AUTOMATED BEHAVIORAL MODELING AND ANALYTICAL MODEL-ORDER REDUCTION BY APPLICATION OF SYMBOLIC CIRCUIT ANALYSIS FOR MULTI-PHYSICAL SYSTEMS

Ralf Sommer[1], Thomas Halfmann[2], Jochen Broz[2]

[1]Institute for Microelectronic and Mechatronic Systems gGmbH & Technical University of Ilmenau, Faculty of Electrical Engineering and Information Technology,
98693 Ilmenau, Ehrenbergstr. 27, Germany
[2]Fraunhofer ITWM, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany

Ralf.Sommer@imms.de (Ralf Sommer)

Abstract

The aim of symbolic analysis that has its origin in the design of analog circuits is the extraction of dominant system behavior by automated derivation of approximated symbolic formulas. Since exact symbolic analysis will yield exceptionally complex expressions even for rather small systems a class of symbolic approximation techniques have been developed that allow a reduction of the complexity of symbolic equations and their later solution by means of mixed symbolic and numerical strategies. Hence, it becomes possible to reduce the underlying nonlinear differential-algebraic systems of equations (DAE systems) of component-based networks and systems to a behavioral description of a predefined accuracy. It is a major advantage of the approach that the model simplification is performed by an automatic error control and that the simplified model is physically interpretable again. The contribution will give an overview of the symbolic tool Analog Insydes (www.analog-insydes.de), algorithms for extraction of dominant behavior of linear systems, e.g. formulas for poles and zeros as well as algorithms for generating behavioral models from nonlinear DAEs. Moreover, the underlying methodology has been extended to the application of analysis and modeling of gas-pipeline nets and mixed electrical and mechanical systems. For the latter a library was developed in cooperation with the Fraunhofer IIS/EAS for symbolic models of micro-mechanical elements that can be connected to networks, even together with electrical components.

Keywords: symbolic analysis, model reduction, behavioral modeling, multi-physical modeling.

Presenting Author's biography

Ralf Sommer received his diploma and Ph.D. in electrical engineering from Technical University of Braunschweig, Germany. In 1993 he was with the Technical University of Kaiserslautern and the Fraunhofer ITWM where he managed the developing of Analog Insydes. In 2000 he joined Infineon Technologies AG, Munich, Germany, where he was a group leader in the central CAD department with responsibility for analog design flow development and research projects. In 2006 he became professor at Technical University of Ilmenau as well as the scientific director of the Institute for Microelectronic and Mechatronic Systems gGmbH

1 General

The paper will give an application oriented introduction into symbolic analysis of analog electronic circuits and multi-physical systems using the EDA tool Analog Insydes. Starting with a brief review of the problems in current industrial analog circuit design an overview of the state of the art concerning the functionality of symbolic analysis methods will be given. A modeling methodology will be introduced where the netlist-based modeling language has been extended for the handling of multi-domain and vector-type through and across variables. With this approach, an automated setup of symbolic model equations in terms of a differential-algebraic system of equations starting from a netlist description is possible. This allows the application of DAE solvers for numerical simulation as well as the application of symbolic model reduction methods of multi-physical systems. Special attention will be devoted to the subject of symbolic approximation strategies in general but particularly to the approximation of linear and nonlinear equations resulting from electrical and multi-physical systems. The strategies have been successfully applied to analyze industrial analog circuits, e.g. extracting symbolic formulas for their dominant and critical poles and zeros as well as for the automatic generation of behavioral models from nonlinear dynamic equations of multi-physical systems.

2 Introduction

With the decreasing structural size going along with expanding complexity of technical systems there is an emerging demand for new design methods and modeling support. This becomes even more important because of the increasing heterogeneity of technical systems. In particular, for the design of mechatronical systems this leads to the following problem: There are established tools for the design of the mechanical or electrical parts like FEM, multi-body, or circuit simulators, but those are usually specialized on their physical domain. Therefore, the consideration of interactions between the mechanical and electrical components is extraordinary laborious. These interactions often have to be taken into account because the assembly of independently optimized sub-components usually does not lead to an optimal system. On the other side, considering coupling effects results in new challenges in many aspects, which lead from the need of designers competence in multiple disciplines (electrical and electronic as well as mechanical engineering, physics and mathematics) up to the high complexity of the mathematical models which demand the employment of adapted simulation tools.

Such software has to be capable of dealing with the multi-physical aspects of such systems. There are several suitable modeling languages like VHDL-AMS [1] or Modelica [2] and corresponding simulators like AdvanceMS™ or Dymola™ available. They allow for a modularized modeling of the complete system or parts of it with arbitrary accuracy. But the modeling process of heterogeneous systems is very time consuming and, moreover, the resulting mathematical models become very complex even for comparatively small systems, posing numerical problems with respect to robustness, efficiency, and stability.

Today the simulation of industrial-sized systems lies beyond the limits of this approach. In order to reduce the numerical effort, model reduction techniques become more important. In this paper, we present a modeling approach which is based on symbolic methods and can be adapted to multi-physical systems due to its general mathematical principle. This includes an automatic generation of behavioral models as well as model reduction for electrical as well as for mechatronical components and a combination of both. Such models allow due to their reduced complexity an interactive processing and a more efficient simulation of the overall system. This may even enable the application of optimization and control methods.

The basic principles of this modeling approach, i.e. the idea of model reduction, and the tool which has been used, are described in Chapters 3 and 4. The Chapters 5 and 6 give examples from the domains hydraulics and mechatronics, respectively. Finally, in Chapter 7 a summary is given.

3 Symbolic Modeling Approach

3.1 Symbolic Analysis

The symbolic modeling principle originates from the field of analog circuit design where the motivation has been to gain a deeper circuit understanding by interpretation of analytic formulas. In this context dedicated techniques for linear as well as for nonlinear applications have been developed [3]. Starting from a netlist description – the topological representation of an analog circuit, which includes information about the connecting graph of the circuit's components and corresponding device models and parameters – it is possible to formulate a mathematical equation system which in general is a nonlinear differential-algebraic equation system (DAE system). The equations consist of Kirchhoff's current and voltage laws as well as the circuit element characteristics given by the corresponding current-voltage relations. The equation system can be set up automatically using standard formulation techniques, e.g. Modified Nodal Analysis (MNA) or Sparse Tableau Analysis (STA). The decisive property of such an equation system is that it can be analytically parameterized in the system parameters, e.g. using the resistor value R_1 instead of the numerical value 10Ω, which motivates the term "symbolic". Due to the symbolic formulation, the equation system is valid not only for one dedicated

parameter set, but for a complete class of models with arbitrary parameter values. Using computer-algebra methods it is possible to analytically investigate the behavior of the corresponding system.

The above described symbolic methods are integrated in the software package Analog Insydes ([4], www.analog-insydes.de), which is an add-on to the computer-algebra system Mathematica [5]. The tool includes functionality for analysis, modeling, and optimization of linear and nonlinear circuits of industrial size. Analog Insydes is based on a hierarchical netlist description language that allows to automatically set up symbolic circuit equations. Besides standard electrical engineering analysis like AC, DC, and transient analysis as well as visualization methods, dedicated model reduction methods are available within Analog Insydes that will be explained in more detail in the next sections. Moreover, the tool has been integrated into industrial design environments and frameworks and, thus, includes interface functionality for exchanging data with commercial circuit simulators like Eldo™, PSpice™, Saber™, or Spectre™.

3.2 Symbolic Model Reduction

Practical application of symbolic analysis would have been rather limited without application of symbolic approximation techniques. Even for relatively small systems the symbolic solutions e.g. for transfer functions are of such a high complexity that an interpretation of the equations or the extraction of an in-depth system understanding gets impossible. For this reason it is necessary to incorporate methods which reduce the complexity of symbolic expressions, i.e. the equations must be simplified. Indeed these techniques hold the key in modern symbolic circuit analysis.

The concept “symbolic approximation” describes a whole class of mixed symbolic/numeric procedures for the simplification of symbolic expressions. These completely automated procedures are based on numerical evaluations and simulations to determine the approximation error. This is different from manual simplifications that are mainly based on qualitative considerations (e.g. $R_1 << R_2$). Automated symbolic approximation may yield compact formulae expressions fulfilling an error bound specified by the user.

In recent years, numerous symbolic approximation algorithms have been developed and implemented in symbolic circuit analysis programs. According to the stage in the circuit analysis process in which they are applied, these algorithms are categorized in the literature as Simplification Before Generation (SBG) [7][8][9][10]. Simplification During Generation (SDG) [11][12][13], and Simplification After Generation (SAG) techniques [14][15].

One of the central prerequisites of the symbolic analysis flow presented in the next section was the development and implementation of efficient symbolic approximation algorithms which impose no restrictions on the formulation of circuit equations, neither linear nor nonlinear, or the set of circuit elements that may be used.

Equation-based approximation procedures own all these requested properties since they are already applied on the level of circuit equations before the solution is determined (SBG). The philosophy behind equation-based approximation is to follow the methodology of a circuit designer or engineer who introduces his simplifications already when formulating equations. Thus, the complexity of the problem and the mathematical effort to solve or process the system is reduced substantially.

Since this paper intends to give an overview of methodologies and results, only the underlying principle of equation-based approximation is presented. Figure 1 shows a general flow chart of the algorithm.

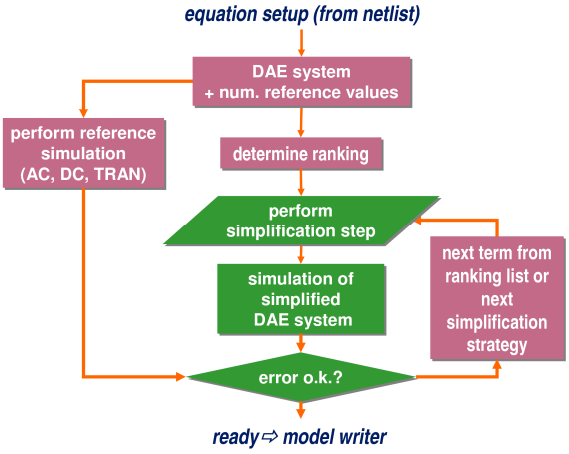

Figure 1: Flow of equation-based approximation

Equation-based approximation starts with the system of symbolic linear or nonlinear equations and a list of corresponding numerical reference values called design point.

Based on these numerical reference values the system of symbolic equations is evaluated and solved. This information is subsequently used to generate a term ranking. The term ranking mechanism plays a key role in the algorithm. Its task is to compute an order of all symbolic terms of the underlying equations such that the terms are sorted with respect to their influence on the solution. Ranking algorithms are an important subject of research since a large variety of different circuit characteristics may be of interest which has to be taken into account by the algorithm. For example in linear analysis magnitude, phase as well as pole and zero locations are of interest while in nonlinear analysis DC transfer, transient behavior, distortion, etc. are to be captured by the approximated system.

ISBN 978-3-901608-32-2

In the next step the output of the ranking algorithm is processed by the term removal mechanism which removes one or more terms from the system of symbolic equations. Now, this manipulated system with one or more terms deleted is passed to the error checking routine. Here the accumulated numerical error caused by the term removal is calculated and compared with the given error bound. If the error bound is exceeded the last term removal is undone and the algorithm terminates returning the approximated system. If the error bound is not exceeded the next terms from the term ranking list are selected and removed from the system followed by the error checking procedure as already described before. There are several extensions to the algorithm, e.g. symbolic simplification and elimination steps as well as more sophisticated term removal operations, e.g. block or cluster removals of elements and to use the error checking routine to control the term ranking [17].

After these preparation steps the actual approximation process in which iteratively different simplification strategies are used is carried out [16]:

Algebraic simplification: The complexity of the system of equations is reduced by exact algebraic transformations (e.g. variable elimination or decoupling of independent blocks).

Example:
The equations

$$x_1 = y_1 + y_2$$
$$x_2 = y_2 + y_3$$
$$x_3 = x_1 - x_2 + y_3$$

can be replaced by

$$x_3 = y_1$$

Branch simplification: Branches of piecewise-defined functions, which are not relevant during the simulation, are deleted from the equations.

Example:
The equation

$$y = x_1 + \begin{cases} x_1 \cdot x_2 & \text{if } x_2 > 1 \\ \frac{1}{2} x_1 \cdot (1 + x_2^{\,2}) & \text{else} \end{cases}$$

can be replaced by

$$y = x_1 + x_1 \cdot x_2 ,$$

if $x_2 > 1$ during the whole reference simulation.

Switch simplification: For models that are implemented suitably with respect to physical effects these different effects may be taken into account (switch parameter s with value 1) or neglected (switch parameter s with value 0) [18]:

Example:
The equation

$$M\ddot{x} = s_1 f(x) - s_2 \eta \dot{x}$$

can be replaced by

$$M\ddot{x} = f(x)$$

Here, s_1 and s_2 are switches for turning on and off certain physical effects.

Term substitution: Terms are replaced by the mean value obtained in the reference simulation.

Example:
The equation

$$x_3 = R \cdot (x_1 + f(x_2))$$

can be replaced by

$$x_3 = R \cdot (x_1 + 1.23)$$

Term deletion: Terms are removed from the system of equations.

Example:
The equation

$$x_3 = (R_1 + R_2) \cdot (x_1 + f(x_2))$$

can be replaced by

$$x_3 = R_1 \cdot x_1$$

The item "term" describes all symbolic expressions that appear as summands in equations, and that can consist of expressions themselves again.

The first two simplification methods do not influence the numeric solution of the DAE system; however, they can influence the numeric stability of the system of equations like the other (approximating) steps. For this reason dedicated methods have been developed [17] that monitor the numerical stability (or index) during the simplification process.

Switch simplification, term substitution, and term deletion are approximations and lead to errors or deviations from the original solution. These deviations are checked at each step by comparison with the reference solution. If the deviations lie within a user-given tolerance specification, the algorithm continues with the next simplification step. Otherwise the simplification step is undone and – if necessary – the cluster information is updated before the next simplification step is performed. This process is repeated, until no further simplification steps can be carried out with respect to the given error tolerances.

4 Modeling Methodology

4.1 Bottom-up Model Generation Flow

The model generation process depicted in Figure 2 is based on the symbolic analysis tool Analog Insydes. This powerful tool offers the functionality to automatically set up circuit equations from a circuit netlist and to use them as basis for generating a behavioral model. Symbolic device models

(corresponding to the simulator's device models) are used to make the strategy as accurate as the circuit simulation itself. The circuit equations are usually set up in an extended modified nodal analysis (MNA) for nonlinear equations. The resulting dynamic nonlinear equations contain the network equations in MNA (as used in most circuit simulators) as well as the nonlinear element relations resulting from symbolic device models. In general, one deals with DAE systems. An example for such a generated system of circuit equations is shown in Figure 3.

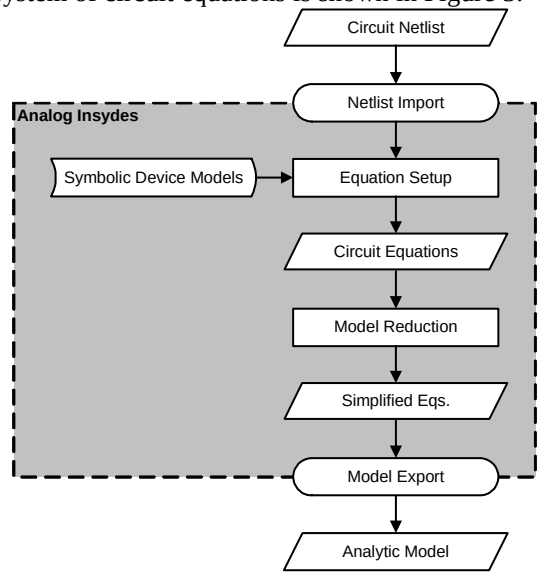

Figure 2: Proposed bottom-up modeling flow (Analog Insydes)

Model reduction methods can be applied to reduce the complexity of the equations by term reduction techniques [6]. The benefit of this symbolic approximation technique is to ensure a user-specified accuracy. Hence, this is one of very few methods that allow satisfying a predefined accuracy of the resulting model. Since the equations' complexity decreases while the resulting error increases with the degree of model reduction, it is up to the user to find a suitable trade-off between complexity and accuracy of the model. Experiments show that for reasonable error margins (5-10%) the complexity can be reduced very efficiently yielding a performance improvement of about factor 10 to 100.

Finally, the behavioral model can be generated from the DAE by using Analog Insydes' model export function [20]. It generates several AHDLs (VHDL-AMS, Verilog-A, etc.) and hence supports the creation of models for the most commonly used behavioral simulators.

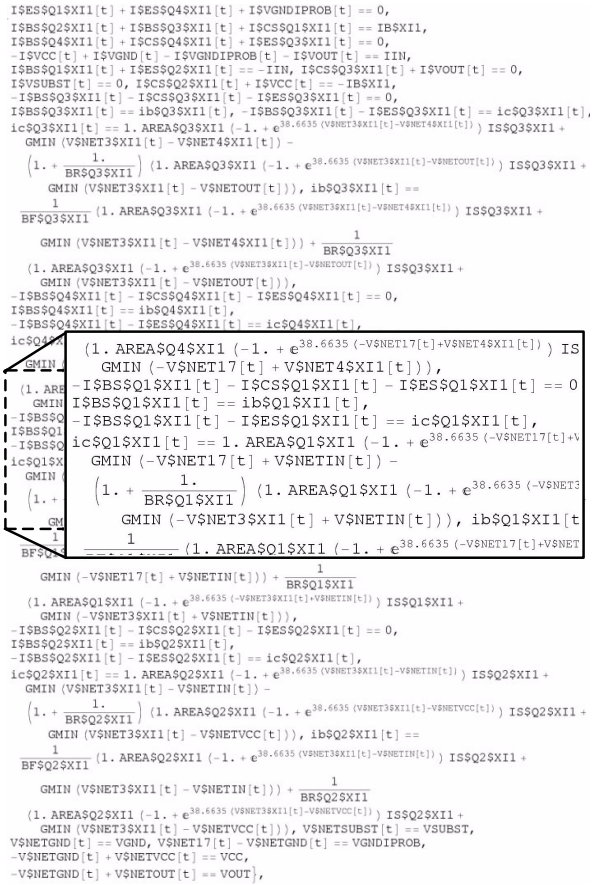

Figure 3: Example of an automatically generated DAE system

4.2 Assignment of the Methodology

Because of the general mathematical approach of the model processing procedures the methods originally developed for analog circuits can be transferred to other application fields. Analog Insydes' capabilities for automatic generation of symbolic equations and their approximation can be applied to any system that is described by generalized Kirchhoff equations by exploiting the analogies to electronic circuits (see Table 1). To allow this, an implementation for the connection of through and across variables of system components was necessary. In this way automated equation setup and symbolic approximation methods could be extended for the analysis of gas-pipeline nets (hydraulics) and mechatronic systems.

domain	**across variables**	**through variables**
electronics	voltage *V*	current *I*
mechanics	displacement (u_x, u_y, u_z)	force (F_x, F_y, F_z)
	rotation (φ_x, φ_y, φ_z)	torque (M_x, M_y, M_z)
hydraulics	pressure P	mass flow Q

Table 1: Across and through variables

5 Gas-pipeline Nets

While voltages and currents are used as system variables for analog electrical circuits gas pipelines are characterized by pressure P and mass flow *Q*. To model the static behavior of a gas pipeline the following relation is used:

$$P_a^2 - P_b^2 = \frac{16 z R T L \lambda}{\pi^2 D^5 M} |Q| Q$$

P_a and P_b mark the pressure at node *a* and *b*, *Q* the mass flow in the pipeline, and λ the Darcy-Weisbach friction factor that can be determined by

$$\lambda = \sqrt{2 \log_{10}\left(\frac{3.7 D}{\varepsilon}\right)}$$

Further parameters can be found in Table 2. In a similar way other system components like compressors, pressure and flow regulators, consumers, and sources can be described [19].

M	Mol mass
R	gas constant
T	temperature
ε	roughness of the pipeline wall
z	compressibility factor
λ	Darcy-Weisbach friction factor
D	diameter
L	length

Table 2: Pipeline model parameters

Figure 4 shows the high pressure gas pipeline net of Belgium which contains 6 sources and 8 consumers, a network of 24 pipelines, and 2 compressors. The system of equations derived from it which shall serve to calculate pressure in different junction nodes and network parts as function of sources and consumers consists of 109 equations with altogether 97 parameters. The result after the application of the model reduction techniques from Section 3.2 for 3 or 7 bar of error tolerance are summarized in Table 3. The mean pressure is about 50 bar. It should be noticed how larger model reductions can be obtained by increasing the error tolerance.

	full model	**3 bar**	**7 bar**
equations	109	14	10
parameters	97	63	47

Table 3: Dimension of system of equations of full and reduced models

In this application case it especially turned out that the resulting simplified system of equations can be interpreted again in the form of component equations and can be mapped to a reduced network (consisting of only 10 pipelines). The nodes of the removed pipelines collapse to a new node ("pressure zones", Figure 4).

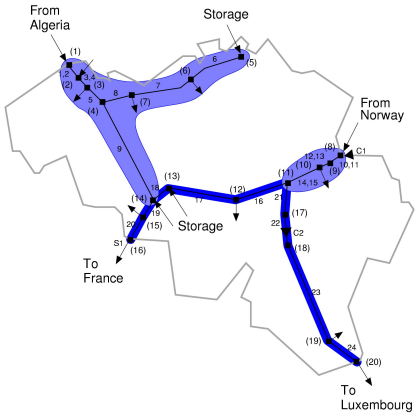

Figure 4: Reduced model (3 bar) of the Belgian high pressure gas pipeline net. Ice blue: pressure zones, deep blue: remaining pipelines

6 Mechatronics

6.1 Basics

To be able to treat mechanical systems with the introduced modeling approach, these are taken to pieces into *finite elements* which are firmly connected to each other in the network nodes. Displacements and rotations as a result of force and torque influences are regarded as small. Forces, torques, displacements, and rotations are used instead of currents and voltages here.

In cooperation with the Fraunhofer IIS/EAS a library which contains linear and nonlinear beam elements as well as different force and displacements sources was developed providing symbolic models of micro mechanical elements.

These element models describe the dynamic behaviour of the components in the form of dynamic equations. In the linear case these have the form

$$\mathbf{F} = -\left(\mathbf{G}^{-1}\mathbf{M}\mathbf{G}\,\ddot{\mathbf{u}} + \mathbf{G}^{-1}\mathbf{D}\mathbf{G}\,\dot{\mathbf{u}} + \mathbf{G}^{-1}\mathbf{K}\mathbf{G}\,\mathbf{u}\right)$$

M is the mass matrix, **D** the damping matrix, and **K** the stiffness matrix of the finite elements. Furthermore,

$$\mathbf{F} = (f_x^a, f_x^a, f_x^a, m_x^a, m_x^a, m_x^a, f_x^b, f_x^b, f_x^b, m_x^b, m_x^b, m_x^b, \cdots)$$

is the vector of the through variables (forces and torques) in the pins a, b, …, and

$$\mathbf{u} = (u_x^a, u_x^a, u_x^a, \varphi_x^a, \varphi_x^a, \varphi_x^a, u_x^b, u_x^b, u_x^b, \varphi_x^b, \varphi_x^b, \varphi_x^b, \cdots)$$

is the corresponding vector of the across quantities (displacements and torsions). With **G** the orthogonal transformation matrix which transforms the node variables of the global reference system into a local reference system for which the matrices **M**, **D** and **G** are formulated. Due to the vector quality an element with two pins (e.g. a beam element) is therefore described by a system of 12 equations.

The condition of the continuity of the displacements in the nodes corresponds to Kirchhoff's voltage law and the principle of d'Alembert corresponds to Kirchhoff's current equations.

6.2 Behavioral Device Models

Due to its origin Analog Insydes comes with a pre-defined device model library for analog electronic components only. For the modeling of mechatronical systems a device model library containing corresponding mechanical components is required. All device model implementations make use of the standard Analog Insydes modeling language in terms of a behavioral model description. This approach allows for modeling nearly arbitrary element characteristics by directly specifying the corresponding device equations which in general may be a nonlinear DAE system. The Analog Insydes model definition is based on a port branch concept which is illustrated in Figure 5 considering a nonlinear junction diode of the electronic domain.

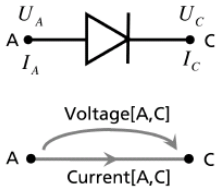

Figure 5: Diode quantities and branch definition

The diode has the two terminals anode and cathode denoted by the identifiers *A* and *C*, respectively. The current-voltage relation is given by the following device equation for the branch voltage V_D and the branch current I_D

$$I_D = I_S \cdot \left(e^{V_D / V_T} - 1 \right)$$

I_S and V_T denote the saturation current and the thermal voltage, respectively. The ports of different components are interconnected at network nodes. Behavioral models are defined in terms of port branches, where each unique pair of port identifiers `(port1, port2)` introduces a port branch between the model ports `port1` and `port2` with a positive reference direction from `port1` to `port2`. The associated port variables in the behavioral model equations can be referred by means of special keywords, which are `Voltage[port1, port2]` and `Current[port1, port2]` for the electronic domain. For the diode example the Analog Insydes format for the device equation reads as

$$\texttt{Current[A,C]} = I_S \cdot \left(e^{\frac{\texttt{Voltage[A,C]}}{V_t}} - 1 \right)$$

Note that an alternative concept to the port branch concept is the currents into the ports and the port voltages approach. But this method is not well suited for other analysis methods than MNA. Due to the fact that *Analog Insydes* supports different analysis methods the port branch concept has been implemented.

If this concept is transferred to mechanical components, then a simple component with two ports has already six accompanying model branches (e.g. see beam element in Figure 6).

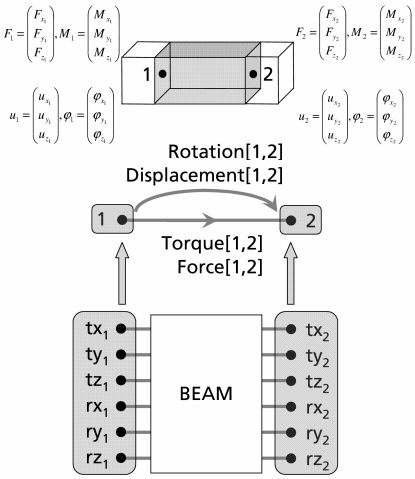

Figure 6: Beam quantities and branch definition

Therefore, the so far scalar-type Analog Insydes ports have been extended to the more general case of vector-type ports. Within a netlist description this is achieved by adding a `NetlistAttributes` section. This new language object is valid for the whole netlist object within it has been defined. One simply has to specify the dimension of a corresponding netlist node using the `NodeDimensions` keyword. The syntax is as follows:

```
NetlistAttributes[
  NodeDimensions -> {
    <node1> -> 6,
    ...,
    Default -> 1
  },
  NodePositions -> {
    <node1> -> {<x1>, <y1>, <z1>},
    ...
  }
]
```

where `<node1>` denotes the name of the mechanical node and `{<x1>, <y1>, <z1>}` is the corresponding numerical coordinate vector, respectively. The coordinate information specified by the `NodePositions` keyword is useful for automatically computing the geometrical parameters of the mechanical component. Note that with the `Default -> 1` setting one can simply define all remaining nodes in the netlist description as being scalar type without stating them explicitly which is very useful when having multi-physics applications. This new language construct is used in the following example.

6.3 Application Example

As an example for a multi-physical system we consider an acceleration sensor [13] consisting of mechanical and electrical parts (see Figure 7). The sensor consists of three parallel conducting plates which form two serial capacities C_1 and C_2. The central plate can be moved from its balanced position (center if $R_A = R_B$) resulting in a Hook's force with constant *K*. In case of an acceleration, the central plate moves away from its central position resulting in changes of the capacities between the electrical connectors E_1/E_0 and E_2/E_0. This yields a potential drop V_{out} for the central plate with respect to the potential in the idle state.

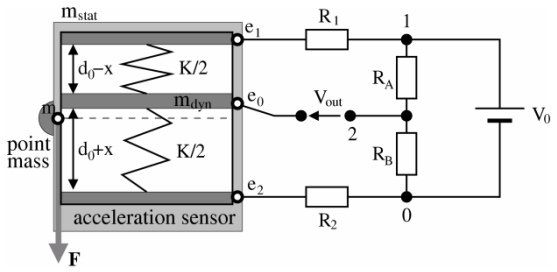

Figure 7: Acceleration sensor with simple circuitry

The acceleration sensor has one mechanical and three electrical ports (the center of mass and each plate). The mechanical port has the vector variables displacement **u** and force **F**. Besides the external accelerating force **F** there are internal forces acting on the system. The internal forces result from electrostatics, Hook's law, and damping:

$$F_{\text{int}} = \frac{Q_1^2 - Q_2^2}{2A\varepsilon_0} - Kx - D\dot{x}$$

Here, Q_1 and Q_2 are the charges of the plates E_1 and E_2, *A* is the plate area, ε_0 the dielectric constant, and *D* the damping constant. The force acts along the axial direction **e** and accelerates the central plate with mass m_{dyn}:

$$F_{dyn} = m_{dyn}\left(\ddot{x} + \mathbf{e}\cdot\ddot{\mathbf{u}}\right)$$

Here, *x* is the local displacement of the central plate from the idle position and **u** the global displacement of the acceleration sensor. Forces acting on the static mass m_{stat} are the external force **F** and internal force $\mathbf{F}_{\text{int}} \cdot \mathbf{e}$, yielding the equation of motion:

$$\mathbf{F} - F_{\text{int}}\mathbf{e} = m_{stat}\ddot{\mathbf{u}}$$

The charges Q_1 and Q_2 depend on the node voltages V_0, V_1 and V_2 at the electrical connection ports E_0, E_1 and E_2:

$$Q_1 = C_1\left(V_1 - V_0\right)$$
$$Q_2 = C_2\left(V_2 - V_0\right)$$

where the capacitances C_1 and C_2 depend on the plate distances (idle distance d_0)

$$C_1 = \varepsilon\frac{A}{d_0 - x}$$
$$C_2 = \varepsilon\frac{A}{d_0 + x}$$

The branch currents E_1 to E_0 and $E2$ to E_0 are given by

$$I_{i,0} = \dot{Q}_i \quad , \quad i \in \{1,2\}$$

Because we use six-dimensional port variables (displacements and rotation angles), we add trivial angular equations with moment of inertia θ and torque *M*

$$\ddot{\varphi}_i = \theta M_i \quad , \quad i \in \{1,2,3\}$$

Changes of the orientation of the sensor are not considered in this example.

The above equations are implemented as an *Analog Insydes* model called `AccelerationSensor`.

```
Netlist[
 {V0, {"1", "0"}, Symbolic → V0, Value → 1},
 {RA, {"1", "2"}, Symbolic → RA, Value → 1},
 {RB, {"2", "0"}, Symbolic → RB, Value → 1},
 {R1, {"1", "e1"}, Symbolic → R1, Value → 1.*^8},
 {R2, {"0", "e2"}, Symbolic → R2, Value → 1.*^8},
 {out, {"2", "e0"}, Value → 0, Type → OpenCircuit},

 {as, {"m" → "M", "e0" → "E0", "e1" → "E1", "e2" → "E2"},
  Model → "AccelerationSensor", DIRECTION → {0, 1., 0}},
 {mass, {"m" → "M"}, Model → "Mass", M → 0.1},
 {force, {"m" → "A"}, Model → "Source",
  FORCE → {0, If[t > 0.01, -1, 0], 0}, MOMENT → {0, 0, 0}},

 NetlistAttributes[
  NodeDimensions → {"m" → 6, Default → 1},
  NodePositions → {"m" → {0, 0, 0}}
 ]
]
```

Figure 8: Multi-physics netlist including acceleration sensor

```
V0 → 1                          DAMP$as → 0.0485322
RA → 1                          D0$as → 8 * 10^-6
RB → 1                          THETA$as → 0.0001
R1 → 10^8                       E1$as → 0
R2 → 10^8                       E2$as → 1
M$mass → 0.1                    E3$as → 0
THETA$mass → 1                  F1$force → 0
AREA$as → 6.25 * 10^-6          F2$force → If[t > 0.01, -1, 0]
EPS0$as → 8.854219 * 10^-12     F3$force → 0
K$as → 60.086                   M1$force → 0
MSTAT$as → 0.001927             M2$force → 0
MDYN$as → 9.8 * 10^-6           M3$force → 0
```

Figure 9: Numerical values for all system parameters (given in SI units)

Figure 8 shows its usage within the netlist description for the system of Figure 7. The acceleration sensor as is oriented in (0, 1, 0) direction. Its connections are specified by `<node> -> <port>` mappings. The electrical ports `E0`, `E1` and `E2` are connected at nodes `e0`, `e1` and `e2` with the circuit. The mechanical port `M` is connected at node `m` to the point mass and the accelerating force. At this node a time-dependent force acts towards (0,-1, 0), accelerating the system starting at *t = 10ms*. The corresponding numerical values for all model parameters are listed in Figure 9.

All component parameters are instantiated by the postfix `$<comp>` where `<comp>` specifies the corresponding component name as given in the netlist. For all values which are not set within the netlist, default values are used instead.

This notation is also used for the DAE system shown in Figure 11, which has been generated automatically from the netlist description. It consists of 39 equations for 39 variables, which are named using a similar convention: e.g. the local displacement *x* of the central plate of as is named `x$as[t]` and the third component of the displacement in node m is denoted by `u$m$3[t]`. Additionally, there are 16 initial conditions for the 14 mechanical variables (location and rotation, velocity and angular speed, displacement and velocity of the central plate) as well as the charges on plates E_1 and E_2.

For performing the approximation as described in Section 3.2, V_{out} has been chosen as output variable. The maximum approximation error has been set to 20 mV (20%). For these settings, the highlighted expressions have been identified to be not relevant for the dynamics of V_{out}. Furthermore, automated exact algebraic manipulation finally leads to the DAE system shown in Figure 12. The system has been reduced to a set of five equations only. Note, that the reduction to the y direction (only displacement variable `u$m$2` is left) and removal of all rotational degrees of freedom has been done completely automatic. Additional approximations, e.g. removal of `MDYN$as x$as´´[t]` in the dynamics of the central plate, have been applied automatically. Finally, Figure 10 illustrates a comparison of V_{out} for the original and simplified DAE system. Starting at *t = 10 ms* the force accelerates the point mass. In the reduced model, the acceleration of the system only depends on the point mass (`M$mass`), neglecting the mass of the sensor (`MSTAT$as+MDYN$as`) which is less than two percent of the point mass. This finally results in the slightly increased absolute value for the stationary voltage V_{out}. The difference in the dynamic behavior for *t* near 10 *ms* is mainly due to neglecting the inertia of the central plate.

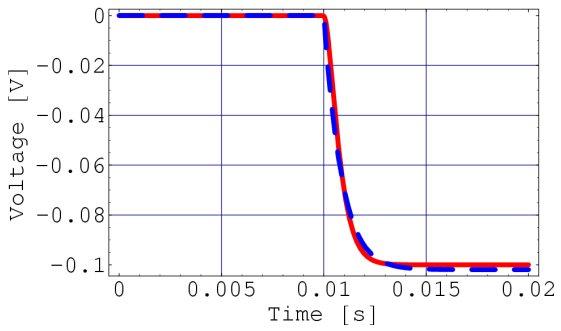

Figure 10: Comparison of original (solid) and simplified model (dashed)

$$i\$V0[t] + \frac{v\$1[t]}{R1} + \frac{v\$1[t]}{RA} - \frac{v\$2[t]}{RA} - \frac{v\$e1[t]}{R1} == 0$$
$$-\frac{v\$1[t]}{RA} + \frac{v\$2[t]}{RA} + \frac{v\$2[t]}{RB} == 0$$
$$-i\$E1E0\$as[t] - i\$E2E0\$as[t] == 0$$
$$i\$E1E0\$as[t] - \frac{v\$1[t]}{R1} + \frac{v\$e1[t]}{R1} == 0$$
$$i\$E2E0\$as[t] + \frac{v\$e2[t]}{R2} == 0$$
$$f\$M\$10\$as[t] + f\$M\$10\$mass[t] + f\$A\$10\$force[t] == 0$$
$$f\$A\$20\$force[t] + f\$M\$20\$mass[t] + f\$M\$20\$as[t] == 0$$
$$f\$M\$30\$mass[t] + f\$A\$30\$force[t] + f\$M\$30\$as[t] == 0$$
$$f\$M\$40\$mass[t] + f\$A\$40\$force[t] + f\$M\$40\$as[t] == 0$$
$$f\$M\$50\$as[t] + f\$A\$50\$force[t] + f\$M\$50\$mass[t] == 0$$
$$f\$M\$60\$as[t] + f\$M\$60\$mass[t] + f\$A\$60\$force[t] == 0$$
$$-V0 + v\$1[t] == 0$$
$$-v\$2[t] + v\$e0[t] + v\$out[t] == 0$$
$$-f\$M\$10\$mass[t] - M\$mass\ u\$m\$1''[t] == 0$$
$$-f\$M\$20\$mass[t] - M\$mass\ u\$m\$2''[t] == 0$$
$$-f\$M\$30\$mass[t] - M\$mass\ u\$m\$3''[t] == 0$$
$$-f\$M\$40\$mass[t] - THETA\$mass\ u\$m\$4''[t] == 0$$
$$-f\$M\$50\$mass[t] - THETA\$mass\ u\$m\$5''[t] == 0$$
$$-f\$M\$60\$mass[t] - THETA\$mass\ u\$m\$6''[t] == 0$$
$$c1\$as[t] - \frac{AREA\$as\ EPS0\$as}{D0\$as - x\$as[t]} == 0$$
$$c2\$as[t] - \frac{AREA\$as\ EPS0\$as}{D0\$as + x\$as[t]} == 0$$
$$q1\$as[t] + c1\$as[t]\ v\$e0[t] - c1\$as[t]\ v\$e1[t] == 0$$
$$q2\$as[t] + c2\$as[t]\ v\$e0[t] - c2\$as[t]\ v\$e2[t] == 0$$
$$0 == fint\$as[t] + \frac{q1\$as[t]^2}{2\ AREA\$as\ EPS0\$as} + \frac{q2\$as[t]^2}{2\ AREA\$as\ EPS0\$as} + K\$as\ x\$as[t] + DAMP\$as\ x\$as'[t]$$
$$i\$E1E0\$as[t] - q1\$as'[t] == 0$$
$$i\$E2E0\$as[t] - q2\$as'[t] == 0$$
$$0 == -fint\$as[t] + E1\$as\ MDYN\$as\ u\$m\$1''[t] + E3\$as\ MDYN\$as\ u\$m\$3''[t] + MDYN\$as\ x\$as''[t] + E2\$as\ MDYN\$as\ u\$m\$2''[t]$$
$$f\$M\$10\$as[t] + E1\$as\ fint\$as[t] + MSTAT\$as\ u\$m\$1''[t] == 0$$
$$f\$M\$20\$as[t] + E2\$as\ fint\$as[t] + MSTAT\$as\ u\$m\$2''[t] == 0$$
$$f\$M\$30\$as[t] + E3\$as\ fint\$as[t] + MSTAT\$as\ u\$m\$3''[t] == 0$$
$$f\$M\$40\$as[t] + THETA\$as\ u\$m\$4''[t] == 0$$
$$f\$M\$50\$as[t] + THETA\$as\ u\$m\$5''[t] == 0$$
$$f\$M\$60\$as[t] + THETA\$as\ u\$m\$6''[t] == 0$$
$$f\$A\$10\$force[t] - F1\$force == 0$$
$$-F2\$force + f\$A\$20\$force[t] == 0$$
$$f\$A\$30\$force[t] - F3\$force == 0$$
$$f\$A\$40\$force[t] - M1\$force == 0$$
$$f\$A\$50\$force[t] - M2\$force == 0$$
$$f\$A\$60\$force[t] - M3\$force == 0$$
$$u\$m\$1[0] == 0$$
$$u\$m\$2[0] == 0$$
$$u\$m\$3[0] == 0$$
$$u\$m\$4[0] == 0$$
$$u\$m\$5[0] == 0$$
$$u\$m\$6[0] == 0$$
$$u\$m\$1'[0] == 0$$
$$u\$m\$2'[0] == 0$$
$$u\$m\$3'[0] == 0$$
$$u\$m\$4'[0] == 0$$
$$u\$m\$5'[0] == 0$$
$$u\$m\$6'[0] == 0$$
$$q1\$as[0] == 0$$
$$q2\$as[0] == 0$$
$$x\$as[0] == 0$$
$$x\$as'[0] == 0$$

Figure 11: Automatically generated DAE system

$$F2\$force == M\$mass\ u\$m\$2''[t]$$
$$q1\$as[t] == \frac{AREA\$as\ EPS0\$as\ (RA\ V0 + (RA+RB)\ (v\$out[t] + R1\ q2\$as'[t]))}{(RA+RB)\ (D0\$as - x\$as[t])}$$
$$q2\$as[t] == \frac{AREA\$as\ EPS0\$as\ (-RB\ V0 + (RA+RB)\ (v\$out[t] - R2\ q2\$as'[t]))}{(RA+RB)\ (D0\$as + x\$as[t])}$$
$$q1\$as'[t] + q2\$as'[t] == 0$$
$$K\$as\ x\$as[t] + DAMP\$as\ x\$as'[t] + E2\$as\ MDYN\$as\ u\$m\$2''[t] == 0$$
$$u\$m\$2[0] == 0$$
$$u\$m\$2'[0] == 0$$
$$q1\$as[0] == 0$$
$$q2\$as[0] == 0$$
$$x\$as[0] == 0$$

Figure 12: Simplified DAE system

7 Summary

In this paper a new modeling approach for multi-physical systems is presented, which is based on symbolic methods. Such methods have already been successfully applied to electronical systems, and corresponding modeling tools are available. In order to transfer the methodology to multi-physical systems,

especially the multi-domain aspects, vector-valued variables and the need for adapted device models for basic system components have been developed and implemented. The capabilities of this new approach have been demonstrated on applications for the modeling and analysis of gas-pipeline nets and mixed electronical and mechanical systems.

8 Acknowledgements

The authors would like to thank the European Commission (FP5 research programme e_GASGRID) as well as the Fraunhofer society (MEF project 662410, "*Modellierungsunterstützung für den Entwurf komplexer mechatronischer Systeme mittels symbolischer Reduktionsverfahren*") for the financial support. We thank Christoph Clauß, Roland Martin and Peter Schwarz of the Fraunhofer IIS/EAS for the excellent cooperation within the MEF project.

9 References

[1] Christen, Bakalar, A hardware description language for analog and mixed signal applications, IEEE Trans. CAS-II, vol. 46, 1999, pp. 1263-1272.

[2] Fritzson, Principles of object-oriented mode-ling and simulation with Modelica 2.1, Wiley, Chichester 2004.

[3] Halfmann, Wichmann, Symbolic Methods in Industrial Analog Circuit Design, Scientific Computing in Electrical Engineering, Springer Verlag, 2006.

[4] Broz, Dreyer, Halfmann, Hennig, Thole, Wichmann, Analog Insydes 2.1 Manual, Fraunhofer ITWM, Kaiserslautern, 2005.

[5] Wolfram, The Mathematica Book, 5th edition, Wolfram Media Incorporated, 2003.

[6] Wichmann, Popp, Hartong, Hedrich, "On the Simplification of Nonlinear DAE Systems in Analog Circuit Design", *Proc. CASC'99*, Munich, Germany, 1999

[7] Sommer, Hennig, Dröge, Horneber, "Equation-Based Symbolic Approximation by Matrix Reduction with Quantitative Error Prediction", Alta Frequenza – Rivista di Elettronica, vol. 5, no. 6, Dec. 1993, pp. 29– 37

[8] Hsu, Sechen, "Fully Symbolic Analysis of Large Analog Integrated Circuits", in *Proc. CICC'94*, San Diego (USA), May 1994, pp. 457–460

[9] Hennig, Tweer, Sommer, "Enhanced Symbolic Matrix Approximation Techniques", in *Proc. SMACD'98*, Kaiserslautern (Germany), Oct. 1998, pp. 199–206

[10] Rodríguez-García, Guerra, Roca, Fernández, Rodríguez-Vázquez, "A New Simplification Before and During Generation Algorithm", in *Proc. SMACD'98*, Kaiserslautern (Germany), Oct. 1998, pp. 110-124

[11] Chang, MacKay, Wierzba, "Matrix Reduction and Numerical Approximation During Computation Techniques for Symbolic Analog Circuit Analysis", in *Proc. ISCAS'92*, San Diego (USA), May 1992, pp. 1153–1156

[12] Wambacq, Gielen, Sansen, "A Cancellation-Free Algorithm for the Symbolic Simulation of Large Analog Circuits", in *Proc. ISCAS'92*, San Diego (USA), May 1992, pp. 1157–1160

[13] Kole, *Algorithms for Symbolic Circuit Analysis Based on Determinant Calculations*, Ph. D. dissertation, University of Twente, Twente (The Netherlands), 1996

[14] Seda, Degrauwe, Fichtner, "A Symbolic Analysis Tool for Analog Circuit Design Automation", in *Proc. ICCAD'88*, pp. 488–491

[15] Gielen, Sansen, S*ymbolic Analysis for Automated Design of Analog Integrated Circuits*, Kluwer Academic Publishers, Boston (USA), 1991

[16] Halfmann, Wichmann, "Symbolic Methods in Industrial Analog Circuit Design", in Scientific Computing in Electrical Engineering, Edited by A.M. Anile, G. Alì, G. Mascali, Springer (2006).

[17] Wichmann, "Symbolische Reduktionsverfahren für nichtlineare DAE-Systeme", Shaker Verlag, Aachen, 2004.

[18] Näthke, Burkhay, Hedrich, Barke, "Hierarchical Automatic Behavioral Model Generation of Nonlinear Analog Circuits Based on Nonlinear Symbolic Techniques", in *Proc. DATE 2004*

[19] Mohring, Hoffmann, Halfmann, Zemitis, Basso, Lagoni, Automated Model Reduction of Complex Gas Pipeline Networks, PSIG 2004, Palm Springs, CA, Oct. 2004

[20] Wichmann, Thole, "Computer Aided Generation of Analytic Models for Nonlinear Function Blocks", 10th Workshop PATMOS, Sep. 2000

[21] Broz, Clauß, Halfmann, Lang, Martin, Schwarz, Automated Symbolic Model Reduction for Mechatronical Systems, CACSD 2006, Munich, Germany, Oct. 2006

[22] Platte, Sommer, Barke, "An Approach to Analyze and Improve the Simulation Efficiency of Complex Behavioral Models", in *Proc. of the IEEE Behavioral Modeling and Simulation Conference*, Sept. 2006

[23] Platte, Sommer, Broz, Dreyer, Halfmann, Barke, „Automatic Nonlinear Behavioral Model Generation using Sequential Equation Structures", in *Proc. SMACD'06*, Firence, Italy, Oct. 2006