tag{5}$$

In Eq. (5), $\dot{\boldsymbol{x}}_k$, $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$ are solution of the original system, $\tilde{\mathbf{F}}$ and $\mathbf{J}_{\tilde{F}}$ are modified system and its Jacobian matrix respectively while $\boldsymbol{x}_k^*$ and $\boldsymbol{y}_k^*$ are corresponding estimates of the solution $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$ of the simplified system.

Usually, only one iteration of Newton-Raphson method is implemented as it is normally sufficient for the appropriate estimation of the solution of the simplified system. In general, several iterations can also be implemented.

Finally, predicted error ϵ_k of the modification, i.e., reduction operation, at time t_k is calculated:

$$\epsilon_k = \left|\left| \begin{bmatrix} \boldsymbol{x}_k^* \\ \boldsymbol{y}_k^* \end{bmatrix} - \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{y}_k \end{bmatrix} \right|\right| = \left|\left| \mathbf{J}_{\tilde{F}}^{-1}(\boldsymbol{x}^*, \dot{\boldsymbol{x}}^*, \boldsymbol{y}^*) \cdot \tilde{\boldsymbol{F}}(\boldsymbol{x}^*, \dot{\boldsymbol{x}}^*, \boldsymbol{y}^*) \right|\right| \tag{6}$$

In Eq. (6), $||\cdot||$ denotes a weighted infinity norm. The weight of each vector's component is the inverse root-mean-square value of the component that is calculated from the reference simulation results.

In Eq. (6), the evaluation of $\tilde{\boldsymbol{F}}$ is straightforward, whereas the Jacobian matrix $\mathbf{J}_{\tilde{F}}$ requires the calculation of $\partial \dot{x}_i / \partial x_i$ which is not present in reference simulation results. However, the estimate of the value can be obtained from the derivative-approximation formulas. For example, a general n-th order backward-differentiation formula (BDF) is given by:

$$\dot{\boldsymbol{x}}_k = \frac{1}{h}\left(a_0 \cdot \boldsymbol{x}_k + \sum_{l=1}^{n} a_l \cdot \boldsymbol{x}_{k-l}\right) \tag{7}$$

In Eq. (7) a_0, a_1, $a_2 \ldots a_n$ are appropriate coefficients of BDF, and h is the step size. If Eq. (7) is differentiated with respect to $\boldsymbol{x}_k$, a constant $\bar{h} = a_0/h$ is obtained that depends on the order of BDF and the corresponding step-size. The results of the ranking are strongly dependent on the choice of h. The literature gives no guidance on how to select this value. Therefore, we set it by trial and error. In most cases, it was set to the fraction of the fastest time constant (e.g., one fifth).

The computational cost of the one-step-solver ranking can be further substantially reduced by using the Sherman-Morisson theorem, given by Eq. (8), which requires only one inversion of the Jacobian matrix for each considered time instant, regardless of the number of reduction operations ranked.

$$\mathbf{J}_{\tilde{F}}^{-1} = \mathbf{J}_F^{-1} - (1 + \boldsymbol{v}^T \mathbf{J}_F^{-1} \boldsymbol{e}_l)^{-1} \mathbf{J}_F^{-1} \boldsymbol{e}_l \boldsymbol{v}^T \mathbf{J}_F^{-1} \tag{8}$$

In Eq. (8), $\boldsymbol{e}_l$ is a vector with all the components set to zero, except the lth component, which is set to one. The equation system was modified in the lth equation. The vector $\boldsymbol{v}$ is a gradient of the difference between the modified and the original equation. For example, if the jth term was removed from Eq. (2) (or replaced with a constant)then the vector $\boldsymbol{v}$ is:

$$\boldsymbol{v} = \nabla t_j(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{y}) = \frac{\partial t_j(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{y})}{\partial [\boldsymbol{x}^T, \boldsymbol{y}^T]^T} \tag{9}$$

More details can be found in Sodja (2012).

3.3.2. Ranking by statistical properties

A simple, intuitive, and computationally non-demanding ranking algorithm can be conceived by observing the statistical properties of the terms in the equations.

For example, the mean value and the standard deviation of the jth term's value (Eq. (2)) are calculated as:

$$m_j = \frac{1}{n}\sum_{k=1}^{n} t_j(\dot{\boldsymbol{x}}_k, \boldsymbol{x}_k, \boldsymbol{y}_k, t_k) \tag{10}$$

$$\sigma_j = \sqrt{\frac{1}{n}\sum_{k=1}^{n} (t_j(\dot{\boldsymbol{x}}_k, \boldsymbol{x}_k, \boldsymbol{y}_k, t_k) - m_j)^2} \tag{11}$$

In Eqs. (10) and (11), n denotes the number of time steps in the simulation results.

The values m_j and σ_j for the jth term obtained by Eqs. (10) and (11) are then compared with the values of the other terms of the same equation. For this purpose, the relative values of the mean value and the standard deviation, $m_{r,j}$ and $\sigma_{r,j}$, respectively, are calculated:

$$m_{r,j} = \frac{m_j}{\frac{1}{N}\sum_{i=1}^{N}|m_i|} \tag{12}$$

$$\sigma_{r,j} = \frac{\sigma_j}{\frac{1}{N}\sum_{i=1}^{N}\sigma_i} \tag{13}$$

In Eqs. (12) and (13) N denotes the number of terms in the equation.

The jth term of the equation can be omitted if both conditions given by Eqs. (14) and (15) are fulfilled

$$|m_{r,j}| \ll 1 \tag{14}$$

$$\sigma_{r,j} \ll 1 \tag{15}$$

meaning that the term's mean value and deviation are small in comparison to the others. Otherwise, if only the condition in Eq. (15) is fulfilled, i.e., the term's value is nearly constant during the simulation, the term is substituted with its mean value.

However, the terms' values in the reference simulation results must have an approximately normal distribution so that the described ranking procedure is meaningful. This is difficult to ensure in practice. An example of an inappropriate model is a model with an integral response that was excited with a step signal.

Of course, there are many possibilities for further optimisation of the code implementation. For example, constant terms can easily be recognised by a structural analysis. In this case, it is obvious that the standard deviation is zero without any calculation of statistical properties.

More details can be found in Sodja (2012).

3.4. Implementation of model-reduction algorithm

The basic intention of our implementation of the *BModGen* (see Section 3) model reduction strategies is to serve as a supplementary reduction method capable of dealing with models represented as an equation system. Therefore, we implemented it in the OpenModelica framework (Open Source Modelica Consortium, 2012).

An existing implementation (Sommer et al., 2000) of the *BModGen* reduction strategies uses the functionality of general-purpose computer algebra software (Wolfram Research, Inc., 2012). In contrast, our implementation has to rely on the symbolic manipulation functionality provided by *OpenModelica*, which is specialised for the translation of the model's equations in a favourable simulation form.

Although reduction strategies employ some of the *OpenModelica* translation routines, the absence of more general symbolic manipulation functionality represents a serious limitation. It is thus more reasonable to export the model's equations into a general-purpose computer algebra environment. However, the focus of our implementation is to provide an intelligible presentation of underlying equation system of the model. Furthermore, it is rarely useful to investigate the complete equation system of a large complex model. Instead, only certain submodels are examined. Therefore, the information about the system decomposition must be available to the model reduction algorithm.

Rather than extending the export with additional information about the model structure, we found it to be more advantageous to integrate reduction algorithm with the *OpenModelica* translation tool so that all information about the model is easily accessible. In the future, the efficiency of the reduction algorithm could be improved by utilising an external symbolic mathematical software.

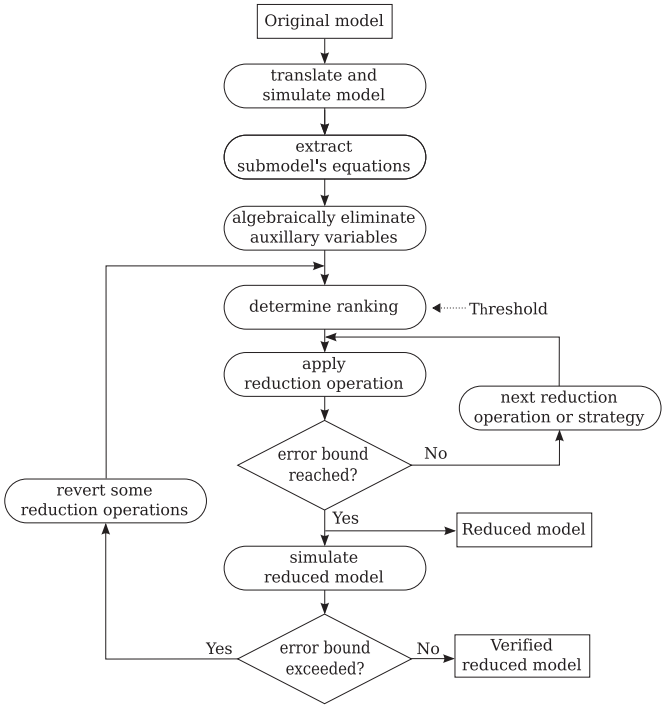

Fig. 1. Flow chart of equation-based model reduction.

In the current implementation, only two model reduction and simplification strategies are used: the algebraic elimination of auxiliary variables (i.e., variables that are not of particular interest) and the elimination of equation terms only on the base level (in the left and right-hand side expression of equation).

The algorithm is illustrated in Fig. 1, and the steps performed by the algorithm are the following:

1. A model is translated and simulated so that numerical reference values (e.g., trajectories) are obtained. An intermediate result of the translation, i.e., underlying equation system of the model, is saved for the later use in reduction procedure.
2. The user decides which submodel (i.e., component of the model) will be reduced. Equations belonging to this submodel must be identified and separated (i.e., marked) from the rest of the equations.
3. Algebraic elimination of the variables that were not specified as variables of interest by the user is carried out. If the user did not specify any variable of interest, a default choice is variables whose derivatives are present in the component's equation system.
   Unnecessary variables are eliminated in two steps. First, variables and equations that are not needed for the calculation of variables of actual interest are removed. This is achieved by executing Tarjan's algorithm (Tarjan, 1972) with appropriate inputs.
   In the translation of the model, Tarjan's algorithm is used for the computation of the *block lower triangular* (BLT) form of an incidence matrix of an equation system. The latter determines from which equation a variable will be solved. When the incidence matrix is converted to the BLT form, the non-zero square blocks are present on the main diagonal, while all blocks above the diagonal are zero blocks. The BLT matrix determines the order in which equations must be solved. However, several equations might form a strongly-connected components block, (i.e., algebraic loops); in such a case, the order of equations in such a block cannot be determined.

Tarjan's algorithm works on an incidence matrix represented as a graph, and it requires a set of starting nodes (i.e., variables that should be solved from the equation system) as an input. If only variables of interest are given as staring nodes, a minimal equation set for computing these variables will be determined with Tarjan's algorithm (Manzoni and Casella, 2011).

In the second step, algebraic substitution is performed. The algebraic substitution of trivial equations is also implemented for the translation process. The trivial equations are equations like $x = \pm y$, $x \pm y = 0$, $x = const.$, etc. However, the elimination of only trivial equations is not sufficient for subsequent reduction operations on equations' terms to be efficient. Therefore, we extended it to handle arbitrary equations consisting of two terms, which is still not the final solution of the problem.

Algebraic equations must reduce the number of simple equations (consisting of only few equation terms) but it should also not create overly complex equations that might affect performance of nonlinear equation solver and are incomprehensible to the user.

However, the intention of the proposed implementation is to serve as a proof of concept and not a development of an advanced algebra system. Therefore, this task remains for a future work.

4. Elementary reduction operations are ranked according to the selected ranking metric. In our implementation, only the basic term removal or replacement with a non-zero constant is supported, and two ranking methods are implemented: ranking by one-step solver and by statistical properties. If there are mutually exclusive reduction operations (e.g., the removal of a term or replacement with a non-zero constant), the priority order must be established (e.g., term elimination before substitution with a constant).
5. After the ranking is known, the highest-ranked reduction operations are applied until their cumulative predicted error does not reach the prescribed error bound.

   To prevent the possibility that the reduced equation system will be structurally singular (i.e., the Jacobian matrix of the system will be singular), the equations are guarded. Before each reduction operation is applied to the system, it is checked to verify that there will still be at least one reference of the variable matched to the equation that is to be modified[1]. However, this simple guard does not protect the consistency of the initialisation problem or the stability of the model.
6. The correctness of the reduced model is verified by a simulation. If the results do not meet the prescribed error criteria, the algorithm exits with a warning. The user must then take the proper action (e.g., lowering the error threshold, choosing another ranking metric, etc.). An attempt could also be made to recover the algorithm automatically by reverting some reduction operations and changing the strategy, but there are no suitable algorithms that would also be computationally efficient present in the literature (Wichmann, 2004).

   Thus, a reduced model is one output of the algorithm, and another is a verified reduced model, which is obtained after the reduction is verified with a simulation. However, the user might decide against a costly verification by using a simulation. Specifically, the simplified and reduced equation system obtained as a reduced model can provide sufficient insight, even if it is potentially wrong, i.e., it violates the specified error tolerances.
7. Before the results of the reduction are presented to the user, the equations of the reduced model are simplified.

[1] A one-step solver ranking calculates the inverse Jacobian matrix for each reduction operation and, therefore, implicitly determines whether the reduction operation makes the equation system structurally singular (when the inverse Jacobian matrix is singular). However, it does not determine singularity for a combination of reduction operations.

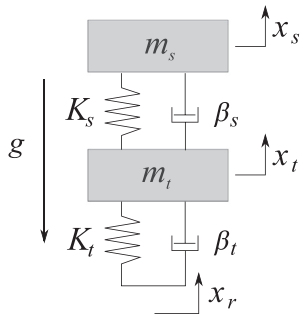

Fig. 2. Idealised physical model of car suspension.

Our user interface of the equation-based reduction algorithm consists of a single command `reduceDAE()`. Before the command is invoked, the model whose submodel(s)' shall be reduced must be simulated. A name of the component (submodel) is then passed as a first argument to the `reduceDAE()` or if the complete model shall be reduced the first argument is omitted. Other commands' arguments set the following options: the time interval on which reference simulation results will be considered, the selection of ranking metric, the error tolerance, and the option that the reduced equation system is verified by simulation.

More details can be found in Sodja (2012).

3.5. Example

The performance of the implemented equation-system reduction algorithm is demonstrated on the simple model of a car suspension system illustrated in Fig. 2.

There are two different displacements: x_s is the displacement of the suspension system (which is equal to the body of the car) and x_t is the displacement of the tyre. The mass of the car m_s also includes the mass of the suspension system but does not include the mass of the tyres and wheels, which are collected separately in the mass m_t. The car hits the smooth curb, which is modelled with the function (see the input signal x_r in Fig. 3)

$$x_r(t) = \begin{cases} 0.4(t-1) - \frac{0.1}{\pi} sin(4\pi(t-1)) & ; \quad 1 \le t \le 1.5 \\ 0 & ; \quad t < 1 \\ 0.2 & ; \quad t > 1.5 \end{cases} \tag{16}$$

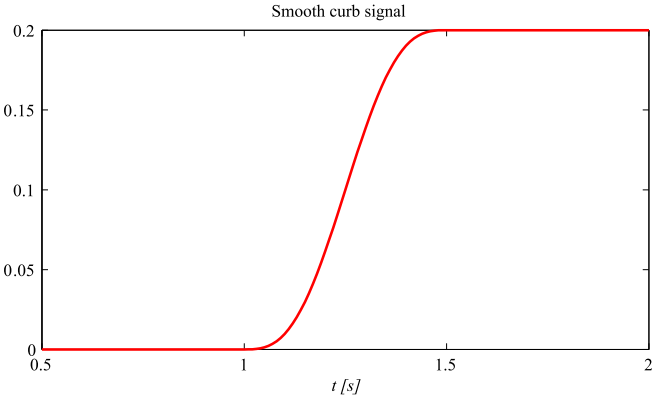

Fig. 3. Smooth curb signal.

```
model Example
  constant Real g = 9.81;
  parameter Real m_t = 36.6;
  parameter Real m_s = 267;
  parameter Real beta_t = 200;
  parameter Real beta_s = 700;

  parameter Real k1_t = 193.915;
  parameter Real k3_t = 2.0e8;
  parameter Real k1_s = 18.742;
  parameter Real k3_s = 2.0e5;
  Real v_t(start=0,fixed=true);
  Real v_s(start=0,fixed=true);
  Real x_t ,x_s;
  Real x_r(start=0,fixed=true);
  Real Fk_t, Fk_s;
initial equation
  der(v_t) = 0;
  der(v_s) = 0;
equation
  der(x_s) = v_s;
  m_s * der(v_s) = -beta_s * (v_s-v_t) - Fk_s - m_s * g;
  der(x_t) = v_t;
  m_t * der(v_t) = - beta_t * (v_t-der(x_r)) - beta_s *
     (v_t-v_s) - Fk_t + Fk_s - m_t * g;
  Fk_t = k1_t * (x_t-x_r)  + k3_t * (x_t-x_r)^3 ;
  Fk_s = k1_s * (x_s-x_t)  + k3_s * (x_s-x_t)^3 ;
  der(x_r) = if noEvent(time < 1.0) then 0.0 else  if
     noEvent(time > 1.5) then 0.0 else 0.4 * (1.0 -
     cos(12.5663706143592 * (time - 1.0)));
end Example;
```

Listing 1. Implementation of the example model.

Table 1
Description of notations in the car suspension system.

x_s, v_s	Displacement, velocity of the suspension system
x_t, v_t	Displacement, velocity of the tyre
m_s	Mass of the the car and suspension system
m_t	Mass of the tyre and wheel
β_s	Damping coefficient of the suspension system
β_t	Damping coefficient of the tyre
F_{Ks}	Nonlinear force of the suspension spring
F_{Kt}	Nonlinear force of the tyre spring
K_s, K_{1s}, K_{3s}	Spring coefficients of the suspension system
K_t, K_{1t}, K_{3t}	Spring coefficients of the tyre
g	Gravitational acceleration

Using Newton's laws for both different displacements, the following equations appear:

$$\begin{aligned} \dot{x}_s &= v_s \\ m_s\dot{v}_s &= -\beta_s(v_s - v_t) - F_{Ks} - m_s g \\ \dot{x}_t &= v_t \\ m_t\dot{v}_t &= -\beta_t(v_t - \dot{x}_r) - \beta_s(v_t - v_s) - F_{Kt} + F_{Ks} - m_t g \\ F_{Ks} &= K_{1s}(x_s - x_t) + K_{3s}(x_s - x_t)^3 \\ F_{Kt} &= K_{1t}(x_t - x_r) + K_{3t}(x_t - x_r)^3 \end{aligned} \tag{17}$$

The notations of parameters and variables used in Fig. 2 and in Eqs. 17 are described in Table 1. We can also observe that the system is nonlinear, as both spring forces F_{Ks} and F_{Kt} are modelled with nonlinear equations.

A set of equations that describe the system's dynamic behaviour, derived from an idealised-physical model written according to the Modelica syntax, is listed in Listing 1.

The first step of the reduction algorithm is an algebraic elimination of the variables not indicated as being of interest. In the example model, there are no variables that could be eliminated by the current implementation of an algebraic-substitution algorithm, and the equation system thus remains intact.

3.5.1. Ranking by one-step solver

After the equation system is algebraically simplified, other reduction strategies are applied, i.e., the elimination of insignificant equation terms. The results of the ranking obtained by the one-step solver method is shown in Table 2 with the column 'Error predicted'. The reduction steps were also verified with simulation (see column 'Error actual' in Table 2).

The terms are arranged with increasing importance (error). As it is evident from the last two columns of Table 2, the error predictions are reasonably accurate except for the 6th ranked term.

Fig. 4 shows the responses of the original model and the reduced models: the one with the five highest ranked reduction operations applied and the other with the six highest ranked operations. Displacements of the tyre x_t and the suspension system x_s are presented. While it is difficult to discern the responses of the original and reduced model with the five highest ranked terms removed, the discrepancy of the model with six terms removed is clearly visible.

The 1st and 3rd highest ranked terms are part of the springs' constitutive laws that consist of linear and nonlinear terms. At a given operating point, the springs are compressed considerably so

Table 2
Ranking obtained with a one-step solver for the equation system in Listing 1.

Rank	Term to eliminate	Error [%]	
		predicted	actual
1^{st}	"`k1_t * (x_t - x_r)`" in equation at line 25	$4.6 \cdot 10^{-2}$	$6.4 \cdot 10^{-2}$
2^{nd}	"`-beta_t * (v_t - v_r)`" in equation at line 24	0.13	0.12
3^{th}	"`k1_s * (x_s - x_t)`" in equation at line 26	0.14	0.17
4^{th}	"`m_t * der(v_t)`" in equation at line 24	1.81	1.03
5^{th}	"`-m_t * g`" in equation at line 24	5.2	2.12
6^{th}	"`-beta_s * (v_s - v_t)`" in equation at line 22	11.6	108.8
7^{th}	"`k3_t * (x_t-x_r)^3`" in equation at line 25	13.2	11.032
...	...	...	...

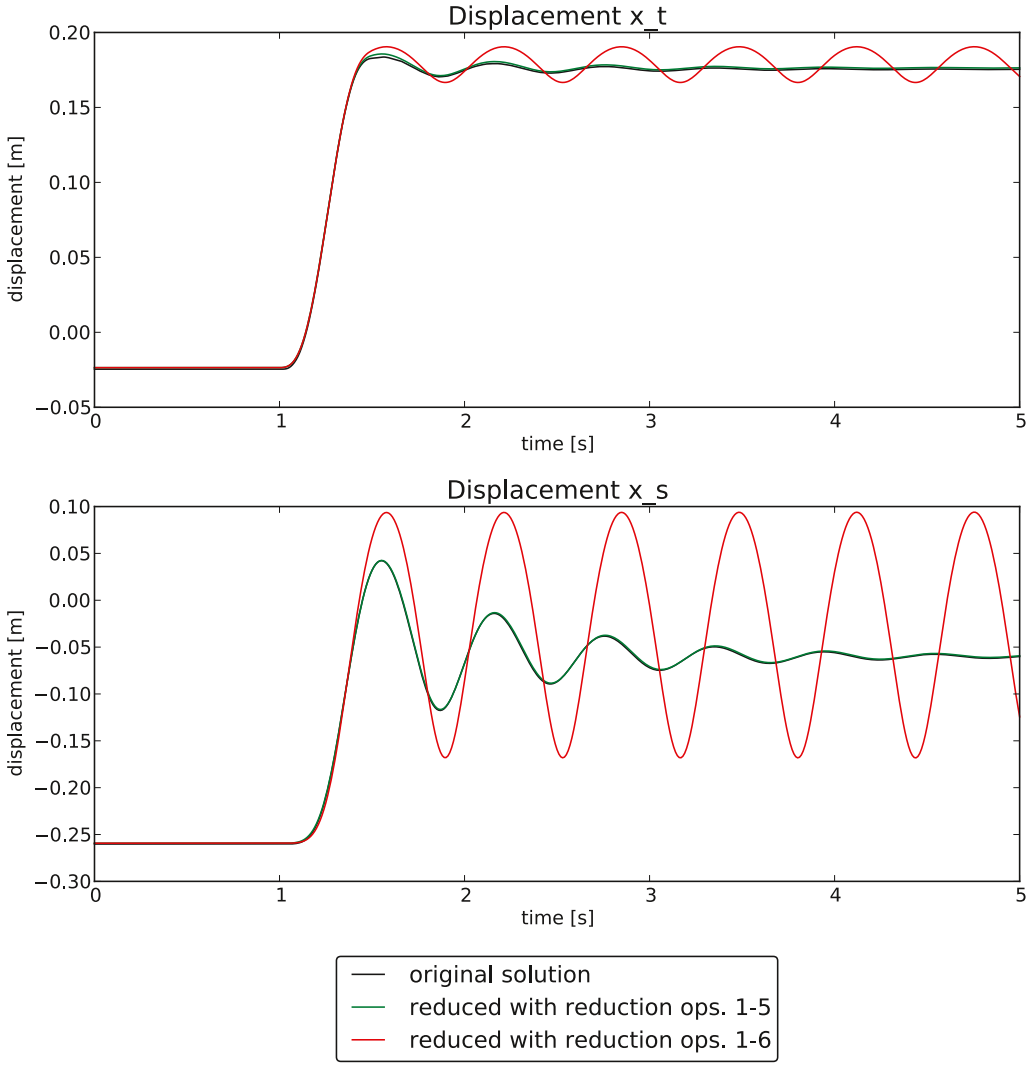

Fig. 4. Effect of reduction operations applied. Black and green curves are almost the same. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

that the nonlinear term is prevailing; therefore, neglecting the linear term is justified (as also proved by simulation).

Elimination of the terms ranked as 2nd, 4th, and 5th corresponds to neglecting of the damping of the tire, the inertia of the tire's mass, and the effect of the gravity on the tire respectively. For the 6th ranked term, it is predicted that it will introduce only a moderate error to the model's behaviour but in reality the response of the model is changed substantially because damping is almost completely eliminated from the model. This is demonstrated in Fig. 4, in which the discrepancy of the model with sex terms removed is clearly visible.

Although the order of the reduction steps is correct (at least for the first five rows), it was shown that the ranking method can fail to predict correctly the impact of particular operations. As mentioned, slight perturbations of the ranking methods parameters (e.g., selection the parameter h (step size), $\bar{h} = a_0/h,$) can significantly influence the accuracy of the ranking procedure.

3.5.2. Ranking based on statistical properties

An alternative ranking of the terms according to their mean value and standard deviation is shown in Table 3.

Table 3
Ranking based on the statistical properties of the terms for the equation system in Listing 1. The last two columns, absolute relative mean value $|m_r|$ and relative standard deviation σ_r, are determined using Eq. (12) and (13). The lines of the table are sorted according to the last column.

Rank	Equation term	$\lvert m_r \rvert$	σ_r
1^{st}	"`-m_t * g`" in equation at line 24	0.06	0.0
2^{nd}	"`-m_s * g`" in equation at line 22	0.50	0.0
3^{th}	"`k1_t * (x_t - x_r)`" in equation at line 25	$7.9 \cdot 10^{-4}$	$2.9 \cdot 10^{-4}$
4^{th}	"`k1_s * (x_s - x_t)`" in equation at line 26	$8.3 \cdot 10^{-4}$	$3.2 \cdot 10^{-4}$
5^{th}	"`-beta_t * (v_t - v_r)`" in equation at line 24	$2.2 \cdot 10^{-7}$	$2.8 \cdot 10^{-3}$
6^{th}	"`m_t * der(v_t)`" in equation at line 24	$1.7 \cdot 10^{-6}$	0.03
7^{th}	"`-beta_s * (v_s - v_t)`" in equation at line 22	$1.0 \cdot 10^{-5}$	0.10
8^{th}	"`Fk_s`" in equation at line 22	0.5	0.44
9^{th}	"`m_s * der(v_s)`" in equation at line 22	$1.4 \cdot 10^{-4}$	0.46
10^{th}	"`Fk_s`" in equation at line 24	0.44	0.48
...	...	...	...

If a threshold for a negligible term is selected to be $|m_r| \ll 0.1$ and $\sigma_r \ll 0.1$, the same equation terms (1^{st}, 3^{rd}, 4^{th}, 5^{th}, and 6^{th}) fall under the threshold as the five highest ranked in the previous ranking using the one-step solver method (see Table 2).

The ranking shown in Table 3 was computationally much less demanding than the ranking shown in Table 2 and yielded almost identical results. However, the ranking with the one-step solver is much easier to extend the other types of reduction eliminations. Furthermore, the ranking can be done without simulation results as the procedure can predict the appropriate errors.

3.5.3. Initialisation problem

The presented model reduction algorithm has some limitations, which prevent running the *verified reduced model* automatically (see Fig. 1). The implemented algorithm is not capable of assuring that it will be possible to initialise the reduced model. The initialisation problem of the model given with Listing 1 becomes invalid when the equation term "`m_t * der(v_t)`" at line 24 is removed because `der(v_t)` is no longer a variable of the system, and hence the equation at line 18 is invalid. To be able to simulate the reduced model, the equation at line 18 has to be removed manually. The possible solution in this case could be that, with the use of appropriate structural analysis, the initial conditions that are matched with derivatives, which have been completely eliminated from the model, could also be eliminated automatically.

4. Reduction of object diagrams on higher hierarchical levels

The approach was elaborated in detail in Sodja et al. (2018), in which more examples are also available. Here, we added the content only for completeness and for brief comparison. More details can also be found in Sodja (2012).

The object-oriented modelling approach provides various means to structure and organise large models. The most important is that structural information about the modelled system can be encoded as a set of submodels connected with each other, whereby the connection also defines the relation, i.e., the interaction between the components. The resulting connection can be represented graphically; in Modelica, it is often referred to as an object diagram, and the corresponding structural equations are then generated automatically.

Therefore, the model-reduction approaches presented in the previous section could be used to reduce the object diagram by reducing the corresponding equation system and then mapping the results of the reduction back to the object diagram so that the model's realisation is preserved (Sommer et al., 2008).

Nevertheless, the object diagram contains more information about the physics of the modelled system; the user might thus find the ranking metrics with a clear physical interpretation more convenient.

4.1. Ranking metrics

As stated in Section 2.1, there are various ranking metrics with a physical meaning, but most of them are difficult to evaluate when dealing with models composed of complex components.

Although different domains are modeled with rather different schemes and connections, acausal connections for modelling physical interactions are of special interest. Each (dynamic) interaction between physical systems results in an energy exchange between the systems. Thus, it is highly intuitive to chose the energy-based metrics for the reduction of the physical-system models.

The majority of mathematical models are derived from first principles, e.g., the conservation of energy. However, the energy or power is usually not explicitly available in the model, i.e., it is not a variable of the model. Therefore, a procedure for energy or energy-flow calculations must be explicitly provided for the model (e.g., as a Lyapunov function (Chang et al., 2001)) or the model must be formulated accordingly, so that the energy can be unambiguously calculated for each component.

Energy-based metrics were systematically developed for bond graphs, a graphical object-oriented modelling formalism, in which the energy flow associated with each component (i.e., the elementary submodel) is simple to determine (Borutsky, 2010). Each component is connected with a bond (acausal link) to the rest of the model, and the bond consists of a pair of variables (effort variable and flow variable) whose product always equals the power (i.e., the energy flow).

The simplest ranking metrics require only the net energy flow through a component's border (or a connection) to be known (Rosenberg and Zhou, 1988; Louca, 1998), while more sophisticated metrics also take some structural information into account, e.g., the net energy flow of neighbouring components (Ersal, 2007), the relation between the net energy exchanged, and the state variables of the component (Ye and Youcef-Youmi, 1999), etc.

4.2. Determination of the energy flows in object diagrams

Modelica object diagrams, when modelling physical systems, share many similarities with bond graphs, which can be efficiently used for object-oriented acausal modelling. A Modelica library for the modelling with bond graphs exists as the implementation was rather simple and natural. Therefore, it is possible to adapt most of the bond-graph simplification techniques to Modelica's object diagrams. Of course, the energy concept in bond graphs is much more unified in comparison with different Modelica libraries.

In Modelica, connectors usually contain a pair of the effort and the flow variables, but unfortunately their product is not necessarily an energy flow as in bond-graph formalisms. For example, besides standardising the Modelica language, the Modelica Association also provides the Modelica Standard Library (Modelica Association, 2010), which contains many elementary models (and corresponding connector definitions) from almost all physical domains. For example, connections describing one-dimensional heat conduction consist of temperature and heat flow rather than temperature and entropy flow in the bonds of the bond graphs (Cellier and Greifeneder, 2009).

Nevertheless, our analysis of the connectors defined and used in the Modelica Standard Library (details for analog electrical circuits, planar mechanics, multi-body mechanics and transfer of heat and mass can be found in Sodja (2012), Sodja et al. (2018)) showed that energy exchanged in an interaction modelled by a connection can, in most cases, be determined from the variables of the connector as long as the connections describing a different physical interaction cannot be connected together (e.g., connector that permits 'mixing' of different fluids; in the Standard Library, a parameter (package) inside the connector is used to prevent such cases).

Additionally, the net energy flow of a component in Modelica's models is not as explicitly available as it is in bond graphs, where it is determined by a bond connecting the component with the rest of the bond graph. It can still be calculated from the energy flows of the component's connections with the rest of the object diagram:

$$\dot{E}_i(t) = -\sum_{k=1}^{n} \dot{E}_{i,k}(t) \tag{18}$$

In Eq. (18), $\dot{E}_i(t)$ denotes the net energy flow of the component, and $\dot{E}_{i,k}(t)$ is the energy flow between the ith and the kth component.

4.3. Implementation of model reduction on higher hierarchical levels

The model-reduction procedure performed on an object diagram is similar to the one for the reduction of the equation systems shown in Fig. 1:

1. the model is simulated;
2. the component-ranking metric is calculated;
3. all the components whose rankings fall below a certain threshold are either removed or replaced with simpler instances.

In order to evaluate an energy-based metric, the net energy flows of the components (or their connections) must be calculated first (because they are not usually variables of the model). Although this is possible to do from the simulation results, it is not always straightforward; for example, it might require numerical differentiation.

We implemented a prototype of component ranking in the OpenModelica framework (Open Source Modelica Consortium, 2012). The procedure is done in two parts, as illustrated in Fig. 5.

In the first part, an additional step, namely instrumentation, is inserted into translation. After model sources are parsed and abstract syntax tree (AST) is generated, the AST is traversed and for every encountered connection statement of appropriate type a variable and an equation for calculation of the energy flow is added to the model. Furthermore, a graph of connections is built simultaneously, so that relevant components and their energy-flows can be extracted in the post-processing part of ranking.

In the second part, ranking weights are calculated for all components listed in the connection graph from the simulation results and then components are ranked.

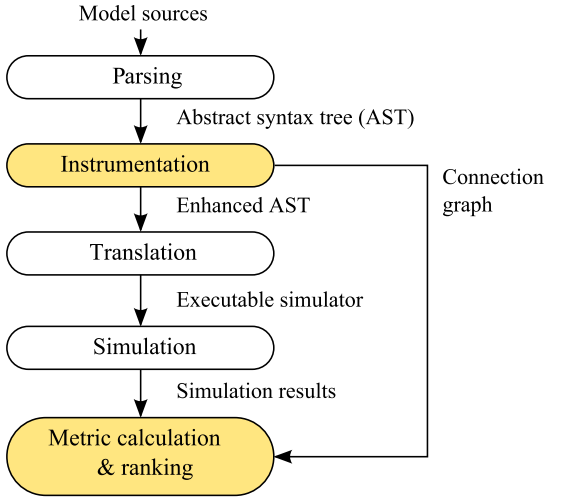

Fig. 5. Implementation of component ranking using energy-based metrics.

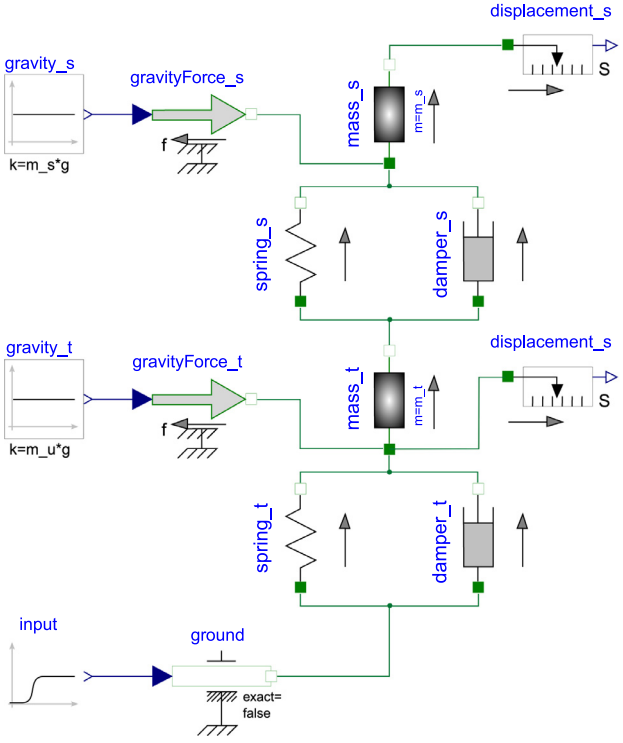

Fig. 6. Car suspension system: model represented with a Modelica object diagram.

Only a command line interface is provided in the prototype implementation. Instrumentation was implemented as a modified function for translation, called `instrumentModel`, which translates and instruments a model. The connection graph is saved in the environment. After translation (and instrumentation), the model is simulated. Ranking of the model's components is calculated and printed by the functions `rankComponents` and `rankConnections`. It is possible to set a time window and choose among three different ranking metrics: RMS power, dubbed activity metric, and importance vector. A simple ranking metric (dubbed *activity metric*) defined by Louca (1998)

$$A_i = \int_{t_1}^{t_2} |\dot{E}_i(t)| \cdot dt \tag{19}$$

is applicable for a wide range of physical models. $\dot{E}_i(t)$ is described with Eq. (18). The activity A_i of the i^{th} component is the integral of

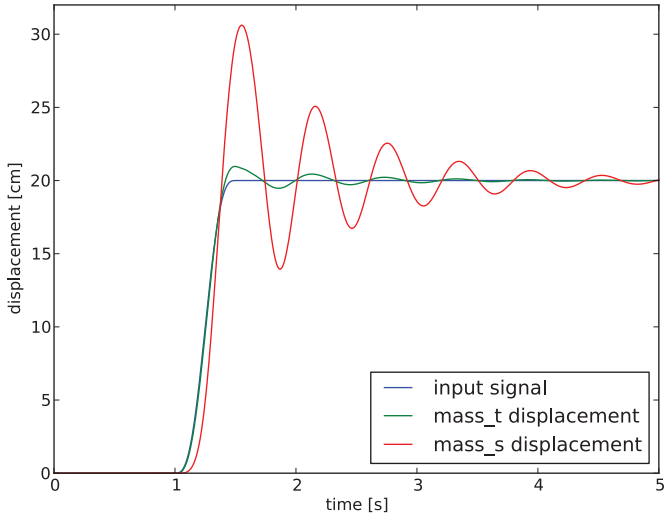

Fig. 7. A car hits a smooth curb: a low-frequency excitation signal is given as an input to the model in Fig. 6. Other responses (displacement of the tyre (*mass_t*) and suspension system (*mass_s*)) are depicted. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Ranking of components when the model from Fig. 6 is fed by the input shown in Fig. 7.

Element	Activity [J]	Relative [%]	Accumulated [%]
gravityForce_s	2,270.06	37.06	37.06
spring_s	1,763.33	28.79	65.85
ground	795.02	12.98	78.82
mass_s	787.65	12.86	91.68
damper_s	198.82	3.25	94.93
spring_t	192.57	3.14	98.07
gravityForce_t	92.98	1.52	99.59
mass_t	24.53	0.40	99.99
damper_t	0.53	0.01	100.00

the absolute net energy flow that the element exchanges with the environment (through connectors in Modelica) in the time interval $[t_1, t_2]$.

When the model consists of several hierarchical levels, whereby each is presented with an object diagram, each hierarchical level is considered separately.

After the ranking of the elements is available, the model can be reduced by removing all the elements that fall below a certain threshold (value of *activity* in our case). However, our current implementation provides only the results of ranking in a printed form (a table). The ranking table can then be used to simplify the model manually. Automatic reduction is a matter for future investigations.

4.4. Example

A model of the system depicted in Fig. 2 can also be built with a Modelica object diagram using components from the Standard Modelica Library (Mechanics/Translation). An exception is the components of the nonlinear springs, which had to be custom-made. The resulting object diagram is shown in Fig. 6.

The model was simulated using the same input (excitation) signal as in the previous example, and the input signal together with the model response are shown in Fig. 7. The components of the model were ranked using the described procedure, and the activity metric and the results are shown in Table 4. The second column of Table 4 consists of the activities of all components calculated with Eq. 19. The third column contains the relative activities of the components (the total equals 100%), and the last column shows the accumulated relative activities. This column very clearly shows how

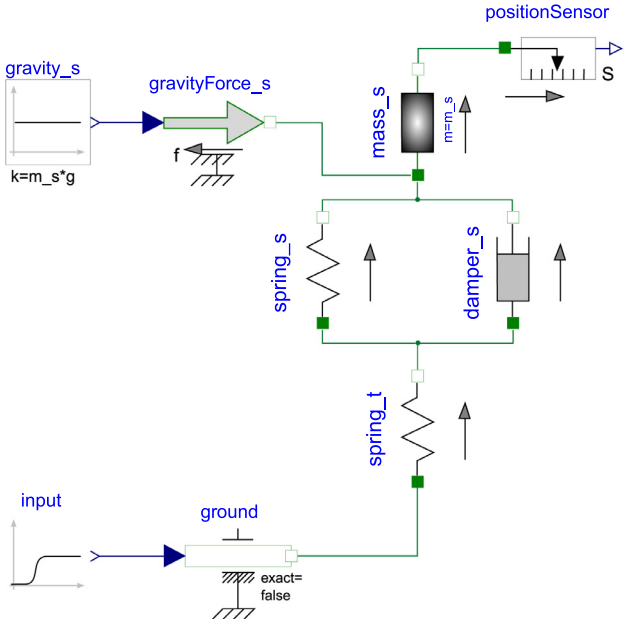

Fig. 8. The reduced model.

many components have to be taken into account to obtain reasonable accuracy.

For example, if the three bottom-ranked components, the `damper_t`, the `mass_t`, and the `gravityForce_t`, together amounting to less than 2% of the total systems' activity, are removed, the reduced model shown in Fig. 8 is obtained. The response of the reduced model on the same input signal differs only negligibly from the full-model response. As can be verified by simulation, the maximum relative error is less than 0.1%. Similar reduction operations were also suggested with ranking with the one-step solver in the previous example, in which the `damper_t`, the `mass_t` and the `gravityForce_t` are covered with the 2nd, 4th, and 5th ranked terms (see Table 2).

However, if we also proceed to remove the fourth bottom-ranked component, the nonlinear `spring_t`, the responses of the full and reduced model change more significantly that its activity would suggest. The main reason is the change of the operating point. In this case, the component should be replaced with a correspondingly simplified component instead of being removed completely. Our algorithm cannot currently handle such cases and, therefore, this part is left entirely to the user.

5. Conclusions

Contemporary modelling techniques and tools allow very detailed and complex models to be built easily from already-prepared components. However, the models of physical systems must match reality, and currently only the modeller can take care of this requirement by a thorough verification and validation of the model. Hence, handling of the models' complexity will be an important topic in the further development of modelling techniques and approaches.

In this paper, we focused on two procedures implemented in Open Modelica for the reduction of equation systems and object diagrams. Due to the realisation of the preservation concept, the described approaches are appropriate for efficient model verification.

The aim of our approach is not an improvement over the current state-of-the-art model-reduction methods, but an extension and adoption of the existing methods to support highly hetero-

geneous and multi-domain models, which can be built in Modelica by imposing as few as possible restrictions on the modelling formalism or possible physical domains that can be used. Without doubt, the reduction method applicable to very heterogeneous models cannot take into account all the reduction possibilities, but when dealing with large complex models, every piece of information regarding simplification can be useful.

The described realisation-preserving model reduction for generic Modelica models is a definitely a very difficult task. From this point of view, even partial results, such as those described in this paper, may eventually also pave the way for the introduction of such techniques for more complex real industrial problems. Of course, it is questionable if the described procedures can be fully automated. We believe that an important feature will be the interaction with the end-user who can properly uses many useful insights and indications from such tools and guide the tool in the process of model reduction in an interactive way, e.g., selecting the parts of a complex model that should be subject to model reduction, selecting the proper techniques and, in general, bringing in some expert judgement that an automated tool can hardly possess. We believe that the described model reduction in Modelica and similar tools will never be a 'one push-button activity'. It will probably work on smaller parts in semi-automatic way with appropriate manual operations.

Although our model reduction (or only the component-ranking procedure) can be applied to almost any model implemented in Modelica, it has also other limitations, the most notable being an inability to handle the initial problem of the model adequately. Modelica permits specifying the initial conditions in various ways (e.g., with the variable's initial value, which can be a fixed or only a guess value, by initial equations, etc.) and it is very complicated to map the original initial problem to the reduced model properly. Specifically, the reduction procedures do not lead to 'well-posed' systems of nonlinear equations. Therefore, the reduction process should also take into account this problem. One idea is to rank the (eliminated) terms in a way that initial conditions of original and reduced system are similar. However, this issue was not in the focus of our work.

Funding

This work has been supported by Slovenian Research Agency with the Research Programme P2-0219.

ORCID iD

Borut Zupančič https://orcid.org/0000-0002-6431-3861

Appendix. Vocabulary of expressions

Model validation: It involves the checking of the overall modelling procedure proofing that the model is an adequate representation of the reality.

Model simplification: It indicates that modifications to a model perform more concise representation but the behaviour of the original and simplified models are exactly the same (e.g., the algebraic elimination of an algebraic loop or the simplification of a block scheme of a control system using the algebra of block schemes).

Model reduction: Some less interesting aspects of the behavior of the original model are omitted. Hence, the reduction only represents a portion of the behavior of the original model (e.g., parts of the model can be exchanged with simpler ones - a lower order or a linearised submodel). Therefore, the resulting reduced model is thus valid only within a limited range.

Bond graph: Graphical representation of systems, which could show simultaneously the topological and computational structure, and which would be general, i.e., applied to all kinds of systems. Basically they clearly represent how energy is moving through a component and, therefore, they can efficiently be used in model reduction techniques.

DAE - differential algebraic equations: A general form of model description in which a nonlinear function $\boldsymbol{F}$ of derivatives $\dot{\boldsymbol{x}}$, states $\boldsymbol{x}$, algebraic variables $\boldsymbol{y}$ and independent variable t equals zero

$$\boldsymbol{F}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{y}, t) = 0 \tag{20}$$

They are distinct from ordinary differential equation (ODE) in that a DAE is not completely solvable for the derivatives of all components of the state vector $\boldsymbol{x}$. Numerical simulation of continuous systems, which is based on numerical integration, is much easier, faster and more efficient if it is possible to convert DAE form into ODE form.

Example of a system described with DAE:

$$\begin{aligned} \dot{x} + sin(y) &= sin(t) \\ y - x &= e^{-0.9y} cos(x) \\ x(0) &= 1 \end{aligned}$$

References

Borchers, C., 1997. Automatische Generierung von Verhaltensmodellen für nichtlineare Analogschaltungen. Ph.D. thesis. Institut für Mikroelektronische Systeme, Universität Hannover.

Borutsky, W., 2010. Bond Graph Methodology, Development and Analysis of Multidisciplinary Dynamic System Models. Springer, Germany.

Bunus, P., 2004. Debugging Techniques for Equation-Based Languages. Ph.D. thesis. Linköping University.

Cellier, F.E., 1991. Continous System Modeling. Springer Verlag, New York.

Cellier, F.E., Greifeneder, J., 2009. Modeling chemical reactions in Modelica by use of chemo-bonds. In: Proceedings of the Seventh Modelica Conference. Como, Italy, pp. 142–150.

Chang, S.Y., Carlson, C.R.C., Gerdes, J.C., 2001. A Lyapunov function approach to energy based model reduction. In: Proceedings of the ASME Dynamic Systems and Control Division – 2001 IMECE. New York, USA, pp. 363–370.

Ersal, T., 2007. Realization-Preserving Simplification and Reduction of Dynamic System Models at the Graph Level. Ph.D. thesis. University of Michigan.

Ersal, T., Fathy, H.K., Rideout, D.G., Louca, L.S., Stein, J.L., 2008. A review of proper modeling techniques. J. Dyn. Syst. Meas. Control 130 (6), 061008.

Fritzson, P., 2014. Principles of Object Oriented Modeling and Simulation with Modelica 3.3. Wiley-IEEE Press, USA.

Kokotovic, P., Sannuti, P., 1968. Singular perturbation method for reducing model order in optimal control design. IEEE Trans. Autom. Control 13, 377–384.

Louca, L.S., 1998. An Energy-Based Model Reduction Methodology for Automated Modeling. Ph.D. thesis. University of Michigan.

Manzoni, V., Casella, F., 2011. Minimal equation sets for output computation in object-oriented models. In: Proceedings of the Eighth Modelica Conference. Dresden, Germany, pp. 784–790.

Matko, D., Karba, R., Zupančič, B., 1992. Simulation and Modelling of Continuous Systems: a Case-Study Approach. Prentice-Hall International Series in Systems and Control Engineering. Prentice Hall.

Modelica Association, 2010. Modelica Standard library 3.1, User's Guide. Modelica Association. Available from: http://www.modelica.org/libraries [Accessed 29th October, 2018].

Modelica Association, 2012. Modelica Specification, Version 3.3. Available from: http://www.modelica.org/documents/ModelicaSpec33.pdf [Accessed 29th October 2018].

Murray-Smith, J.D., 2009. Simulation model quality issues in product engineering: a review. Simul. News Eur. 19 (2), 47–57.

Open Source Modelica Consortium, 2012. OpenModelica. Available from: http://www.openmodelica.org [Accessed 29th October 2018].

Rosenberg, R., Zhou, T., 1988. Power-based model insight. In: Proceedings of the ASME WAM Symposium on Automated Modeling for Design. New York, USA, pp. 61–67.

Sheehan, B.N., 1999. TICER: realizable reduction of extracted RC circuits. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, pp. 200–203.

Sodja, A., 2012. Object-Oriented Modelling and Simulation Analysis of the Automatically Translated Models. Ph.D. thesis. University of Ljubljana, Faculty of Electrical Engineering. Available from http://msc.fe.uni-lj.si/Download//Zupancic/Dissertation_Anton_Sodja.zip [Accessed 29th of October 2018].

Sodja, A., Škrjanc, I., Zupančič, B., 2018. Realization-preserving model reduction of object oriented models using energy-based metrics. Simul. Trans. Soc. Model. Simul. Int. 95 (7), 607–620.

Sodja, A., Zupančič, B., 2012. Realisation-preserving model reduction of models in Modelica. In: Proceedings of the Mathmod 2012, the Seventh Vienna Conference on Mathematical Modelling. Vienna, Austria, pp. 322–328.

Sommer, R., Halfmann, T., Broz, J., 2008. Automated behavioral modeling and analytical model-order reduction by application of symbolic circuit analysis for multi-physical systems. Simul. Model. Pract. Theory 16, 1024–1039.
Sommer, R., Hennig, E., Thole, M., Halfmann, T., Wichmann, T., 2000. Analog Insydes 2 – new features and applications in circuit design. In: Proceedings of the Sixth International Workshop on Symbolic Methods and Applications in Circuit Design. Lisbon, Portugal.
Tarjan, R.E., 1972. Depth first search and linear graph algorithms. SIAM J. Comput. 1, 146–160.
Wichmann, T., 2003. Transient ranking methods for the simplfication of nonlinear DAE systems in analog circuit design. Proc. Appl. Math. Mech. 2, 448–449.
Wichmann, T., 2004. Symbolische Reduktionsverfahren Für nichtlineare DAE-Systeme. Ph.D. thesis. Technische Universität Kaiserslautern, Fachbereich Mathematik.
Wolfram Research, Inc., 2012. Mathematica. Available from: http://www.wolfram.com/mathematica [Accessed 29th October 2018].
Ye, Y., 2002. Model Reduction in Physical Domain. Ph.D. thesis. Massachusttes Institute of Technology.
Ye, Y., Youcef-Youmi, K., 1999. Model reduction in the physical domain. In: Proceedings of the American Control Conference. San Diego, CA, USA, pp. 4486–4490.
Zupančič, B., Sodja, A., 2013. Computer-aided physical multi-domain modelling: Some experiences from education and industrial applications. Simul. Model. Pract. Theory 33, 45–67.

Anton Sodja received a Ph.D. from the University of Ljubljana, Faculty of Electrical Engineering. Currently, he is working as a research and development engineer at ABB Gomtec GmbH, Germany.

Igor Škrjanc is a full professor at the University of Ljubljana, Faculty of Electrical Engineering, Slovenia. He is Humboldt and JSPS research fellow.

Borut Zupančič is a full professor at the University of Ljubljana, Faculty of Electrical Engineering, Slovenia. He was also a guest professor at the Technical University Vienna and the president of EUROSIM - the Federation of European Simulation Societies.