Page Proof Instructions and Queries

Journal Title: SIM
Article Number: 815338

Thank you for choosing to publish with us. This is your final opportunity to ensure your article will be accurate at publication. Please review your proof carefully and respond to the queries using the circled tools in the image below, which are available **by clicking "Comment"** from the right-side menu in Adobe Reader DC.*

Please use *only* the tools circled in the image, as edits via other tools/methods can be lost during file conversion. For comments, questions, or formatting requests, please use . Please do *not* use comment bubbles/sticky notes .

*If you do not see these tools, please ensure you have opened this file with **Adobe Reader DC**, available for free at get.adobe.com/reader or by going to Help >Check for Updates within other versions of Reader. For more detailed instructions, please see us.sagepub.com/ReaderXProofs.

Sl. No.	Query
	Please note, only orcid ids validated prior to acceptance will be authorized for publication; we are unable to add or amend orcid ids at this stage.
	Please confirm that all author information, including names, affiliations, sequence, and contact details, is correct.
	Please review the entire document for typographical errors, mathematical errors, and any other necessary corrections; check headings, tables, and figures.
	Please ensure that you have obtained and enclosed all necessary permissions for the reproduction of art works (e.g. illustrations, photographs, charts, maps, other visual material, etc.) not owned by yourself. please refer to your publishing agreement for further information.
	Please note that this proof represents your final opportunity to review your article prior to publication, so please do send all of your changes now.
	Please confirm that the funding statements are accurate.
1	Please check that the Table 1 caption is appropriate.
2	The tables have been renumbered to accommodate the first, unnumbered, table. Please check that all citations in the text are as intended.
3	Reference 5: Please provide the editor name(s), the location of the conference, the full date of the conference, the paper number, city of publication and the name of the publisher.
4	Reference 8: Please provide the city of publication.
5	Reference 9: Please provide the editor name(s), the full date of the conference, the paper number, city of publication and the name of the publisher.
6	Reference 10: Please provide the editor name(s), the full date of the congress, the paper number, city of publication and the name of the publisher.
7	Reference 11: Please provide the editor name(s), the full date of the conference, the paper number, city of publication and the name of the publisher.
8	Reference 13: Please provide the editor name(s), the full date of the symposium, the paper number, city of publication and the name of the publisher.
9	Reference 16: Please provide the editor name(s), the full date of the conference, the paper number, city of publication and the name of the publisher.
10	Reference 17: Please provide the editor name(s), the full date of the conference, the paper number, city of publication and the name of the publisher.
11	Reference 18: Please provide the editor name(s), the full date of the conference, the paper number, city of publication and the name of the publisher.
12	Reference 21: Please provide the editor name(s), the full date of the conference, the paper number, city of publication and the name of the publisher.
13	Reference 23: Please provide the city of publication.
14	Reference 27: Please provide the editor name(s), the full date of the conference, the paper number, city of publication and the name of the publisher.
15	Please provide references 28 and 29.

Simulation

Simulation: Transactions of the Society for Modeling and Simulation International
1–14
© The Author(s) 2018
DOI: 10.1177/0037549718815338
journals.sagepub.com/home/sim
SAGE SCS

Realization-preserving model reduction of object-oriented models using energy-based metrics

Anton Sodja, Igor Škrjanc and Borut Zupančič

Abstract
This paper deals with the adaptation of energy-based realization-preserving model-reduction techniques for object-oriented models implemented in Modelica. An approach originally developed for bond graphs, where energy-related heuristics were used to identify the componentf8.s contributing the most to the model's salient dynamics, was extended to more heterogeneously formulated models in Modelica. A prototype toolbox was developed inside the modeling environment OpenModelica. Its applicability was demonstrated with two examples. Although the model-reduction tool cannot reduce the model completely automatically, it may still offer many useful insights and hints to the user.

Keywords
Object-oriented modeling, model simplification, realization-preserving modeling, model reduction, energy-based ranking metrics, Modelica

1. Introduction

A central guideline in modeling is that a model should not be more complex than is necessary for a given purpose. So, possible simplifications must be thought of in all modeling phases. Good and usable models are usually appropriately simplified models. However, it is not always obvious what to include in the model and what to leave out. A more accurate model usually also needs to be more complex, which can reduce the physical insights provided by the model and can be inadmissable for some applications (e.g., control design). Therefore, a trade-off between the accuracy and the simplicity of the model must be also taken into account. Although this paper is focused on a very specific realization-preserving model reduction in Modelica, perhaps the most important modeling phase – model validation – must not be forgotten. This is a very complex procedure that checks also the complete range of possible model operations. If the complete model (before simplification) is supposed to work for a wide range of operations, it is obvious that the simplified model probably works in more limited area. These restrictions or limitations should be known when using simplified models.

The proper complexity of a model is even more difficult to achieve in a component-based modeling approach because a rich selection of detailed components is often available, which makes it possible to build very complex models at the outset.[1] However, in many cases the model is found to be too complex and is subsequently simplified. This reduction of the model's complexity is an intricate and tedious task. Consequently, numerous automatic model-reduction methods have been developed and the subject remains an active research topic.[2] In some fields, for example, integrated circuit design, they have become an indispensable part of the system analysis and hence are provided as part of domain-specific modeling environments.[3]

There are numerous approaches to reduce the complexity of a model. All of them depend to a large extent on the problem domain, the modeling methodology used, and the purpose of the model.[2] The fewer restrictions that are imposed on a reduced model the more efficient the reduction algorithm can be. Hence, the most successful model-reduction methods are mostly not realization-preserving – the reduced model retains only the input–output behavior of the system and loses the physical interpretability of its structure and parameters. For example, model reduction using principal component analysis (PCA; also referred to as the Karhunen–Loève expansion)[4] transforms a set of state variables of a model to a different set of state

Faculty of Electrical Engineering, University of Ljubljana, Slovenia

Corresponding author:
Borut Zupančič, Faculty of Electrical Engineering, University of Ljubljana, Tržaška 25, Ljubljana, 1000, Slovenia.
Email: borut.zupancic@fe.uni-lj.si

variables that are usually no longer physically interpretable. Furthermore, in some cases it may be no longer possible to simulate the reduced model with the simulator of the modeling environment that was used to design the full model.[5]

Although the preservation of realization is a very suitable and desirable property in many modeling applications, realization-preserving reduction methods are not so frequently dealt with in the literature or used in applications, because the reduced models usually have diminished performances, at least in comparisons with reduction approaches where structures between inputs and outputs can be arbitrarily changed. Moreover, most of the existing realization-preserving methods are limited to certain types of models, for example, RC circuits,[5] or a modeling methodology, for example, bond graphs.[6] Therefore, the implementation of these approaches in modeling Modelica-like environments that support the building of very heterogenous and multi-domain models with mixed modeling methodologies is not a straightforward task. There are no realization-preserving model-reduction methods known to the authors that could adequately handle multi-domain models implemented in contemporary object-oriented modeling languages, such as Modelica.

This paper focuses on lumped-parameter, continuous-time, and deterministic dynamic models and attempts to generalize the realization-preserving model-reduction techniques developed for bond graphs so as to be also applicable to more general and heterogenous models implemented in Modelica.[7,8]

The structure of the paper is the following: in Section 2 the basic idea of the realization-preserving model reduction is presented. The types of elementary reduction operations are described. Then the general procedure for model reduction and appropriate ranking metric are introduced. Section 3 is devoted to the energy-based ranking metrics developed to evaluate the importance of model components. Section 4 describes the reduction of object-oriented models in Modelica. Here the problems of energy-flow calculations from the variables defined in connector definitions is discussed. The ranking procedure and component elimination are implemented by modifying the Modelica translation process, adding instrumentation, metric calculation, and ranking. Section 5 includes illustrative examples: a simplified car suspension model and a direct current (DC) motor model. Section 6 briefly discusses the reduction based on model equations.

2. Realization-preserving model reduction

All realization-preserving model-reduction techniques use the same strategies to reduce the complexity of the model: in order to preserve the realization, parts of the model that contribute to the negligible dynamics of the system can either be completely removed or replaced with simpler parts.

The choice of elementary reduction operations (i.e., the removal or replacement of an elementary part of the model) depends strongly on the modeling methodology: the more complex the modeling formalism (e.g., modeling language) used, the more complex the reduction steps that must be undertaken in order to enable sufficient reduction and simplification of the model.

Furthermore, the sequence and applicability of the reduction operations must be determined. This is accomplished by calculating the error prediction of each reduction operation with respect to an objective function (i.e., ranking metric) and applying reduction operations whose cumulative error prediction falls below a certain threshold. Most commonly, error predictions are calculated from the model's response, which is obtained by a simulation. Therefore, the general procedure for model reduction consists of the following steps:

1. a series of simulation runs;
2. individual reduction operations are ranked by the appropriate metric;
3. reduction operations that fall below a certain threshold are applied;
4. the reduced model is verified by simulation.

Besides the chosen set of appropriate elementary reduction operations, which is usually fairly limited in the case of realization-preserving model reduction, the ranking metric is the most important aspect of reduction methods that influences their effectiveness.

The most straightforward ranking metric is simply to perform simulation runs for all possible reduction operations and compare the responses of the full and reduced models for particular variables of interest through particular error or criteria evaluations. However, such an approach is obviously too computationally demanding to have a practical meaning. In practical solutions, a trade-off between the accuracy of a reduction solution and the computational efficiency is sought.

However, it is usually difficult to efficiently predict the error that the reduction operation would introduce to the model in absolute terms, that is, the difference between the full and reduced model trajectories. Therefore, alternative ranking metrics are frequently used to produce a relative ranking, that is, only the order of the reduction operations according to their impact on the model is provided. These metrics are mostly based on intuition, for example, the characteristic time constant of each part of the model is calculated and then those parts outside the frequency range of interest are removed. Similarly, the energy associated with submodels can be used to identify submodels that are contributing (in)significantly to the dominant system dynamics.

The choice of the ranking metric is closely related to a model formulation from which the required quantities (e.g., characteristic time constants, energy, etc.) for the metric evaluation should be easily extracted.

3. Energy-based ranking metrics

Successful ranking metrics based on intuition are usually related to energy or power. A basic presumption of energy-related metrics is that the components associated with negligible energy, that is, a little energy is stored or dissipated by the component in comparison with the other components, also negligibly influence the dominant system dynamics and can thus be eliminated from the model without changing its behavior significantly.

Although the majority of analytical models are derived from first principles, that is, the conservation of energy, the energy or power is usually not explicitly available in the model, that is, it is not a variable of the model. Therefore, an explicit procedure for energy or energy-flow calculations must be explicitly provided for the model (e.g., Lyapunov function[9]) 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 modeling formalism, where the energy flow associated with each component (i.e., the elementary submodel) is straightforward to determine. Each component is connected with a bond (acausal link) to the rest of the model and the bond consists of a pair of variables whose product always equals the power (or energy).

Some of the ranking metrics developed to evaluate the importance of components or bonds (connections) for either model reduction[6,10] or model partitioning[11] are briefly presented in this section. Because they rely on the results obtained by the simulation and therefore depend strongly on the chosen simulation scenario (i.e., model excitation), the resulting component ranking is only valid inside a limited range.

3.1. Root mean square power

Rosenberg and Ermer[12] used bond graphs to visualize the dynamic response. Based on their simulation experiments they decided to use the RMS (root mean square) value of the energy flow (for the time interval $[t_1, t_2]$) associated with each bond (connected to a component) as a time-independent measure:

$$\mathcal{P} = \sqrt{\frac{1}{t_2 - t_1} \int_{t_1}^{t_2} \left(\sum_j \dot{e}_j(t) \right)^2 \cdot dt} \tag{1}$$

In Equation (1), $\dot{e}_j(t)$ designates the jth energy flow through the boundary of an element. Each bond was colored according to the hierarchical color scheme.

Rosenberg and Zhou[13] suggested that the bonds (and associated submodels) with low RMS power could be omitted without introducing a significant error. However, Rosenberg and Zhou did not provide any systematic procedure for the model reduction.

3.2. Activity

Louca[14] introduced the *activity* of elements, an integral of the absolute value of all the energy the element (submodel) has exchanged with its surroundings within a given time interval $[t_1, t_2]$:

$$A_i = \int_{t_1}^{t_2} |\sum_j \dot{e}_j(t)| \cdot dt \tag{2}$$

In Equation (2) $\dot{e}_j(t)$ has the same meaning as in Equation (1).

Activity has a physical meaning: it represents the amount of energy that flows through the element during a given time interval. It differs from the total RMS energy flow of an element by putting less weight on the peak values, since it uses the absolute norm instead of the square averaging.

Before the element ranking, the activities of all the elements should be normalized, which means that they are divided by the sum of all the element activities (the total activity of the system):

$$\mathrm{AI}_i = \frac{A_i}{\sum_{j=1}^{n} A_j} \tag{3}$$

so that a time-independent measure is obtained. The normalized activity measure was dubbed the *activity index* (AI) by Louca.[14]

3.3. Karhunen–Loève expansion of the bond energy flows

The RMS power and activity metrics depend solely on the magnitude of the energy flows associated with the bonds. Ersal[15] tried to further improve the relative importance of the evaluation of energetic connections by also considering the correlations between energy trajectories. Therefore, he applied the Karhunen–Loève expansion (also referred to as PCA) of the energy-flow trajectories.

Firstly, all of the energy-flow trajectories are arranged column-wise in the matrix $\mathbf{S}$:

$$\mathbf{S} = [\dot{\boldsymbol{e}}_1 \quad \dot{\boldsymbol{e}}_2 \quad \ldots \quad \dot{\boldsymbol{e}}_n] \tag{4}$$

Each trajectory $\dot{e}_i$ in Equation (4) is a vector consisting of m observations and the matrix $\mathbf{S}$ consists of n trajectories.

Secondly, a singular value decomposition of the matrix $\mathbf{S}$ yields the following:

$$\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} \tag{5}$$

In Equation (5), the matrix $\mathbf{\Sigma}$ is a diagonal matrix with singular values $\sigma_1, \sigma_2, \ldots\sigma_n$ on the diagonal and the columns of $\mathbf{U}$ and the rows of $\mathbf{V}$ are the left- and right-hand singular vectors, respectively.

Finally, the *importance vector* I_i is calculated from the singular values and vectors of the matrix $\mathbf{V}$ as follows:

$$I_i = \sum_{j=1}^{n} \sigma_j^2 |v_{i,j}| \quad \text{for } i = 1, 2, \ldots, n \tag{6}$$

The relative importance of the element is obtained by normalizing the importance vector of the associated bond.

3.4. Relative energy

The energy-related ranking metrics described so far perform badly when used on models with a salient dynamics that is sensitive to elements associated with a small energy. They also fail to eliminate the elements associated with high energy levels contributing little to the system dynamics.

In order to overcome these limitations, Ye and Youcef-Toumi[16] developed a new metric. The basic premise of their metric is that the dynamic behavior of an individual independent energy-storage element is dictated by the energy exchange with the neighboring bonds. Neighboring bonds are bonds connected to the same junction as the bond associated with the energy-storage element. Since energy is conserved in the exchange, the change of energy associated with the element, $\Delta E_1(t)$, can be expressed as the sum of the energy changes of the neighboring bonds, where n is the number of bonds in the junction:

$$\Delta E_1(t) = \sum_{j=2}^{n} \Delta E_j(t) \tag{7}$$

Each independent storage element introduces a state variable into the model and the change of the state variable Δx_1 attributed to the removal of the ith neighboring bond, which has a low energy level, is stated by the following linearized expression with respect to $E_i(t)$:

$$\Delta x_1(t) = \Psi(\Delta E_1(t)) - \Psi'(\Delta E_1(t))\Delta E_i(t) \tag{8}$$

In Equation (8), $\Psi(\cdot)$ designates the function that determines a state variable from the energy of the element and $\Psi'(\cdot)$ is its derivative: $\Psi'(x) = d\Psi(x)/dx$.

Two propositions are derived from Equation (8) for the determination of the bonds connected to a junction, only negligibly affecting the dynamic behavior of the ith storage element, which is also connected to the same junction:

$$|| \sum_{k=1}^{m} |\Delta E_k(t)| ||_\infty \ll ||\Delta E_i(t)||_\infty \tag{9}$$

$$||\Psi'(\Delta E_i(t)) \sum_{k=1}^{m} |\Delta E_k(t)| ||_\infty \ll ||\Psi(\Delta E_i(t))||_\infty \tag{10}$$

Equation (9) requires that the sum of the energy flows of m bonds, which may be removed, must be much smaller than the energy flow of the ith storage element connected to the same junction. Furthermore, the sensitivity of the storage element, the second term on the right-hand side of Equation (8), is taken into account by Equation (10). In Equations (9) and (10), $||\cdot||_\infty$ denotes the infinity (Chebyshev) norm.

Finally, a set of all the negligible bonds in the model is a union of all the bonds that can be removed locally according to Equations (9) and (10).

4. Reduction of object-oriented models in Modelica

4.1. Energy in Modelica models

Object-oriented models implemented in Modelica use similar structuring concepts as bond graphs and thus it is not unreasonable to expect that bond-graph reduction techniques can be also used for models in Modelica with appropriate adaptations. However, Modelica leans toward generality and does not make any assumptions about the physics of the modeled system in order to optimize the translation or simulation of the model. Therefore, there is no notion of energy or power in the language itself or, in other words, information about the energy calculation must be provided explicitly to the reduction algorithm so that an energy-based ranking of the components can be calculated.

The structuring concept of Modelica is illustrated in Figure 1.

Submodels, which in Figure 1 are represented by squares, are connected with each other, that is, through their ports, which are called *connectors* in Modelica. The connections represent interactions between the submodels, and the type of the interaction is defined by a connector definition. The latter is a list of variables, and for all mutually connected connectors (i.e., a connection set) appropriate equations are generated that relate the corresponding variables of the connectors. The kind of equations that need to be generated is defined at the declaration of the connector variables. A variable in the connector can be either the potential (effort in bond-graph terminology)

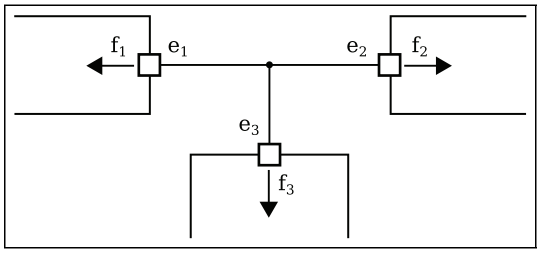

Figure 1. The connection of three components.

variable e_i, which means that the variables in a connection set are equalized:

$$e_i = e_j \quad \forall i \neq j \tag{11}$$

or a flow variable f_i, which means that the sum of all the variables in a connection set equals zero:

$$\sum_i f_i = 0 \tag{12}$$

There is also a special type of connector variable, called *stream*, that is intended for modeling interactions including mass and energy transfer. No equations are generated by default for this variable (instead, special operators are provided to generate equations depending on the context in which they are used).

In Modelica (using the Modelica bond-graph library[17]) it is possible to describe a model with a bond graph by defining the connections with the bonds, each having a pair of variables, potential and flow, respectively, whose product is the power. In this case the model-reduction methods described in the previous section can be used directly. However, the majority of model libraries in Modelica do not use bond-graph formalism and there is no uniform way to calculate the energy associated with a component or a connection provided by most of the libraries. Therefore, this information must be supplied explicitly to the model-reduction method. The most sensible approach is to calculate the energy flow of the connections, because usually very few different types of connectors are used, and then the net energy flow to the component can be obtained from the law of energy conservation:

$$\dot{e}_i(t) = -\sum_j \dot{e}_{i,j}(t) \tag{13}$$

In Equation (13), $\dot{e}_i(t)$ denotes the net energy flow of the ith component and $\dot{e}_{i,j}(t)$ represents the energy flows of its connections (i.e., the energy exchanged with the environment).

Nevertheless, this approach is only applicable if the energy flow can only be calculated from the variables declared in the connector definition – usually this is the case. An investigation of connector definitions in the Modelica Standard Library, a collection of the most common components from all the major physical domains, proved that all connectors of the library are defined appropriately.[18,19] Some are analyzed in the next section.

4.2. Analysis of energy interactions in Modelica libraries

As mentioned, connectors usually contain a pair of potential and flow variables. However, their product is not necessarily an energy flow like in bond-graph formalisms. This can be seen by inspecting the Modelica Standard Library,[20] where elementary connector definitions for almost all physical domains are gathered.

- Interaction between components in analog circuits (*Modelica.Electric*) is determined by voltage v and current i, the latter is a flow variable, and the power of the interaction is the product of both variables: $\dot{e} = v \cdot i$.
- Similar features also have a connector in *Modelica.Magnetic*, which is composed of variables for magnetic potential difference V_m and magnetic flux Φ, potential and flow variables, respectively. The power of the connection is the product of both variables: $\dot{e} = V_m \cdot \Phi$.
- Connectors used for modeling one-dimensional (1D) translational and rotational mechanical systems consist of position s and angle ϕ, respectively, and force f and torque τ, respectively. However, the product of connector potential and flow variables is no longer the power. For determination of the power of the connection, the displacement variable has to be differentiated: $\dot{e} = \frac{d}{dt}s \cdot f$ and $\dot{e} = \frac{d}{dt}\phi \cdot \tau$ for translational and rotational mechanics, respectively.
- In the *Modelica Multibody* library, which deals with three-dimensional (3D) mechanics, potential and flow variables are no longer scalars; rather, they are six-dimensional vectors, so the state of a free-body (having six degrees-of-freedom) can be determined. Furthermore, due to computational restrictions, implementation of the connector takes into account also the suitable selection of a frame of reference (forces, torques, and orientation are expressed in the local frame of reference, while position is in the global frame of reference). A definition of the connector is as follows:

```
connector Frame
SI.Position r_0[3];
Frames.Orientation R;
flow SI.Force f[3];
flow SI.Torque t[3];
end Frame;
```

The position is determined with the variable *r*_0, while the orientation *R* is a structure containing the transformation matrix *T* from the global to the local frame of the reference and the vector of angular velocities $\boldsymbol{\omega}$ in the local frame of reference. Forces and torques are given by vectors $\boldsymbol{f}$ and $\boldsymbol{t}$, respectively. The power of the connection can be calculated by the expression $\dot{e} = \frac{d}{dt}(\mathbf{T} \cdot \boldsymbol{r}_0) \cdot \boldsymbol{f} + \boldsymbol{\omega} \cdot \boldsymbol{t}$, where again there is a need to differentiate the position after transformation to the local frame.

- The connector for modeling the heat transfer in one dimension consists of the potential variable for temperature *T* and the flow variable for the heat-flow rate Q_{flow}. The energy transfer is in this case equal to the flow variable $\dot{e} = Q_{flow}$.
- The library *Modelica.Fluid* deals with the modeling of heat and mass transfer. The connector used in the library components that also covers mass transfer is implemented as follows:

```
connector FluidPort
replaceable package Medium =
Modelica.Media.Interfaces.PartialMedium;
flow Medium.MassFlowRate m_flow;
Medium.AbsolutePressure p;
stream Medium.SpecificEnthalpy h_outflow;
stream Medium.MassFraction
Xi_outflow[Medium.nXi];
end FluidPort;
```

Besides potential and flow variables, pressure *p* and mass-flow rate m_{flow}, respectively, the connector includes also additional information about properties of the substance that is being exchanged in the interaction modeled by a connection of type *FluidPort*: specific enthalpy *h* and composition of the substance (vector of mass fractions X_i if the substance is a mixture). The thermodynamic state of the substance is uniquely determined by the variables of the connector and all the other (thermodynamic) properties can be calculated by using functions provided by the package *Medium*, which is a parameter of the connector. However, thermal diffusion is not covered by this connector (it is neglected).

Energy flow associated with the connector is composed of thermal, hydraulic, and chemical terms and can be calculated as follows[21]: $\dot{e} = \dot{m} \cdot s \cdot T + \dot{m} \cdot p/\rho + \sum \mu_i \cdot \dot{N}_i$. Quantities of specific entropy *s*, temperature *T*, density ρ, chemical potential μ_i, and molar flow $\dot{N}_i$ can be calculated from the thermodynamical state equations provided by the package *Medium*.

Although it is possible to calculate the energy flow of the connector from the variables of the connector, this can be problematic to do from simulation results, because some variables may not be available. For example, the derivative of a position or an angle in the connector of the library for 1D mechanics may not be available if this variable is not chosen as a state, input variable, or output variable. This problem is solved by proper instrumentation, which will be described in the next section.

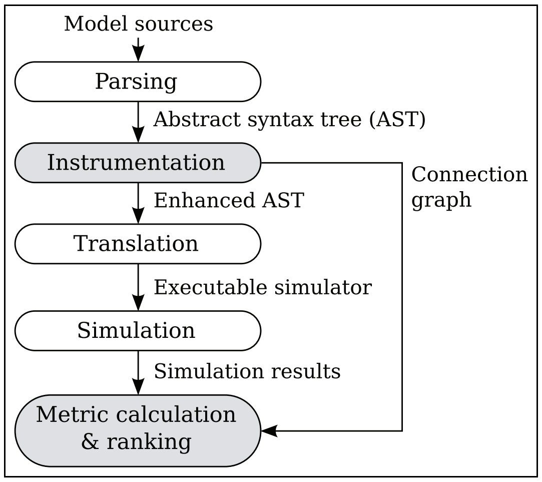

Figure 2. Modified translation process for the implementation of component ranking with energy-based metrics.

4.3. Instrumentation and ranking procedure

We implemented a prototype of instrumentation and component ranking in the OpenModelica framework.[8,22] The original model is automatically extended with energy equations during the model translation. The metric is then calculated using simulation results in the postprocessing phase. The implementation, which shows two important add-ons in the modified translation – instrumentation, and metric calculation and ranking – is presented in Figure 2.

The overall procedure can be described as follows.

- After the model sources are parsed and the abstract syntax tree (AST) is generated, the instrumentation is performed.
- Firstly, all appropriate connection statements are extracted. This is done by traversing the model equations (AST) and, whenever a connection statement that describes a physical interaction is encountered, it is added to the list.
- A set of all admissible connector types and how to calculate the corresponding energy flow for each is given as a Modelica package. The package contains a component wrapping equation(s) for energy-flow calculation(s) for each connector type. An implementation example of such a component for the connector from the `Mechanics.Translational` library is as follows:

```
block MechanicsTranslational
input SI.Position s;
input SI.Force f;
output SI.Power power;
annotation(connectorType=
{"Modelica.Mechanics.Translational
.Interfaces.Flange_a",
"Modelica.Mechanics.
Translational.Interfaces.Flange_b"});
equation
power = der(s)*f;
end MechanicsTranslational;
```

The component is the block with inputs (potential and flow variables `SI.Position s` and `SI.Force f`) from the connector definition and output – the energy flow is named `power`. In the equation section, the energy flow is calculated. The component is matched with the connection type by the use of the tool-defined annotation `connectorType` where fully qualified names of matching connector types are stated.

- Simultaneously, the connection graph with information about the matching of the added equations with corresponding connectors is composed and stored in the environment. Using this information, relevant components and their energy flows can be extracted after the simulation.
- For each connection describing physical interaction, an equation for energy flow of this interaction is added into the model.
- The described instrumentation approach ensures that all the required information needed for further simulation and metric calculation and ranking is available (e.g., variables for appropriate metric calculation are available).
- Simulation of the expanded model is performed and energy flows associated with components are calculated using Equation (13) and the information provided by the connection graph.
- Then the importance of each component according to a selected ranking metric (e.g., Equation (2) or (3)) can be evaluated and, finally, the components are sorted according to their importance. This concludes the ranking procedure.

4.4. Component elimination

Ranking the components of the model is just the first step of model reduction. It is followed by the elimination of the low-ranked components or their replacement with simpler ones.

The method of Louca[14] relies on the classification of elements, enabling the interpretation of eliminated elements in an idealized physical model. The same approach is also applicable for Modelica models by compiling a database of rules on how to simplify each component, which is of course a tedious task, especially when third-party libraries are to be included. For example, if a resistor is associated with low energy, but the connections of its pins are not, it should be replaced with a short circuit, otherwise with an open circuit, etc. The removal of a component with three pins (e.g., a transistor) requires even more rules.

However, it is difficult to conceive very general rules for component removal because it depends on the definition of the connector. In some cases the eliminated component must be replaced with a component that enforces the appropriate boundary conditions (e.g., if a position is used as a connector variable instead of the velocity and a zero-velocity boundary condition is required).

An additional option for the automatic removal of the components is to flatten each component and analyze its equations using a designated algorithm. However, in order to reduce very heterogenous models completely automatically, many aspects and side-effects of the component removal must be considered and, consequentially, the algorithm becomes very complicated and might still need manual intervention in many cases.

Finally, it is reasonable and efficient if the task of component elimination can be left to the user. For many purposes just the ranking of the components is sufficient. For example, when a model is inspected for components that are too complicated or should be more detailed, an explanation for a specific model behavior is sought, etc.

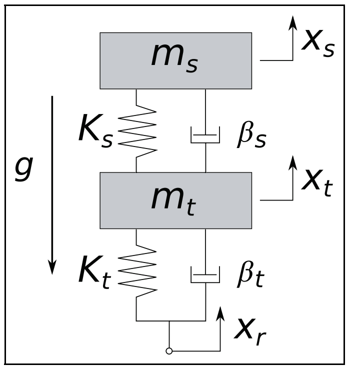

Figure 3. Ideal physical model of a car suspension.

5. Illustrative examples

5.1. A simple mechanical system

In Figure 3, a model of a simplified car suspension is depicted.[14]

The model is composed of a wheel with a tire with mass m_t, compliance K_t, and damping β_t. Above the wheel is a suspension system with compliance K_s and damping β_s, and above that is the remaining mass m_s (mass of the system), which consists of the car body mass

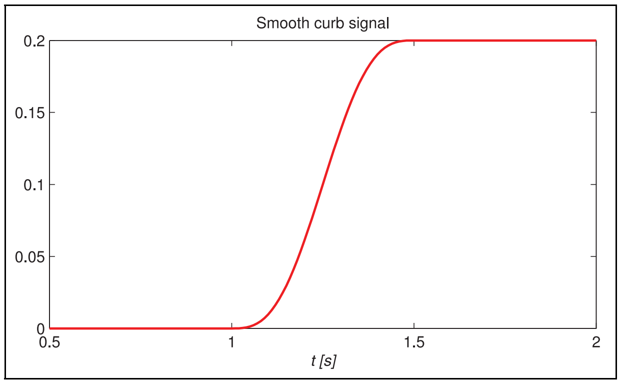

Figure 4. Smooth curb signal.

Table 1. Values of the model parameters.[AQ: 1]

$m_t = 36.6$ [kg]	$m_s = 267.0$ [kg]
$\beta_t = 200.0$ [N · s/m]	$\beta_s = 700.0$ [N · s/m]
$k_{t,1} = 193.915$ [N/m]	$k_{s,1} = 18.742$ [N/m]
$k_{t,3} = 2 \cdot 10^8$ [N/m^3]	$k_{s,3} = 2 \cdot 10^5$ [N/m^3]

and the mass of the suspension system. The mass m_s is much larger than the mass of the wheel, and also the compliance and damping of the suspension system, K_s and β_s, are much larger than that of the wheel. Both springs have nonlinear characteristics:

$$F_{Ks} = K_{1s}(x_s - x_t) + K_{3s}(x_s - x_t)^3$$

$$F_{Kt} = K_{1t}(x_t - x_r) + K_{3t}(x_t - x_r)^3$$

The input to the system is the elevation of the ground x_r. The car hits a smooth curb, which is modeled with the following function (see the input signal x_r in Figure 4):

$$x_r(t) = \begin{cases} 0.4(t-1) - \frac{0.1}{\pi} sin(4\pi(t-1)) & ; \quad 1 \leqslant t \leqslant 1.5 \\ 0 & ; \quad t < 1 \\ 0.2 & ; \quad t > 1.5 \end{cases} \tag{14}$$

The reference (and initial) elevation is defined to be zero. The parameters of the model have the values shown in Table 1.

[AQ: 2]The corresponding model is built in Modelica using components from the Modelica Standard mechanical library except the spring components, which have a nonlinear constitutive law and are custom made. The schematic diagram of the model is shown in Figure 5.

There is only one type of interaction between the components, that is, translation in one dimension, and consequently only one type of connector is used. This connector consists of two variables: the absolute position s and the

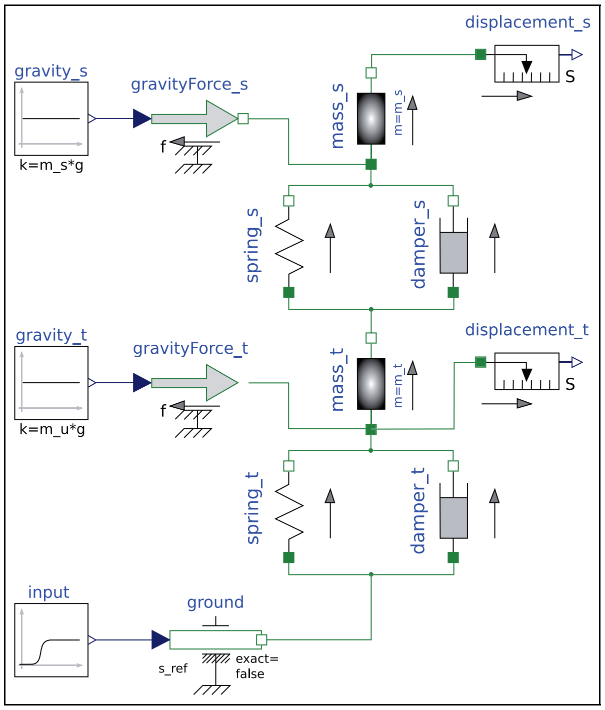

Figure 5. Realization of the model of the car suspension system as a Modelica object diagram.

force f. Thus, the equation for a determination of the energy flow $\dot{e}$ is as follows:

$$\dot{e} = \frac{d}{dt} s \cdot f \tag{15}$$

In the first experiment the model is excited with a smoothed step signal. Figure 6 depicts the input signal and the appropriate responses – displacements of the mass of the wheel *mass_t* and of the system (body of the car) *mass_s*.

The components are ranked according to the importance determined by the *activity* metric in the first case and the PCA of the component power (Karhunen–Loève expansion) in the second case. The results are shown in Tables 2 and 3, respectively.

There are no significant differences between the rankings in Table 2 and Table 3, which is to be expected for such a simple system. The most insignificant components having the lowest relative activity (importance) are `damper_t`, `mass_t`. If they are eliminated from the model, the response of the model, with the same input signal, changes only slightly: the RMS errors of displacement of the wheel and the system (car body) are only 0.06 and 0.03 cm, respectively. An idealized physical representation of the reduced model and its realization in Modelica are illustrated in Figures 7 and 8, respectively.

Note that the same reduced model as depicted in Figure 7 was obtained using simplifications with theoretical modeling by Matko et al.[23]

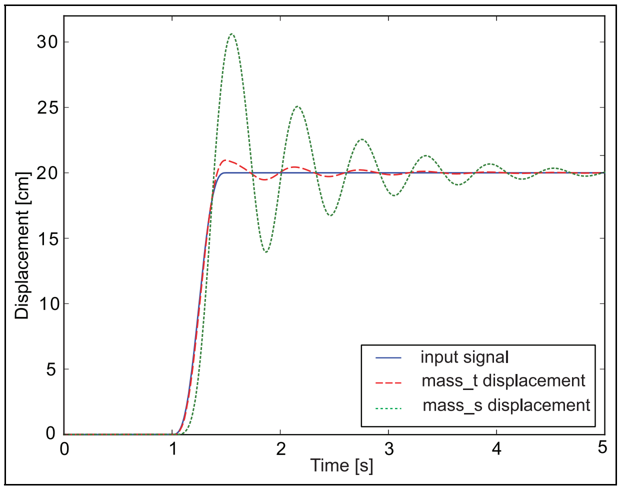

Figure 6. Excitation signal of the simple mechanical system model and appropriate responses.

Table 2. Component ranking of the model from Figure 5 using the activity metric for the smoothed input signal.

Element	Activity [J]	Relative [%]	Accumulated [%]
gravityForce_s	2270.06	37.06	37.06
spring_s	1763.33	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

Table 3. Component ranking of the model from Figure 5 using the principal component analysis of the component power and smoothed input signal.

Element	Relative importance [%]	Accumulated [%]
gravityForce_s	30.84	30.84
spring_s	26.13	56.97
ground	25.83	82.80
mass_s	10.18	92.97
spring_t	2.76	95.73
gravityForce_t	2.39	98.12
damper_s	1.43	99.56
mass_t	0.44	99.99
damper_t	0.01	100.00

In the second experiment, an exact (sharp) step signal was used instead. The response of the system is illustrated in Figure 9.

Comparing Figures 9 and 6, we notice much higher frequencies when the system is excited with the sharp step

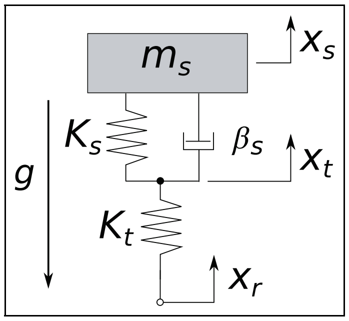

Figure 7. Reduced model from the first experiment.

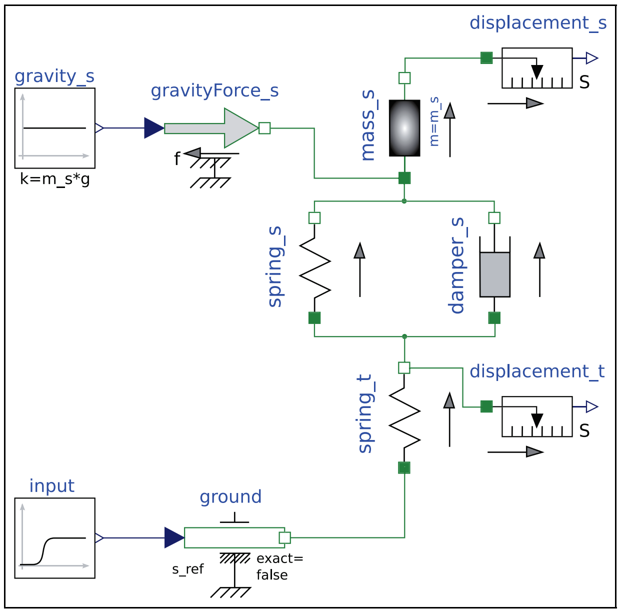

Figure 8. Realization of the reduced model from the first experiment in Modelica.

signal. The component ranking using the *activity* metric is given in Table 4.

The component rankings given in Tables 2 and 4 differ at first glance because a different input signal was used, the latter having a higher frequency range than the former. Therefore, the components `mass_t` ($m_t \ll m_s$) and `spring_t` (much stiffer than `spring_s`) are of the highest rank in Table 4, while mass_s has a low rank. If we follow the same procedure as in the previous experiment, the components having lowest relative activity are removed from the model. However, `ground` is a restricted component as it is connected to the input signal and hence cannot be removed. Instead, the components `gravityForce_t` and `mass_s` are removed from the model so that the reduced model depicted in Figures 10 and 11 is obtained.

A comparison with the reduced model from Figure 7 shows that the removal of a component representing a

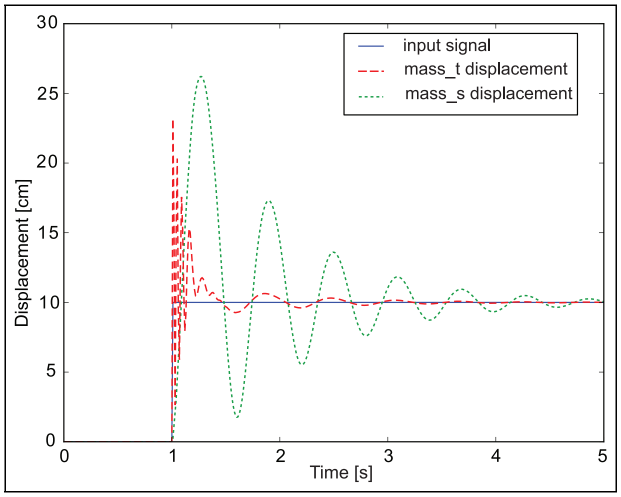

Figure 9. The response of the mechanical system depicted in Figure 5.

Table 4. Component ranking of the model from Figure 5 using the activity metric with a sharp step signal at the input.

Element	Activity [J]	Relative [%]	Accumulated [%]
mass_t	$8.16 \cdot 10^4$	43.42	43.41
spring_t	$7.92 \cdot 10^4$	42.10	85.52
spring_s	$9.33 \cdot 10^3$	4.96	90.48
damper_s	$8.75 \cdot 10^3$	4.65	95.13
gravityForce_s	$2.78 \cdot 10^3$	1.48	96.61
damper_t	$2.39 \cdot 10^3$	1.27	97.88
mass_s	$1.93 \cdot 10^3$	1.02	98.91
ground	$1.60 \cdot 10^3$	0.85	99.76
gravityForce_t	$4.51 \cdot 10^2$	0.24	100.00

body with mass (e.g., components `mass_t` and `mass_s`) can be interpreted in two different ways:

1. the dynamic force of the body is considered negligible, that is, its mass is small ($m \approx 0$);
2. the movement of the body is considered negligible, that is, it has a very large mass ($m \to \infty$).

In the former case the connections of the components are joined into one connection point representing a massless body (m_t in the first experiment) and in the latter case the connections are grounded (m_s in the second experiment).

Furthermore, special attention must be given to the elimination of the components from the Modelica model, that is, how the removal of a component will affect the initialization problem. Inappropriate removal means that the solver will not be able to initialize the model. For example, grounding the component `mass_s` causes the removal of two degrees of freedom. If the simulation starts from the steady state for `damper_s` and `damper_t`, the

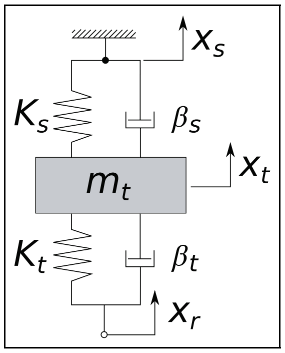

Figure 10. Reduced model from the second experiment.

initial values of the displacement velocity cannot be set for both of them (which was possible in the full model). In that case, removing `mass_s` from the model thus also requires the reconfiguration of either `damper_s`, `damper_t`, or `mass_t`.

As mentioned before, the step signal is a relatively frequency-rich signal and all the components of the model become excited, so the model cannot be significantly reduced. However, the *activity* metric produces misleading results. It only gives merits to the fast dynamics and hence an error in the behavior of the reduced model in comparison with the full model is significant. The responses of the full model (Figure 3) and the reduced model (Figure 10) excited by the sharp step signal are shown in Figure 12. The displacement of the mass $mass_s$ is totally mismatched due to the grounding of the component and, for the same reason, the displacement of the wheel has an error in the steady state.

It is also a good example showing that the most important modeling phase – model validation – must be considered. It also checks operating conditions – in this case the shape of the input signal. If the complete model (before model reduction) is supposed to work for different input signals, it is obvious that the simplified models can only be used with limitations (e.g., when the sharp step signal is implemented, the reduced model takes into account the fast dynamics and satisfactorily describes only the movement of the wheel).

5.2. DC motor

Another illustrative example is the model of an electrical DC motor[23,24] that can be found in the Modelica Standard Library (Modelica.Mechanics.MultiBody.Examples.Systems.Robot R3.oneAxis) and is shown in Figure 13.

The model consists of components describing the mechanical and electrical properties of the motor (`Ra`, `La`, `emf`, `Jmotor`) and its control electronics: a proportional–integral (PI) controller realized with analog electrical

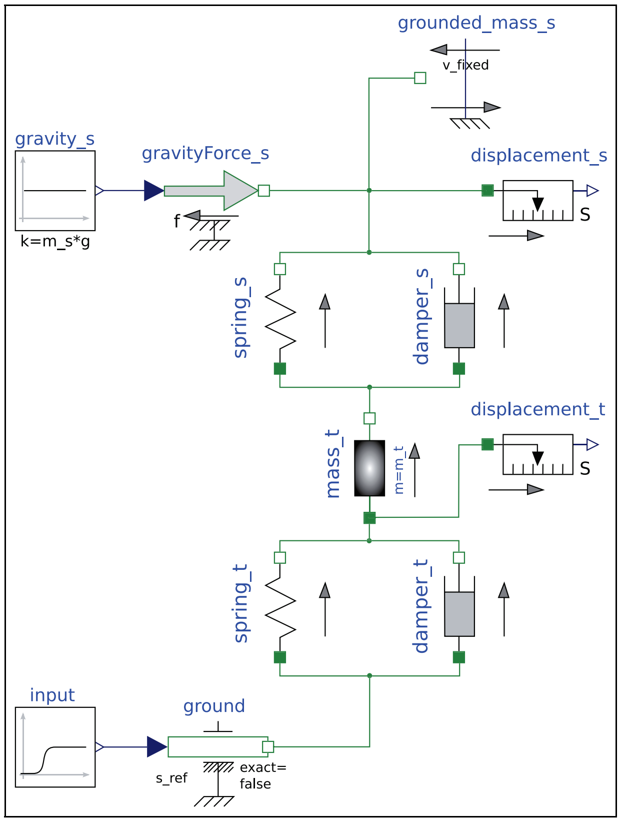

Figure 11. Realization of the reduced model from the second experiment in Modelica.

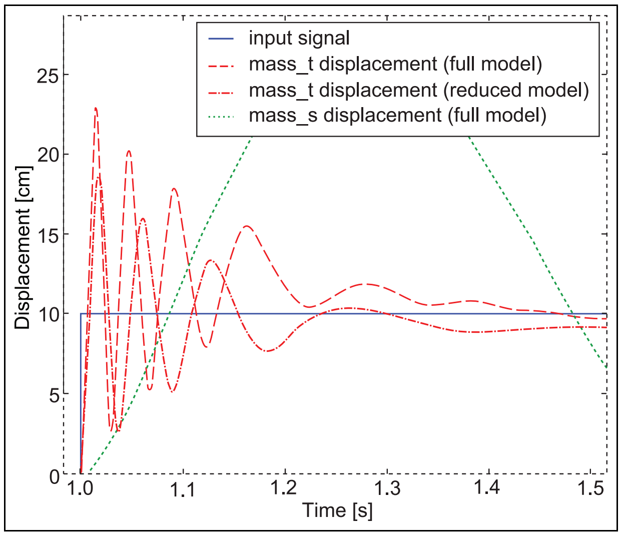

Figure 12. Comparison of the full and reduced model responses to the sharp step input signal.

components (`OPI`, `C`, and `Ri`), a power amplifier (`power`, `Rp1`, and `Rp2`), etc.

The results of the component rankings for the DC motor model using the activity metric are shown in Table 5.

The highest-ranked components are the components comprising the electrical and mechanical properties of the motor and the power-amplifier components (components from `Rp1` to `La`). Considering only those components, the ranking results suggest that the component `La`– armature inductivity (i.e., inductivity of the motor windings) – has an insignificant effect on the model dynamics and can thus be removed. This is the well known simplification usually used when modeling electrical DC motors.[23,24]

The remaining components, which belong to the model of the motor controller, can be found at the bottom of Table 5 with an AI that is an order of magnitude smaller because they are associated with much less energy than the power part of the motor. However, a model of the controller or any of its parts cannot be removed from the model without significantly changing the model behavior.

The ranking presented in Table 5 must therefore be carefully used and the physical understanding is usually important. Definitely, we can obtain useful information that can be used for ‘‘manual’’ reduction. We doubt that the reduction process can be fully automated.

The ranking based on the Karhunen–Loève expansion of the component energy flows and the calculation of the importance vectors using Equation (6) generates very similar results. Some improvements can be obtained with a normalization of the energy-flow trajectories prior to the Karhunen–Loève expansion being performed.

The weakness of the energy-based metrics could be mitigated by also considering the sensitivity of the energy flow to the removal of the neighboring components (connections), as suggested by Ye and Youcef-Toumi.[16] However, it is difficult to implement this method for most Modelica models, because it requires each energy-storing element in a separate component to have a function mapping the energy flow back to the state variable available.

An example of model reduction based on object diagrams and energy-based metric modeling of thermal and radiation flows in a building is described and analyzed by Zupančič.[25]

So, the component elimination of object diagrams with an energy-based metric can be problematic even with a more sophisticated metric. Sometimes, other model-reduction methods should be used, for example, those that operate on the equation system of the model. These methods also represent a part of our research activities. As the focus of the paper is on the reduction of object diagrams, the reduction at the equation level will be only briefly presented in the next section.

6. Realization-preserving reduction at the equation level

Models in Modelica are usually implemented in several hierarchical levels. At the bottom of the hierarchy,

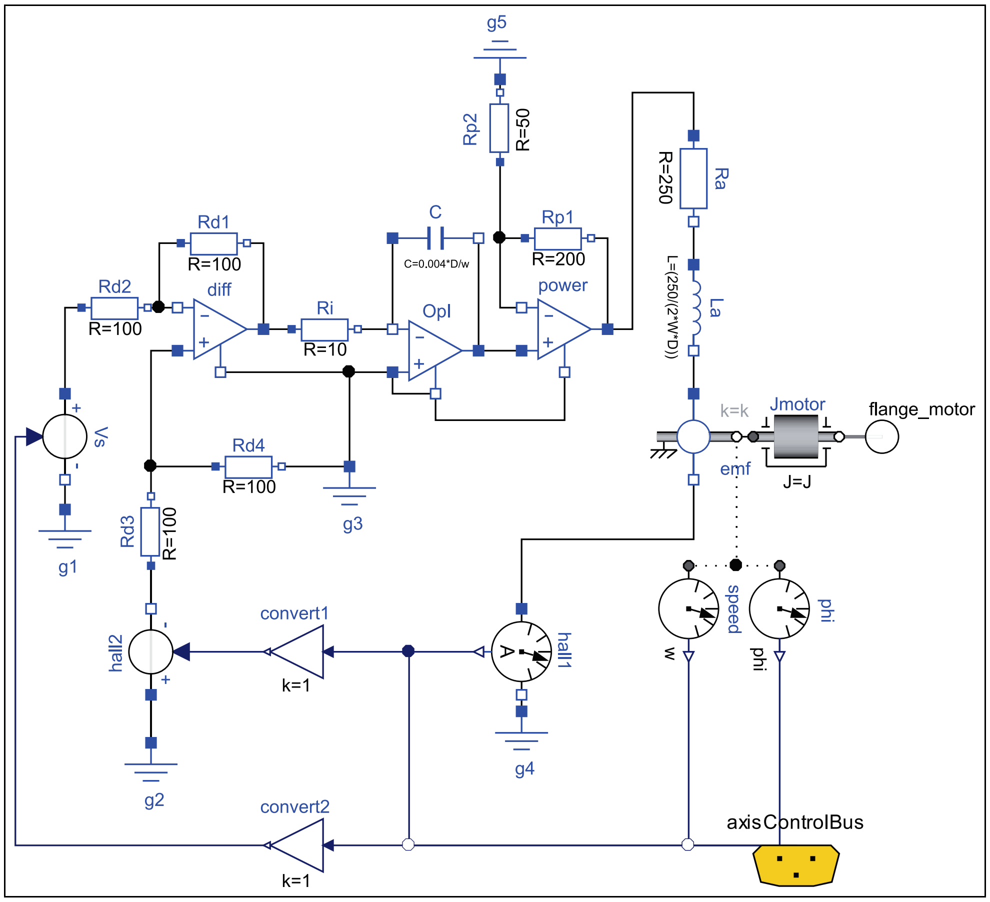

Figure 13. Schematic of a motor model.

Table 5. Ranking of the motor model components using the activity metric.

Element	Activity [J]	Relative [%]	Accumulated [%]
power	$5.78 \cdot 10^3$	47.70	47.70
Rp1	$2.98 \cdot 10^3$	24.56	72.26
Ra	$1.65 \cdot 10^3$	13.66	85.92
Rp2	$7.44 \cdot 10^2$	6.14	92.06
Jmotor	$5.24 \cdot 10^2$	4.33	96.38
flange_motor	$4.36 \cdot 10^2$	3.60	99.98
La	1.85	0.02	100.00
OpI	0.08	0.00	100.00
C	0.08	0.00	100.00
Vs	0.03	0.00	100.00
hall2	0.03	0.00	100.00
Rd1	0.02	0.00	100.00
Rd2	0.02	0.00	100.00
...	...	...	...

differential-algebraic equations (DAEs) are used for the component description, while for higher levels model it is described graphically with object diagrams. Bearing in mind the difficulties described in previous sections when the reduction was implemented on object diagrams, the possibilities for reduction at the equation level must also be considered.

The most general representation of a dynamical system is the DAE is as follows:

$$\boldsymbol{F}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{y}, t) = 0 \tag{16}$$

where $\boldsymbol{x}$ is the vector of system states, $\dot{\boldsymbol{x}}$ is the vector of corresponding derivatives, $\boldsymbol{y}$ is the vector of algebraic variables, and t is the independant variable. This basic description is usually composed of several (n) terms:

$$t_1(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{y}) + \ldots + t_j(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{y}) + \ldots + t_n(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{y}) = 0 \tag{17}$$

$t_j(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{y})$ designates expressions, that is, terms, of the variables $\boldsymbol{x}$, $\dot{\boldsymbol{x}}$, and $\boldsymbol{y}$. An elementary reduction acting on term $t_j(\cdot)$ in Equation (17) could thus delete it or replace it with a simple expression, that is, a constant value or a linearized expression. The procedure is based on terms ranking, which shows the importance of a particular term to the model trajectories.

There are already tools available commercially[26] for the reduction and simplification of a general set of DAEs. The method combines various algebraic manipulations and approximation techniques, for example, deletion of a single term in an equation, replacement of a term with a constant, deletion of a variable or its derivative, etc.

We developed a prototype toolbox[19,27] for the Open Modelica environment in which we used an approach originally developed for the model reduction of nonlinear DAEs, entitled *Behavioural model generation (BModGen)*.[26,28,29] A decision about which elemental reduction operation should be performed depends entirely on an estimation of how much the considered operation will change the behavior of the model in the specific simulation experiment(s). 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.[29]

7. Conclusions

An energy-based model-reduction technique and its implementation in Open Modelica was presented. To the best of our knowledge there are no implementations for Modelica, one of today's most commonly used object-oriented multidomain modeling languages, which would enable realization-preserving model reduction.

The presented approach does not require any special formulation of the model and thus can be applied to many already-existing models. However, the price we pay for this generality is a loss of performance and an increased complexity of the algorithm, because less a priori knowledge about the model is available. So, either an extensive set of reduction rules must be provided or only the most basic reduction operations can be performed, for example, elimination of the components associated with low energy.

Whereas a component ranking can be implemented quite effectively with a minimum required input from the user by the approach described, a proper removal of the components is more difficult. Although the appropriate rule for the removal of a component can be derived automatically from its constitutive law, there must also be side-effects taken into account that may not be resolvable locally, for example, the initial values of the other components might need to be modified in order to preserve the consistency of the initial value problem, as demonstrated in the examples.

Nevertheless, even if the model-reduction tool cannot reduce the model completely automatically, it may still offer many useful insights and hints to the user. We are also aware of the fact that we refer to problems of complex dynamical models but that the examples are rather simple. Namely, our activity can be treated as an initial attempt. It works on smaller parts in a semi automatic way, so that usually some manual operations are needed. We are not sure if the approaches can be automatically used for real complex models in the future.

Funding

This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

ORCID iD

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

References

1. Zupančič B and Sodja A. Computer-aided physical multidomain modelling: some experiences from education and industrial applications. *Simulat Model Pract Theor* 2013; 33: 45–67.
2. Ersal T, Fathy HK, Rideout DG, et al. A review of proper modeling techniques. *J Dynam Syst Measur Contr* 2008; 130: 061008.
3. Ugryumova MV. *Applications of model order reduction for IC Modeling*. Dissertation, Eindhoven University of Technology, 2011.
4. Moore BC. Principal component analysis in linear systems: controllability, observability, and model reduction. *IEEE Trans Autom Contr* 1981; AC-26: 17–32.
5. Sheehan BN. TICER: realizable reduction of extracted RC circuits. In: *IEEE/ACM international conference on computer-aided design*, digest of technical papers, 1999, pp.200–203.[AQ: 3]
6. Louca LS, Rideout DG, Stein JL, et al. Generating proper dynamic models for truck mobility and handling. *Int J Heavy Vehicle Syst* 2003; 11: 209–236.
7. Modelica specification. Version 3.3, http://www.modelica.org/documents/ModelicaSpec33.pdf (2012, accessed 29 October 2018).
8. Fritzson P. *Principles of object oriented modeling and simulation with Modelica 3.3*. Wiley-IEEE Press, 2014.[AQ: 4]
9. Chang SY, Carlson CR and Gerdes JC. A Lyapunov function approach to energy based model reduction. In: *proceedings of the ASME dynamic systems and control division (2001 IMECE)*, New York, 2001, pp.363–370.[AQ: 5]
10. Browne G, Krouglicof N and Rideout G. An energy-based proper model of an automotive fuel delivery system. In: *SAE 2010 world congress & exhibition technical papers*, Detroit, 2010, p.01–0421.[AQ: 6]
11. Rideout DG, Stein JL and Louca LS. System partitioning and improved bond graph model reduction using junction structure power flow. In: *proceedings of the international conference on bond graph modeling (ICBGM '04)*, New Orleans, 2005, pp.43–50.[AQ: 7]
12. Rosenberg RC and Ermer GE. A Bond graph visualization tool to improve engineering system design. *Syst Anal Model Simulat* 1995; 18–19: 173–178.
13. Rosenberg R and Zhou T. Power-based model insight. In: *proceedings of the ASME WAM symposium on automated modeling for design*, New York, 1988, pp.61–67.[AQ: 8]

14. Louca LS. *An energy-based model reduction methodology for automated modeling*. Dissertation, University of Michigan, 1998.
15. Ersal T. *Realization-preserving simplification and reduction of dynamic system models at the graph level*. Dissertation, University of Michigan, 2007.
16. Ye Y and Youcef-Youmi K. Model reduction in the physical domain. In: *proceedings of the American control conference*, San Diego, CA, 1999, pp.4486–4490.[AQ: 9]
17. Cellier FE and Nebot A. The Modelica bond graph library. In: *proceedings of 4th Modelica conference*, Hamburg, 2005, pp.57–65.[AQ: 10]
18. Sodja A and Zupančič B. On using model approximation techniques for better understanding of models implemented in Modelica. In: *proceedings of 8th Modelica conference*, Dresden, 2011, pp.697–703.[AQ: 11]
19. Sodja A. *Object-oriented modelling and simulation analysis of the automatically translated models*. Dissertation, Faculty of Electrical Engineering, University of Ljubljana, 2012.
20. Modelica standard library 3.1. User's guide, http://www.modelica.org/libraries (2010, accessed 29 October 2018).
21. Cellier FE and Greifeneder J. Modeling chemical reactions in Modelica by use of chemo-bonds. In: *proceedings of the 7th international Modelica conference*, Como, 2009, pp.142–150.[AQ: 12]
22. Open Source Modelica Consortium. OpenModelica, http://www.openmodelica.org (2012, accessed 29 October 2018).
23. Matko D, Karba R and Zupančič B. *Simulation and modelling of continuous systems: a case-study approach. Prentice-Hall International Series in Systems and Control Engineering*. Prentice Hall, 1992.[AQ: 13]
24. Cellier FE. *Continuous system modeling*. New York: Springer, 1991.
25. Zupančič B. Computer aided support for the temperature control in buildings. *Int J Simulat Proc Model* 2017; 12: 459–469.
26. Sommer R, Halfmann T and Broz J. Automated behavioral modeling and analytical model-order reduction by application of symbolic circuit analysis for multi-physical systems. *Simulat Model Pract Theor* 2008; 16: 1024–1039.
27. Sodja A and Zupančič B. Realisation-preserving model reduction of models in Modelica. In: *proceedings of the 7th Vienna conference on mathematical modelling (Mathmod 2012)*, Vienna, 2012, pp.322–328. [AQ: 14] [AQ: 15]

Author biographies

Anton Sodja received a PhD from the University of Ljubljana, Faculty of Electrical Engineering. Currently he works as a research and development (R&D) 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.