Internalized State-Selection: Generation and Integration of Quasi-Linear Differential-Algebraic Equations

Christoph Höger[1] Andreas Steinbrecher[2]

[1]Institute of Software Engineering and Theoretical Computer Science, TU Berlin, Germany,
christoph.hoeger@tu-berlin.de
[2]Department of Mathematics, TU Berlin, Germany, anst@math.tu-berlin.de

Abstract

In modeling and simulation of dynamical processes frequently higher index differential-algebraic equations (DAEs) arise. Since an attempt to solve higher-index DAEs directly yields several numerical problems, a regularization in combination with a robust and efficient integration is required. QUALIDAES is a DAE solver designed to make explicit use of such a regularization. It allows for the solution of over-determined quasi-linear DAEs of the form $M(x,t)\dot{x} = f(x,t)$, $0 = g(x,t)$. Such DAEs arise naturally if a quasi-linear DAE is regularized by augmentation with the set of its (hidden) constraints. General DAEs can be brought into the quasi-linear form. To this end, equations can be transformed into the specific input format expected by QUALIDAES. This transformation can be implemented in a functional style and yields a non-trivial result. Additionally it provides an on-the-fly solution for the occurrence of higher-order derivatives.

Keywords: Differential-Algebraic Equations, Quasi-Linear, Modelica, Translation, Regularization, Solver, QUALIDAES

1 Introduction

MODELICA is a language for modeling of dynamical processes. In general, the model equations that describe the dynamical process consist of differential equations in combination with algebraic constraints, i.e., we have to deal with so-called *differential-algebraic equations* (DAEs).

The solutions of such systems have to satisfy the algebraic constraints, but, in general, not all constraints are stated in an explicit way. In particular, if the resulting system of DAEs is of higher index there exist so-called *hidden constraints* and the numerical treatment leads to instabilities, inconsistencies and possibly non-convergence of the numerical methods, see Brenan et al. (1996); Griepentrog and März (1986); Hairer and Wanner (1996); Kunkel and Mehrmann (2006). On the other hand, if a DAE does not contain any hidden constraint then its numerical treatment by use of implicit ordinary differential equation methods is not affected by instabilities. Furthermore, all constraints are preserved such that no drift-off effects arise in the numerical treatment.

Thus, a *regularization* or *remodeling* of the model equations resulting in an equivalent formulation with no hidden constraints is required to guarantee stable and robust numerical computations, see also Gear (1988); Hairer and Wanner (1996); Kunkel and Mehrmann (2006); Steinbrecher (2006).

The current state of the art in many modeling and simulation tools to deal with high index DAEs is to use some kind of analysis of the system to identify the constraints, to determine the index of the system, and to compute an index-reduced system model. Hereby, a crucial step is the so-called *state selection* that is required in order to introduce new algebraic variables (the so-called *dummy derivatives*) for the selected differential components of the DAE system in order to obtain a regular index-reduced formulation.

In this paper, we present in Section 2 a different regularization approach for the remodeling of dynamical systems that uses the hidden constraints to construct an over-determined system regularization that can be solved using a specially adapted numerical integrator implemented in the software package QUALIDAES (QUAsi LInear DAE Solver), see Section 3. This approach is developed for the numerical treatment of *quasi-linear DAEs* of the form

$$E(x,t)\dot{x} = k(x,t) \tag{1}$$

and has the great advantage that the problem of state selection can be moved into the numerical integrator such that it can be performed during the run-time of the simulation.

The software package QUALIDAES requires and exploits the quasi-linear structure of the model equations. If QUALIDAES is used in the MODELICA-framework then this requires a representation of the MODELICA model equation in quasi-linear form, as illustrated in Section 4.

2 Regularization Using Over-determined Formulations

In the following, we consider quasi-linear DAEs of the form (1) on the domain $\mathbb{I} = [t_0, t_f]$ with initial values $x(t_0) = x_0 \in \mathbb{R}^n$, where $E \in \mathscr{C}(\mathbb{R}^n \times \mathbb{I}, \mathbb{R}^{n,n})$ is called the *leading matrix* of the quasi-linear DAE and $k \in \mathscr{C}(\mathbb{R}^n \times \mathbb{I}, \mathbb{R}^n)$ its *right-hand side*. Furthermore, $x : \mathbb{I} \to \mathbb{R}^n$ represent the *unknown variables*. The DAE system (1) is assumed to be uniquely solvable and non-redundant. Furthermore, we assume that the rank of the leading matrix E is constant for all $(x,t) \in \mathbb{R}^n \times \mathbb{I}$ and that the rank of the partial derivatives of the (hidden) constraints with respect to x is constant for all consistent $(x,t) \in \mathbb{R}^n \times \mathbb{I}$.
Regularization approaches for high index DAEs like the Dummy Derivatives Approach from Mattsson and Söderlind (1993) or index reduction by Minimal Extension from Kunkel and Mehrmann (2004) consists of adding the hidden constraints to the model equations and the selection of certain differential components of the state x that can then be replaced by new algebraic variables in order to lower the index of the system and to obtain a new regular index-reduced system formulation. Hereby, a problem is that the selection of differential components can change during the numerical integration. Thus, if this state selection is performed outside the numerical integrator this often takes too long and is computational inefficient.

In the following, we will present a regularization of quasi-linear DAEs (1) of higher index, i.e., that contain hidden constraints. This regularization is based on an over-determined system formulation in order to overcome the difficulties in the numerical simulation.
Certain analysis tools, like Pantelides' algorithm (Pantelides (1988)), the structural analysis by Pryce (Pryce (2001)), the analysis via the strangeness-index concept (Kunkel and Mehrmann (2006)), the algebraic procedure proposed in Steinbrecher (2006), a combined structural-algebraic approach proposed in Scholz and Steinbrecher (2013), or other, gives us the required information about the hidden constraints in the system.

These information consists mainly of the order of differentiation of (parts of) the DAE to determine the hidden constraints by algebraic manipulations of the equations and their derivatives. The minimal order of differentiation of (parts of) the DAE required for the determination of a certain (hidden) constraint is called the *level of the (hidden) constraint*. Furthermore, the maximal level v_c of existing hidden constraints is called *maximal constraint level* of the DAE. See Steinbrecher (2006). Let us denote the set of all constraints including the hidden constraints by

$$0 = h(x,t). \tag{2}$$

Adding the hidden constraints to the quasi-linear DAE (1) leads to an over-determined DAE

$$\begin{aligned} E(x,t)\dot{x} &= k(x,t), && \text{(3a)}\\ 0 &= h(x,t) && \text{(3b)} \end{aligned}$$

consisting of a differential part (3a) and an algebraic part (3b). This over-determined formulation (3) then is equivalent to the original DAE (1) in the sense that both have the same solution set, i.e., for a given consistent initial value, the corresponding initial value problems have the same solution. Note that the leading matrix E not necessarily has to have full rank and the unknowns x are unchanged, i.e., a transformation of the state variables is not necessary and the number of unknowns is not increased (in contrast to the dummy derivative approach).

The over-determined formulation (3) has the advantage that all constraints explicitly are stated, Therefore, for (3) no hidden constraints exist. A further advantage of the over-determined formulation (3) is the fact that it is not necessary to apply analytic manipulations for the determination of a square, regular system of DAEs.
The proposed remodeling can be seen as regularization of the model equations. For more details on the regularization of quasi-linear DAEs we refer to Steinbrecher (2006).

Example 2.1 The Cartesian Pendulum: Let us consider the Cartesian pendulum, see Figure 1. We choose absolute coordinates p and q denoting the position of the mass m in the two dimensional space $\mathbb{R}^2$ for the description of the configuration of the pendulum. The equations of motion have the form

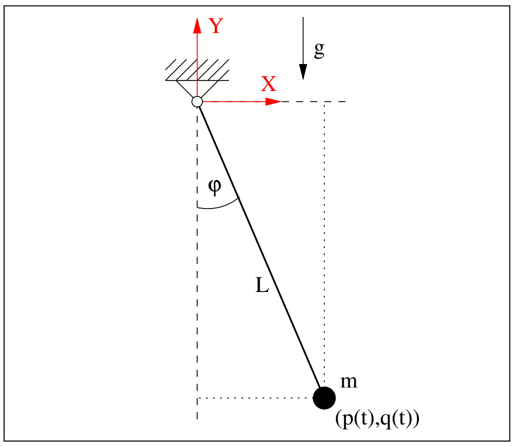

Figure 1. Topology of the Cartesian pendulum

$$\begin{aligned} \dot{p} &= v, && \text{(4a)}\\ \dot{q} &= w, && \text{(4b)}\\ m\dot{v} &= -2p\lambda, && \text{(4c)}\\ m\dot{w} &= -mg - 2q\lambda, && \text{(4d)}\\ 0 &= p^2 + q^2 - L^2, && \text{(4e)} \end{aligned}$$

where v and w denote the velocities of the mass point in X- and Y-direction while λ corresponds to the Lagrange multiplier. The constraint (4e) is of level 0 since no differentiation of the DAE is necessary to determine this constraint. The hidden constraint of level 1 obtained after only one total differentiation of (4e) with respect to t and replacing $\dot{x}$ and $\dot{y}$ using (4a), (4b) is given by

$$0 \;=\; 2pv + 2qw. \tag{4f}$$

Furthermore, from the second total derivative of (4e) with respect to t and replacing $\ddot{x}$, $\ddot{y}$ using the the first total derivative of (4a), (4b), and subsequent replacing of $\dot{x}$, $\dot{y}$, $\dot{v}$, and $\dot{w}$ using (4a)-(4d) we get the hidden constraint of level 2 as

$$\begin{aligned} 0 \;=\; & 2v^2 + 2w^2 + 2p(-2p\lambda)/m \\ & +2q(-mg - 2q\lambda)/m. \end{aligned} \tag{4g}$$

A further differentiation of the DAE does not lead to further constraints. Consequently the minimal order of differentiation of (parts of) the DAE to determine all (hidden) constraints is two. Therefore, the model equations for the Cartesian pendulum is a set of DAEs of maximal constraint level $\nu_c = 2$. With all hidden constraints the regularized DAE via over-determined formulation is given by equations (4a)-(4g). This regularized DAE consists of 7 equations for 5 unknowns $x = \begin{bmatrix} p & q & v & w & \lambda \end{bmatrix}^T$ and, due to its over-determinedness, is not solvable within the MODELICA-framework, e.g., OpenModelica, Dymola, MapleSim. $\triangleleft$

3 The Software Package QUALIDAES

In the following, we consider over-determined quasi-linear DAEs of the form

$$M(x,t)\dot{x} \;=\; f(x,t), \tag{5a}$$
$$0 \;=\; g(x,t) \tag{5b}$$

on the domain $\mathbb{I} = [t_0, t_f]$ with initial values $x(t_0) = x_0 \in \mathbb{R}^n$, where $M \in \mathscr{C}(\mathbb{R}^n \times \mathbb{I}, \mathbb{R}^{m_D,n})$ is called the *leading matrix* of the quasi-linear DAE and $f \in \mathscr{C}(\mathbb{R}^n \times \mathbb{I}, \mathbb{R}^{m_D})$ as well as $g \in \mathscr{C}(\mathbb{R}^n \times \mathbb{I}, \mathbb{R}^{m_C})$ form its *right-hand side*.

Such over-determined formulations of the form (5) have currently the disadvantage that within the common MODELICA-frameworks it is impossible to model and integrate over-determined systems. Therefore, a direct numerical integrator for possibly over-determined formulations in form (5) has been implemented in the software package QUALIDAES which requires and exploits the quasi-linear structure of the model equations.

The software package QUALIDAES is suited for general over-determined quasi-linear DAEs of the form (5) with the assumption that the constraints (5b) are neither contradictory nor redundant, i.e.,

$$\operatorname{rank}\left(\frac{\partial g}{\partial x}(x,t)\right) = m_C \tag{6}$$

for all consistent $(x,t) \in \mathbb{R}^n \times \mathbb{I}$. For a successful integration with QUALIDAES DAEs (5) with no hidden constraints are preferable. But often an integration of DAEs (5) containing hidden constraints of level 1 at most is successful. In case of no hidden constraints it holds

$$\operatorname{rank}\left(\begin{bmatrix} M(x,t) \\ \frac{\partial g}{\partial x}(x,t) \end{bmatrix}\right) = n \tag{7}$$

for all consistent $(x,t) \in \mathbb{R}^n \times \mathbb{I}$. Such over-determined formulations with no hidden constraints are obtained e.g. by application of the regularization approach described in the previous section.

The software package QUALIDAES is implemented in FORTRAN.

Certain features of QUALIDAES are to be emphasized which distinguish QUALIDAES from other solvers. Important is the fact that QUALIDAES respects all provided constraints. In particular, if no hidden constraints exist in (5), i.e., (7) holds, drift or instabilities are avoided during the numerical integration.

Interface for the model equations: The information of the model equations needed for the integration algorithm has to be provided in residual form, as following. The user or the calling subroutine has to provide the residual of the right hand side f for the differential part, the residual of the right hand side g for the constraint part, as well as the leading matrix M. All evaluated at a point (x,t). Furthermore, there exists a rough graphical user interface in MATLAB (Higham and Higham (2005)) suited for model equations provided in MODELICA, see Altmeyer and Steinbrecher (2013).

Integration method: In QUALIDAES the 3-stage implicit Runge-Kutta Method Radau IIa of fixed order 5, see Hairer and Wanner (1996), as discretization of the overdetermined formulation is implemented.

As mentioned above, the code QUALIDAES offers the possibility to combine the discretization method with the regularization technique presented in the previous section. Therefore, the algorithm may use the over-determined regularization in form (5) as basis for the discretization. For more details on the discretization we refer to Steinbrecher (2006).

The discretization of the over-determined system (5) using the 3-stage Radau IIa method leads to an over-determined nonlinear stage equation of the form

$$0 = \begin{bmatrix} D(\xi_k) \\ C(\xi_k) \end{bmatrix} \quad \text{with} \quad \xi_k = \begin{bmatrix} X_{k1} \\ X_{k2} \\ X_{k3} \end{bmatrix} \tag{8}$$

for the determination of the three stages $X_{ki} \in \mathbb{R}^n$, $i = 1,2,3$ on the current integration interval $[t_k, t_{k+1}]$ with $t_{k+1} = t_k + \delta_k$. Here δ_k denotes the current step size. The stages $X_{ki} \in \mathbb{R}^n$, $i = 1,2,3$ approximate the solution at the points $t_{ki} = t_k + c_i\delta_k$. In (8) D represents the discretization of the differential part and C represents the discretization of the constraints to determine the next iterate x_{k+1} from ξ_k. Unfortunately, the nonlinear system

(8) is no longer solvable because of discretization and rounding errors. Therefore, it is only possible to find an approximation $\widetilde{\xi}_k$ which minimizes the residual r of size $3(m_D + m_C)$ of the discretized over-determined DAE in a certain sense with

$$r = \left[\begin{array}{c} r_D \\ r_C \end{array} \right] = \left[\begin{array}{c} D(\xi_k) \\ C(\xi_k) \end{array} \right].$$

In general, such an approximation yields a residual $r_C \neq 0$, which in turn leads to unfulfilled constraints, i.e., $C(\xi_k) \neq 0$ and, therefore, also $g(x_k, t_k) \neq 0$, not even within machine precision. This would lead to the typical difficulties in the numerical integration of higher index DAEs, i.e., instabilities, convergence problems, inconsistencies, or the solution drifts away from the original solution manifold.
In order to avoid these problems it is necessary to make sure that the constraints are always satisfied during numerical integration. This can be achieved if the nonlinear system (8) is treated separately such that $\widetilde{\xi}_k$ satisfies the lower part, i.e., the constraints, exactly or within a prescribed precision, while $\widetilde{\xi}_k$ yields a minimal residual in the upper part, i.e., in the differential part.
For solving (8) as described above, an adaption of a simplified Newton method is implemented in `QUALIDAES`. For more details on Newton methods we refer to Deuflhard (2004). In particular, a constant Newton iteration matrix is used for a certain number of Newton iteration steps inside the current integration step $[t_k, t_{k+1}]$. The usage of the simplified Newton method saves evaluation of Jacobians and decomposition of the Newton iteration matrix in every except the first Newton iteration step. Therefore, during the Newton iteration a linear system of the form

$$\left[\begin{array}{c} J_D \\ J_C \end{array} \right] \Delta^j = \left[\begin{array}{c} d(\xi_k^j) \\ c(\xi_k^j) \end{array} \right] \tag{9}$$

has to be solved in each Newton iteration step $j = 0, 1, \ldots$ to obtain the next iteration $\widetilde{\xi}_k^{j+1} = \widetilde{\xi}_k^j + \Delta^j$ in the Newton iteration. The upper part in (9) represents the differential part while the lower part represents the constraint part. The solving of the linear algebraic system (9) has to be done in an efficient but stable way. For that the code `QUALIDAES` decomposes the differential part and the algebraic part via different decomposition methods. The LU decomposition with full pivoting is used for the constraint part and the LU decomposition with partial pivoting is used for the differential part. While the first full pivoting detects the set of locally constrained state variables the second decomposition is faster and detects a minimal set of differential equations for the locally dynamic state variables. For a $J \in \mathbb{N}$ we accept $\widetilde{\xi}_k^J$ as the numerical solution of (8) if a certain stopping criteria of the Newton iteration is satisfied. Furthermore, from this $\widetilde{\xi}_k^J$ we determine the next iterate x_{k+1} as approximation of the solution at t_{k+1}.
In particular, this strategy leads to a (numerically) precise fulfillment of the constraints while solving the differential part in an "approximate sense". For more details see also Scholz and Steinbrecher (2014).

Further features of `QUALIDAES`: The numerical integration implemented in `QUALIDAES` uses a variable step size strategy. For that an adaptation of the error estimation and the step size control implemented in the code `RADAU5` is used in `QUALIDAES`.
Furthermore, `QUALIDAES` offers the possibility to check and (if necessary) to correct initial values. For that the user or the calling subroutine has to provide further initial conditions in addition to the provided constraints (5b).
If the model equations have solution invariants, e.g., energy conservation or mass conservation, then it is often desirable to preserve these solution invariants explicitly because in general the numerical solution of the model equations does not satisfy the solution invariants. `QUALIDAES` is able to preserve solution invariants if they are provided by the user as additional equations in the constraints (5b).
Furthermore, `QUALIDAES` offers the possibility to determine a continuous output. This is helpful for example for an event detection or a visualization in the post processing.
If `QUALIDAES` is used in the MODELICA-framework then this requires a representation of the MODELICA model equation in quasi-linear form. For the most real applications, the model equations arise naturally in quasi-linear form. But unfortunately, in the MODELICA-framework this quasi-linear structure is not obviously reflected. Therefore, it is necessary to develop strategies to represent MODELICA model equations in quasi-linear form or to reformulate, e.g., by extension into this structure, as illustrated in the next section.

4 Quasi-Linear Model Equations

As already mentioned, MODELICA does not support quasi-linear equations directly, but allows the user to write arbitrary expressions to describe the dynamic behavior of the model. It is the responsibility of the underlying interpreter to transform the expressions into an equivalent suitable form (or, arguably, report an error if no such transformation can be found). Therefore, to use MODELICA as the model language for `QUALIDAES`, we have to provide said transformation.

In the following section we will resort to the following style of notation:

A language will be defined in a simple BNF-form: Nonterminals are expressed with the same small letters as meta-variables of the corresponding syntactic sort (e.g. we will use e to denote both the set of terms and a variable from that set). Productions are defined by ::= and alternatives are distinguished via $|$. The context will allow for easy distinction between both uses. Multiple variables of the same syntactic sort are introduced before they are used.

We will introduce the signature of functions in a blackboard style using $\times$ for Cartesian products and $\rightarrow$ to distinguish domain and co-domain. Partial functions will be introduced using the same style but with a $\hookrightarrow$. Additionally, we consider partial functions as sets of tuples that can be augmented using the $\mapsto$ operator (i.e. $p \cup \{1 \mapsto 2\}$ is the partial function p augmented by mapping 1 to 2. The domain of a partial function p is written $\mathrm{dom}(p)$.

Functions over elements of a language will be defined using a freely adapted denotational style: $[\![e]\!]_X$ denotes the function X applied to e. All these (recursive) functions are defined using pattern matching on their arguments: $[\![e_1 + e_2]\!]_X$ means the application of X to terms formed by the addition of two (possible distinct) terms. Meta-variables are bound in the patterns or corresponding where-clauses.

4.1 Input Language

As input we consider a small excerpt from the MODELICA abstract syntax of terms:

$$\begin{array}{rclcl} e & ::= & e+e & | & e\times e \\ & | & u_i & | & \mathrm{DER}(u_i) \\ & | & \tau & | & c \end{array}$$

Terms e (alternative variables are d, c) consist of addition, multiplication, numbered unknowns (u_i) a MODELICA-style derivative-operator $\mathrm{DER}()$, the simulation time τ and constants $c \subseteq \mathbb{R}$.

4.2 Quasi-Linear Language

A quasi-linear equation ql (or $\hat{ql}, \tilde{ql}$) forms one row of the aforementioned $E(x,t)$. Without loss of generality, we assume that all ql are of the form $e_i(x,t)\dot{x} + k_i(x,t) = 0$. We also assume that all our system consists of n unknowns $u_1 \dots u_n = x$. Then, a quasi-linear equation can be represented as a tuple of a constant term e and a partial function γ (alternatively β, α), mapping derivatives to their respective coefficients:

$$\begin{array}{rl} & ql \subseteq \gamma \times e \\ \text{where} & \gamma : u \hookrightarrow e \end{array}$$

An ordered set $QL = \{ql_1 \dots ql_n\}$ ($\tilde{QL}$ when an alternative is needed) of n quasi-linear equations forms a *system*. Such a system is equivalent to the leading matrix E augmented with its right-hand-side k in the sense that each quasi-linear equation defines a row of $E|k$:

$$\begin{array}{rclcl} QL & = & \{ql_1 \dots ql_n\} & \triangleq & E|k \\ & & e_{ij}(\dot{x}, x, t) & \triangleq & c_{ij} \cdot \dot{x} \end{array}$$

where

$$ql_i = \langle \gamma_i, e_i \rangle$$

$$c_{ij} = \begin{cases} [\![\gamma_i(u_j), x, t]\!]_e & \text{when } u_j \in \mathrm{dom}(\gamma_i) \\ 0 & \text{otherwise} \end{cases}$$

$$k_i(x,t) = [\![e_i, x, t]\!]_e$$

In this definition, the helper function $[\![\dots]\!]_e$ is a straightforward interpretation of terms:

$$\begin{array}{rcl} [\![]\!]_e & : & e \times \mathbb{R}^n \times \mathbb{R} \hookrightarrow \mathbb{R} \\ [\![e_1 \times e_2, x, t]\!]_e & \triangleq & [\![e_1, x, t]\!]_e [\![e_2, x, t]\!]_e \\ [\![e_1 + e_2, x, t]\!]_e & \triangleq & [\![e_1, x, t]\!]_e + [\![e_2, x, t]\!]_e \\ [\![\tau, x, t]\!]_e & \triangleq & t \\ [\![u_i, x, t]\!]_e & \triangleq & x_i \\ [\![c, x, t]\!]_e & \triangleq & c \end{array}$$

Note, that the interpretation function is undefined for derivative-terms. However, this does not cause any problems in our application, as any derivatives will be removed by our transformation (the matrix E does not contain any derivatives).

4.3 Transformation

With the above definitions, the remaining problem is how to transform a general MODELICA-style equation into a quasi-linear form. Naturally, there is a trivial transformation that replaces all derivative with simple identities of the form $\mathrm{DER}(u_i) = u_j$. Since these identities are trivially quasi-linear, this does not violate the requirements for the output of the transformation. However, the resulting system would be unnecessary large and not leverage the structure of the system for efficient simulation. In fact, `QUALIDAES` would have to solve the whole nonlinear system as hidden constraints. While the result (if it can be computed) might be (numerically) exact, this is certainly not the best or even an acceptable strategy.

Instead, we are going to keep the amount of additional identities to a minimum. We capture the identities in a partial function ι (also: κ, λ):

$$\iota : u_i \hookrightarrow u_j$$

The transformation $[\![\dots]\!]_{qlt}$ itself is again defined in a denotational style using pattern-matching:

$$[\![]\!]_{qlt} : e \times \mathbb{N} \times \iota \rightarrow ql \times \mathbb{N} \times \iota$$

The simplest cases are the simulation time, constants and unknowns. In these cases, the result is the quasi-linear equation with an empty set of coefficients, while the size of the system and the identities remain unchanged:

$$\begin{aligned}
[\![\tau,n,\iota]\!]_{qlt} &\triangleq \langle\langle\emptyset,\tau\rangle,n,\iota\rangle\\
[\![c,n,\iota]\!]_{qlt} &\triangleq \langle\langle\emptyset,c\rangle,n,\iota\rangle\\
[\![u_i,n,\iota]\!]_{qlt} &\triangleq \langle\langle\emptyset,u_i\rangle,n,\iota\rangle
\end{aligned}$$

For the summation of quasi-linear equations, we need a way to sum their coefficients in a way that maintains the interpretation of the coefficients as products of derivatives with terms. Hence, the sum of two coefficients is either undefined for a given unknown (when both summands are undefined for the unknown), the result of looking up that unknown in one of the coefficients (when it is only defined in one of them, mimicking addition with zero) or the sum of both right-hand sides (when it is defined in both coefficients):

$$(\gamma \uplus \beta)(u_i) \triangleq \begin{cases}\gamma(u_i)+\beta(u_i) & \text{when } u_i \in \mathrm{dom}(\gamma)\cap\mathrm{dom}(\beta)\\ \gamma(u_i) & \text{when } u_i \in \mathrm{dom}(\gamma)\setminus\mathrm{dom}(\beta)\\ \beta(u_i) & \text{when } u_i \in \mathrm{dom}(\beta)\setminus\mathrm{dom}(\gamma)\\ \text{undefined} & \text{otherwise}\end{cases}$$

With this definition, the transformation of the addition-term is the simple in-order transformation of both summands, followed by the construction of the quasi-linear sum:

$$[\![e_1+e_2,n,\iota]\!]_{qlt} \triangleq \langle\gamma\uplus\beta,d+c,\rangle,l,\lambda\rangle$$
where
$$\begin{aligned}
\langle\langle\gamma,d\rangle,m,\kappa\rangle &= [\![e_1,n,\iota]\!]_{qlt}\\
\langle\langle\beta,c\rangle,l,\lambda\rangle &= [\![e_2,m,\kappa]\!]_{qlt}
\end{aligned}$$

In order to transform a derivative, we have to check, whether said derivative is already identified with an (artificial) variable. In such a case, the derivative is replaced with the corresponding unknown and put into the right-hand-side term. If the derivative is not yet identified with another unknown, it yields a new coefficient:

$$[\![\mathrm{DER}(u_i),n,\iota]\!]_{qlt} \triangleq \begin{cases}\langle\langle\emptyset,u_{\iota(i)}\rangle,n,\iota\rangle & \text{when } i\in\mathrm{dom}(\iota)\\ \langle\langle\{u_i\to 1\},0\rangle,n,\iota\rangle & \text{when } i\notin\mathrm{dom}(\iota)\end{cases}$$

The transformation of multiplication-terms requires another auxiliary function, $[\![\ldots]\!]_\times$:

$$[\![e_1\times e_2,n,\iota]\!]_{qlt} \triangleq [\![qlt,e_2,m,\kappa]\!]_\times$$
where
$$\langle qlt,e_2,m,\kappa\rangle = [\![e_1,n,\iota]\!]_{qlt}$$

$[\![\ldots]\!]_\times$ allows to multiply a quasi-linear equation directly with a term. Again, it is defined in a denotational style using pattern-matching:

$$[\![]\!]_\times : ql\times e\times\mathbb{N}\times\iota\to ql\times\mathbb{N}\times\iota$$

Multiplication with non-derivative terms is again straight-forward (with $\otimes$ being the multiplicative equivalent to $\uplus$ defined above).

$$\begin{aligned}
[\![\langle\gamma,e\rangle,\tau,n,\iota]\!]_\times &\triangleq \langle\langle\gamma\otimes\tau,e\times\tau\rangle,n,\iota\rangle\\
[\![\langle\gamma,e\rangle,c,n,\iota]\!]_\times &\triangleq \langle\langle\gamma\otimes c,e\times\tau\rangle,n,\iota\rangle\\
[\![\langle\gamma,e\rangle,u_i,n,\iota]\!]_\times &\triangleq \langle\langle\gamma\otimes u_i,e\times\tau\rangle,n,\iota\rangle
\end{aligned}$$

An addition-term can be multiplied with a quasi-linear equation by multiplying its summands and summing up the result:

$$[\![ql,e_1+e_2,n,\iota]\!]_\times \triangleq \langle\langle\gamma\uplus\beta,e+d\rangle,l,\lambda\rangle$$
where
$$\begin{aligned}
\langle\langle\gamma,e\rangle,m,\kappa\rangle &= [\![ql,e_1,n,\iota]\!]_\times\\
\langle\langle\beta,e\rangle,l,\lambda\rangle &= [\![ql,e_2,m,\kappa]\!]_\times
\end{aligned}$$

Multiplication with a derivative is uncomplicated, when there is no coefficient mapped to a derivative in the quasi-linear equation:

$$[\![\langle\emptyset,e\rangle,\mathrm{DER}(u_i),n,\iota]\!]_\times \triangleq \langle\langle\{u_i\to e\},0\rangle,n,\iota\rangle$$

If the derivative is identified with another unknown, multiplication is defined recursively by multiplication with that unknown. In the general case, however, the multiplication with a derivative requires the addition of a new identification:

$$[\![ql,\mathrm{DER}(u_i),n,\iota]\!]_\times \triangleq \begin{cases}[\![ql,\iota(u_i),n,\iota]\!]_\times & \text{when } u_i\in\mathrm{dom}(\iota)\\ [\![ql,u_m,m,\iota\cup\{u_i\to u_m\}]\!]_\times & \text{otherwise}\end{cases}$$
where $m = n+1$

The final case is the multiplication of multiplication-terms. In that case, we can simply resort to the distributive property of multiplication:

$$[\![ql,e_1\times e_2,n,\iota]\!]_\times \triangleq \langle\tilde{ql},l,\lambda\rangle$$
where
$$\begin{aligned}
\langle\hat{ql},m,\kappa\rangle &= [\![ql,e_1,n,\iota]\!]_\times\\
\langle\tilde{ql},l,\lambda\rangle &= [\![\hat{ql},e_2,m,\kappa]\!]_\times
\end{aligned}$$

This transformation obviously deals just with a tiny fraction of the syntactically valid MODELICA equations and it is also quite obvious (at least to the experienced developer), that a lot of work needs to be put into a full coverage. However, adding more operators or syntactic variants does not add anything more insight into the discussed principles. On the contrary, if we would add an operation like MODELICA's power-operator $\wedge$, we would have to expand our transformation with a corresponding $[\![]\!]_\wedge$ routine. It should be quite clear that just a few such additions would make the transformation process unreadable. Hence, we conjecture (but do not prove for practical reasons) that *all* MODELICA equations can, in principle, be transformed into an equivalent quasi-linear form.

4.4 Derivatives and Hidden Constraints

The regularization (or index-reduction) of a DAE requires the (arbitrary-order) differentiation of equations with respect to the independent variable. For a quasi-linear representation however, the coefficients may only be multiplied with first-order derivatives of the system's unknowns. To maintain that invariant, it is necessary to break down an arbitrary-order differentiation into several steps of first-order differentiation and transformation into quasi-linear form.

The (first-order) derivative of a quasi-linear equation is computed by $[\![\ldots]\!]_{\nabla ql}$. It takes a quasi-linear equation, system size and identities and yields a term (which may contain applications of the $\textsc{der}()$-operator) and a new system size m and identities κ:

$$\begin{aligned}
[\![]\!]_{\nabla ql} &: \quad ql \times \mathbb{N} \times \iota \rightarrow e \times \mathbb{N} \times \iota \\
[\![\langle \gamma, e\rangle, n, \iota]\!]_{\nabla ql} &\stackrel{\wedge}{=} \quad \langle d + [\![e]\!]_{\nabla e}, m, \kappa\rangle \\
\text{where} & \\
\langle d, m, \kappa\rangle &= [\![\gamma, n, \iota]\!]_{\nabla \gamma}
\end{aligned}$$

The total derivative of the set of coefficients $[\![\ldots]\!]_{\nabla\gamma}$ is the sum of the total derivative of every coefficient. A coefficient can be differentiated by interpreting it as the product of the derivative with a term. To avoid generating a higher-order derivative, the coefficient's derivative has to be identified with a new or existing variable:

$$\begin{aligned}
[\![]\!]_{\nabla \gamma} &: \quad \gamma \times \mathbb{N} \times \iota \rightarrow e \times \mathbb{N} \times \iota \\
[\![\emptyset, n, \iota]\!]_{\nabla \gamma} &\stackrel{\wedge}{=} \quad \langle 0, n, \iota\rangle \\
[\![\{u_i \mapsto e\} \cup \gamma, n, \iota]\!]_{\nabla \gamma} &\stackrel{\wedge}{=} \quad \langle c, m+1, \kappa \cup \{u_i \mapsto u_{m+1}\}\rangle \\
\text{where} & \\
c &= [\![e]\!]_{\nabla e} \times u_{m+1} + e \times \textsc{der}(u_{m+1}) + d \\
\langle d, m, \kappa\rangle &= [\![\gamma, n, \iota]\!]_{\nabla \gamma}
\end{aligned}$$

Calculating the total derivative of a (derivative-free) term is a straightforward implementation of calculus:

$$\begin{aligned}
[\![]\!]_{\nabla e} &: & e &\hookrightarrow e \\
[\![e_1 + e_2]\!]_{\nabla e} &\stackrel{\wedge}{=} & [\![e_1]\!]_{\nabla e} &+ [\![e_2]\!]_{\nabla e} \\
[\![e_1 \times e_2]\!]_{\nabla e} &\stackrel{\wedge}{=} & [\![e_1]\!]_{\nabla e} \times e_2 &+ e_1 \times [\![e_2]\!]_{\nabla e} \\
[\![c]\!]_{\nabla e} &\stackrel{\wedge}{=} & &0 \\
[\![\tau]\!]_{\nabla e} &\stackrel{\wedge}{=} & &1 \\
[\![u_i]\!]_{\nabla e} &\stackrel{\wedge}{=} & &\textsc{der}(u_i)
\end{aligned}$$

Again, it comes in handy that our term language is so small. However, the above function can be generalized for more complicated input languages using techniques like automatic differentiation (Höger (2013)). Hence we conjecture that differentiation is possible for *all* quasi-linear equations derived from *all* Modelica equations.

In order to calculate the constraints of a regularized DAE we consider the output of the regularization as a function $\mathbf{c} : ql \rightarrow \mathbb{N}$ from quasi-linear equations to the amount of desired differentiations. Given such a function, a system of quasi-linear equations QL can be expanded by regularization $[\![\ldots]\!]_{\text{reg}}$:

$$\begin{aligned}
[\![]\!]_{\text{reg}} \quad &: \quad (ql \rightarrow \mathbb{N}) \times QL \times \mathbb{N} \times \iota \rightarrow QL \times \mathbb{N} \times \iota \\
[\![\mathbf{c}, \emptyset, n, \iota]\!]_{\text{reg}} &\stackrel{\wedge}{=} \quad \langle \emptyset, n, \iota\rangle \\
[\![\mathbf{c}, \{ql\} \cup QL, n, \iota]\!]_{\text{reg}} &\stackrel{\wedge}{=} \quad \langle \{ql_0 \ldots ql_k\} \cup \tilde{QL}, m, \kappa\rangle \\
\text{where} & \\
k &= \mathbf{c}(ql) \\
\langle ql_0, n_0, \iota_0\rangle &= \langle ql, n, \iota\rangle \\
\langle ql_{i+1}, n_{i+1}, \iota_{i+1}\rangle &= [\![[\![ql_i, n_i, \iota_i]\!]_{\nabla ql}]\!]_{qlt} \\
\langle \tilde{QL}, m, \kappa\rangle &= [\![\mathbf{c}, QL, n_k, \iota_k]\!]_{\text{reg}}
\end{aligned}$$

After this process, the resulting augmented matrix $\tilde{E|k}$ (including rows from identities) is probably non-squared. To reconcile this property and gather all hidden constraints, we attempt to eliminate superfluous rows from the matrix e.g. by symbolic Gaussian elimination.This reconcilation depends on the symbolic equivalence of equation terms. While this is possible (e.g. by using a suitable computer algebra system) for our small term-language, equivalence is of course undecidable for Turing-complete terms (as they are used in Modelica). Hence, the process is not practically applicable to *every* model. We conjecture however, that such a limitation exists for every symbolic processing of models.

If this process succeeds, all the removed rows have no coefficients and are thus hidden constraints.

4.5 Example

To support our claim that $[\![]\!]_{qlt}$ is not a trivial and hence useless transformation, we resort to example 2.1. After setting $m = L = 1$ for simplification, it can be expressed in our simple term language (extended with subtraction for brevity) as:

$$\begin{aligned}
\textsc{der}(u_1) - u_3 \\
\textsc{der}(u_2) - u_4 \\
u_1 \times u_1 + u_2 \times u_2 - 1 \\
\textsc{der}(u_3) + 2 \times u_1 \times u_5 \\
\textsc{der}(u_4) + 2 \times u_2 \times u_5 + g
\end{aligned}$$

The first and second equation are obvious identities in the sense of ι. Hence, our transformation can be jump-started (if we omit this jump-start, the identities would be copied later on) using these identities and yields 3 quasi-linear equations and said identities:

$$\begin{aligned}
ql_1 &= \langle \emptyset, u_1 \times u_1 + u_2 \times u_2 - 1\rangle \\
ql_2 &= \langle \{u_3 \mapsto 1\}, 2 \times u_1 \times u_5\rangle \\
ql_3 &= \langle \{u_4 \mapsto 1\}, 2 \times u_2 \times u_5 + g\rangle \\
\iota &= \{u_1 \mapsto u_3, u_2 \mapsto u_4\}
\end{aligned}$$

The output of a regularization step will ask us to add the first and second derivative of equation ql_1 to the system (it will also ask us to add the first derivative of the identities, but this can be ignored since identities and their derivatives are inlined implicitly). Doing so yields:

$$\begin{aligned} ql_4 &= \langle \emptyset, u_1 \times u_3 + u_2 \times u_4 \rangle \\ ql_5 &= \langle \{u_3 \mapsto u_1, u_4 \mapsto u_2\}, u_3 \times u_3 + u_4 \times u_4 \rangle \end{aligned}$$

No additional identities are required for this simple example. Equations ql_1 and ql_4 are arguably constraints (although ql_1 is not hidden). The resulting augmented coefficient matrix can be seen below:

$$\left[\begin{array}{cccc|c} u_3 & & & & 0 \\ & u_4 & & & 0 \\ & & 1 & & 2 \times u_1 \times u_5 \\ & & & 1 & 2 \times u_2 \times u_5 + g \\ & & u_1 & u_2 & u_3 \times u_3 + u_4 \times u_4 \end{array}\right]$$

The last row can be eliminated by subtracting the third and fourth rows (multiplied with the corresponding coefficient), which yields the third hidden constraint:

$$u_3 \times u_3 + u_4 \times u_4 - 2 \times u_1 \times u_1 \times u_5 - 2 \times u_2 \times u_2 \times u_5 + g$$

This is precisely the term we would expect from translating equation (4g). The resulting system E, k and h can be fed into `QUALIDAES` and (given a consistent initial point) integrated over time without any further state-selection.

5 Conclusions

In this article we have discussed the efficient and robust numerical simulation of dynamical systems that are modeled with MODELICA. We have presented a regularization method for quasi-linear DAEs that is based on an over-determined system formulation that is obtained by adding all hidden constraints explicitly to the original model equation. The over-determined system formulation can then directly be integrated using the software package `QUALIDAES`. The great advantage of the direct discretization of the over-determined formulation is the fact that it is not necessary to determine a dynamic (state) selector outside of the solver `QUALIDAES` since this is achieved automatically within the separated treatment of (8) by its numerical solution, described above. Performing the state selection within the numerical integrator also allows us to switch between different state selections and also opens the door to handle structure varying system models Pepper et al. (2011). Furthermore, the number of unknowns in the DAE is not increased. A further advantage of an over-determined regularization with respect to the numerical integration is the possibility to add solution invariants, e.g., mass, impulse or energy conservation laws, to the constraints, which often stabilizes numerical integration.

Nevertheless, `QUALIDAES` requires a quasi-linear representation of the model equations. As we have shown, MODELICA-style equations can be transformed into quasi-linear form in a non-trivial way. This transformation preserves enough symbolic information about the equations to allow for the description of the hidden constraints. On the other hand, non-symbolic (i.e. algorithmic) parts can still be dealt with due to the introduction of identities with new variables.

5.1 Future Work

Although clearly necessary, the expansion of the input language of the quasi-linear transformation seems to be merely technically challenging. There are however some areas of future research that should be considered:

First, the search for a consistent initial point is a well-known challenging problem. It remains an open question, whether the quasi-linear transformation could provide any help in that area.

Furthermore, the transformations are currently implemented in a straightforward manner. Hence, the outcome might be non-optimal for practical applications (e.g. the size of the derived expressions might harm the simulation performance). It could be interesting to search for variants of the transformation that maintains practically useful properties (e.g. minimal tree size, minimal identities added).

Finally, we have already shown that the regularization of a structurally varying DAE can be implemented in an efficient, dynamic algorithm (see Höger (2014)). This is, obviously, of little value when the application of its results remains a non-dynamic monolithic algorithm. Hence any representation, but especially the quasi-linear form (since it is well-suited for structurally varying systems) should be enhanced with a *dynamic* regularization that preserves as much information from earlier modes as possible.

Acknowledgments

This work has been supported by the European Research Council through Advanced Grant *MODSIMCONMP* and by the German Research Foundation (Deutsche Forschungsgemeinschaft DFG) within the project "Automatische Modellierung und Simulation von technischen Systemen mit Unsicherheiten" *AMSUN*.

References

R. Altmeyer and A. Steinbrecher. Regularization and numerical simulation of dynamical systems modeled with Modelica. Preprint 29-2013, Institut für Mathematik, TU Berlin, 2013.

K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential Algebraic Equations*, volume 14 of *Classics in Applied Mathematics*. SIAM, Philadelphia, PA, 1996.

P. Deuflhard. *Newton methods for nonlinear problems. Affine invariance and adaptive algorithms*, volume 35 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2004.

C.W. Gear. Differential-algebraic equation index transformations. *SIAM Journal on Scientific and Statistic Computing*, 9:39–47, 1988.

E. Griepentrog and R. März. *Differential-Algebraic Equations and Their Numerical Treatment*, volume 88 of *Teubner-Texte zur Mathematik*. BSB B.G.Teubner Verlagsgesellschaft, Leipzig, 1986.

E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II - Stiff and Differential-Algebraic Problems*. Springer-Verlag, Berlin, Germany, 2nd edition, 1996.

D.J. Higham and N.J. Higham. *MATLAB Guide*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2005. ISBN 0-89871-578-4.

C. Höger. Operational semantics for a modular equation language. *AVICPS 2013*, page 5, 2013.

C. Höger. Dynamic structural analysis for daes. In *Proceedings of the 2014 Summer Simulation Multiconference*, page 12. Society for Computer Simulation International, 2014.

P. Kunkel and V. Mehrmann. Index reduction for differential-algebraic equations by minimal extension. *Zeitschrift für Angewandte Mathematik und Mechanik*, 84(9):579–597, 2004.

P. Kunkel and V. Mehrmann. *Differential-Algebraic Equations. Analysis and Numerical Solution*. EMS Publishing House, Zürich, Switzerland, 2006.

S. Mattsson and G. Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM Journal on Scientific and Statistic Computing*, 14:677–692, 1993.

C.C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM Journal on Scientific and Statistic Computing*, 9:213–231, 1988.

P. Pepper, A. Mehlhase, Ch. Höger, and L. Scholz. A compositional semantics for Modelica-style variable-structure modeling. In P. Fritzson F.E. Cellier, D. Broman and E.A. Lee, editors, *4th International Workshop on Equation-Based Object-oriented Modeling Languages and Tools (EOOLT 2011)*, number 56 in Linköping Electronic Conference Proceedings, pages 45–54, Zurich, Switzerland, 2011. September 5, 2011.

J. Pryce. A simple structural analysis method for DAEs. *BIT Numerical Mathematics*, 41:364–394, 2001.

L. Scholz and A. Steinbrecher. A combined structural-algebraic approach for the regularization of coupled systems of DAEs. Preprint 30-2013, Institut für Mathematik, TU Berlin, 2013.

L. Scholz and A. Steinbrecher. Efficient numerical integration of dynamical systems based on structural-algebraic regularization avoiding state selection. In K.-E. Arzen H. Tummescheit, editor, *Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden*, number 96 in Linköping Electronic Conference Proceedings, pages 1171–1178. Modelica Association and Linköping University Electronic Press, 2014.

A. Steinbrecher. *Numerical Solution of Quasi-Linear Differential-Algebraic Equations and Industrial Simulation of Multibody Systems*. PhD thesis, Technische Universität Berlin, 2006.