Multi-Mode DAE Models - Challenges, Theory and Implementation

Albert Benveniste[1], Benoît Caillaud[1], Hilding Elmqvist[2], Khalil Ghorbal[1(✉)], Martin Otter[3(✉)], and Marc Pouzet[4]

[1] Inria, Rennes, France
{albert.benveniste,benoit.caillaud,khalil.ghorbal}@inria.fr
[2] Mogram AB, Lund, Sweden
hilding.elmqvist@mogram.net
[3] DLR-SR, Oberpfaffenhofen, Germany
martin.otter@dlr.de
[4] ENS, Paris, France
marc.pouzet@ens.fr

Abstract. Our objective is to model and simulate Cyber-Physical Systems (CPS) such as robots, vehicles, and power plants. The structure of CPS models may change during simulation due to the desired operation, due to failure situations or due to changes in physical conditions. Corresponding models are called multi-mode. We are interested in multi-domain, component-oriented modeling as performed, for example, with the modeling language Modelica that leads naturally to Differential Algebraic Equations (DAEs). This paper is thus about multi-mode DAE systems. In particular, new methods are discussed to overcome one key problem that was only solved for specific subclasses of systems before: How to switch from one mode to another one when the number of equations may change and variables may exhibit impulsive behavior? An evaluation is performed both with the experimental modeling and simulation system Modia, a domain specific language extension of the programming language Julia, and with SunDAE, a novel structural analysis library for multi-mode DAE systems.

Keywords: Multi-Mode systems · Cyber-Physical Systems · CPS
Modia · Modelica · Differential algebraic equations · DAE
Differential index · Structural analysis · Operational semantics
Constructive semantics · Nonstandard analysis

1 Introduction

Modeling with block diagrams described with Ordinary Differential Equations in state space form (ODE) has become a key pillar in the development of Cyber-Physical Systems (CPS). Block diagrams, however, suffer from a lack of modularity and reuse that is best illustrated in Fig. 1. This figure shows two

B. Steffen and G. Woeginger (Eds.): Computing and Software Science, LNCS 10000, pp. 1–28, 2018.
https://doi.org/10.1007/978-3-319-91908-9_16

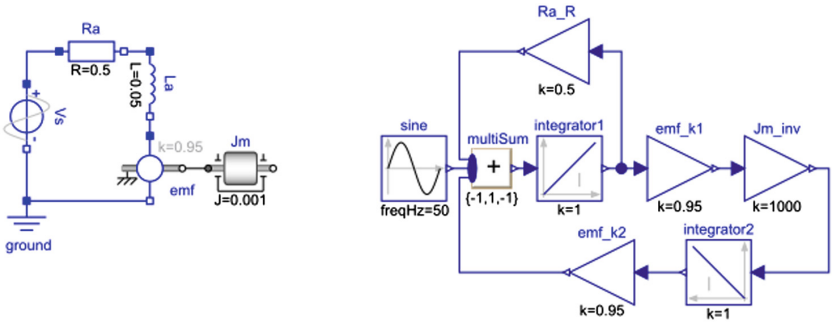


Fig. 1. DAE (left) vs. ODE (right) based modeling.

models of the same system consisting of a simple model of an electrical motor and of the rotational inertia of the motor. On the left hand side a (Modelica) schematic/component diagram of the system is shown that connects physical components through non-directed interactions resulting from the first principles of physics. On the right hand side the same model is shown as a block diagram in which input/output oriented blocks are connected with a directed wiring manually specified by the designer. Adding one more physical component is straightforward in the schematic, whereas it may need a complete redesign in the block diagram.

This should not come as a surprise, as the first principles of physics naturally lead to considering acausal models such as the one in Fig. 1-left. Consider, for example, the case of electric circuits. So-called *circuit laws* such as Kirchhoff laws, are naturally expressed as balance equations: the algebraic sum of currents in a network of conductors meeting at a point is zero; or, the sum of all the voltages around a loop is equal to zero. Similarly, some components (such as, e.g., resistors or capacitors) come with no input/output prespecified orientation. A same circuit can be assigned different input/output status for its variables, depending on which ones are declared as sources. The same situation arises in mechanics or in thermodynamics. Engineers interested in multi-physics modeling have identified this fact since the 70's by proposing *bond graphs* [17,31], in which electric circuits, mechanical systems, and thermodynamical systems, are abstracted to a common framework manipulating *efforts, flows,* and *junctions.* To summarize, for systems made of a large number of interconnected components, getting the usual input/output state space model (such as in Simulink block diagrams) becomes intractable, whereas it remains manageable if acausal models are supported.

In addition to being naturally described by an acausal model, a system may have different modes for different reasons. One reason is the physics: mechanical impacts where the bodies may remain in contact after a collision or an active grasping/docking, idealized electrical or hydraulic switching elements, are examples of situations requiring different sets of equations for their modeling. Another reason is the control of complex scenarios like repairing a satellite with robots,

changing operations of a system like a power plant with start-up, normal operation and shut down. Finally, safety needs may lead to so-called degraded modes in which the system dynamics may become very different (some components, sensors or actuators getting down for example). Models of this kind are called *multi-mode* models.

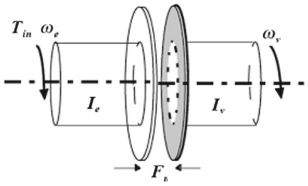


Fig. 2. A simple clutch with two shafts.

Acausal multi-mode models present, however, subtle difficulties at mode changes. Consider for instance the case of an idealized clutch model, illustrated in Fig. 2. The clutch possesses two modes: released (the two shafts rotate freely) and engaged (the two shafts are in contact). Deriving acausal models from the first principles is simple and rather elegant for each single mode alone. The intuition tells us that, when the clutch gets engaged, the different velocities of the two shafts will, in zero time, merge to a unique identical angular velocity under impulsive torques. One also expects that the resulting common velocity is entirely determined and depends only on the individual inertia and velocity of each shaft, prior to engagement. Determining manually the restart conditions is required if one resorts to using oriented block-diagrams like in Simulink. We argue that such task is difficult for this simple example and becomes even impossible for real-life models. Those restart conditions should rather be synthesized automatically by the compiler and made available at run time to simulate the model. This work advocates therefore the use of acausal modelling languages since they

1. make the specification of large (single mode) Cyber Physical Systems more modular, more elegant, and with better reuse;
2. are indispensable to synthesize proper restart conditions at mode changes, for models where this cannot be done manually—the idealized clutch model being a simple example.

It should also be clear that classical state space input/output formalisms (such as Simulink block diagrams) do not address the above challenges. Neither does the formalism of *hybrid automata* [1] used as the model of hybrid systems by several verification tools such as Flow* or SpaceEx.

Causal input/output state space models are based on Ordinary Differential Equations (ODE), of the form $\dot{x} = f(x, u)$ where x is the state and u the input, whereas acausal models à la Modelica are based on Differential Algebraic Equations (DAE) of the form $f(\dot{x}, x, v) = 0$ where x is a state and v an algebraic variable, for which no input or output status can be a priori specified.

In this paper we propose a theory to support the compilation of multi-mode DAE based models in a systematic and mathematically sound way. We want to reject some models that are in some sense spurious. And we want to synthesize (whenever needed, not in all cases) the computation of the restart conditions at mode changes. All of this will be performed at compile time, through a special static analysis called the *structural analysis,* which is an important topic of this paper. This structural analysis is an essential preprocessing step prior to generating simulation code. We will illustrate our purpose by an example from electro-mechanics that includes a clutch as one of its subsystems. The specification, analysis, compilation, and simulation of this model will be illustrated using the recently proposed experimental modeling language called Modia.

2 State-of-the-Art and Related Work

There exists an extensive body of work related to the numerical solution of *single-mode DAEs*, especially [9], as well as to the modeling and simulation of *single-mode multi-domain* models leading to DAEs[1].

Mechanical systems featuring impulses are known since the 18th century. A large literature about multi-body systems with contacts and impulses exists, for example [25,26]. The current research focuses on simulating contacts between many bodies using time stepping methods. Contact equations are usually described approximately on velocity or acceleration level (if contact occurs, the relative velocity or acceleration between the relevant bodies is constrained to be zero). Since the constraints on position level are not explicitly taken into account, typically a drift occurs that is usually not relevant for systems with many contacting bodies but is not acceptable for general multi-mode DAEs. An exception is [29], where constraints on position level are enforced. It is not obvious how the specialized multi-mode methods for multi-body systems can be generalized to any type of multi-mode DAE.

Also for electrical circuits with idealized switches, like ideal diodes, impulses can occur. Again a large literature is available concentrating mostly on specialized electrical circuits such as piecewise-linear networks with idealized switches [2,16,32]. Again, it is not obvious how to generalize methods in this specialized area to general multi-mode DAEs.

The paper of Mehrmann et al. [20] contains interesting results regarding numerical techniques to detect chattering between modes. It, however, assumes that consistent reset values are explicitly given for each mode. Such an assumption does not hold in general, especially for a compositional framework where one wants to assemble pre-defined physical components.

In the PhD-thesis of Zimmer [33] variable-structure multi-domain DAEs are defined with a special modeling language and a run-time interpreter is used that processes the DAE equations at run-time, when the structure and/or the index is changing. Limitations of this work are that impulsive behavior is not supported

[1] See the extensive literature available in https://www.modelica.org/publications.

and that the user has to define the transfer of variable values from one mode to the next mode explicitly, which is not practical for large models.

Describing variable structure systems with *causal* state machines is discussed by Pepper et al. [24].

Elmqvist et al. [13,19] propose a high level description of multi-mode models as an extension to the synchronous Modelica 3.3 state machines by using continuous-time state machines having continuous-time models as "states". Besides ODEs as used in hybrid automata, also *acausal DAE models with physical connectors* can be a "state" of a state machine. Such a state machine is mapped into a form so that the resulting equations can be processed by standard symbolic algorithms supported by Modelica tools. The major restrictions of this approach are that mode changes with impulsive behavior are not supported and that not all types of multi-mode systems can be handled due to the static code generation.

Benveniste et al. [3,5] tackle the problem of variable structure, varying index DAEs from a fundamental point of view by using a low level description of DAEs with a few language elements only and a precise mathematical description of the semantics based on non-standard analysis. A proof-of-concept mockup, SunDAE, was developed implementing this approach.

3 The Modia Language

Modelica [21] is a state-of-the-art modeling language for multi-domain modeling. Recently, an experimental language which is similar to Modelica and called Modia [11,12] has been designed and implemented. Modia is a domain specific language extension of Julia [8] by means of structured macros, that is, the Julia parser is used to parse Modia models. The Modia language elements will be introduced through a small set of examples.

Elementary Modia Constructs. A model of a rotating inertia can be defined as follows (`#` start a comment until end of the line):

```
@model Inertia begin
  J = 1
  w0 = 0.0
  w = Float(start = w0)
  flange = Flange()
  @equations begin
    w = flange.w
    J*der(w) = flange.tau
  end
end
```

The `@model` macro has one section for declarations and one section for equations and connections. The construct `J = 1` defines the parameter `J` with default value of 1. A variable `w` with a start value of `w0` is introduced by `w = Float(start = w0)`. The construct `flange = Flange()` calls the constructor for Flange which is defined as follows:

```
@model Flange begin
  w = Float()                 # Angular velocity
  tau = Float(flow=true)      # Torque
end
```

This model consists just of two variable declarations with type Float modeling the interaction when the Inertia is mechanically coupled via a shaft to other mechanical components. The attribute `flow = true` tells that if several shafts are connected to flange, the internal and external torques are all summed to zero.

The equation `J*der(w) = flange.tau` is Euler's equation for the rotational motion of the inertia. The operator `der()` denotes time derivative.

A model with two rigidly connected inertias can be defined as shown below:

```
@model TwoInertias begin
  inertia1 = Inertia(J=0.1)
  inertia2 = Inertia(J=0.4, w0=10.0)
  @equations begin
    connect(inertia1.flange, inertia2.flange)
  end
end
```

Note that parameters are changed in the Inertia constructor call. The `connect()` primitive models a physical coupling. In this case the semantics is:

```
  inertia1.flange.w = inertia2.flange.w
  inertia1.flange.tau + inertia2.flange.tau = 0
```

Note, that inertia2 has initial angular velocity of 10 rad/s, whereas inertia1 has the default of 0. The result of simulation shows a constant angular velocity for both inertias of 8 rad/s. Since the initial angular velocities are different there will be a Dirac impulse in the torque to make them equal. The resulting joint angular velocity is according to theory weighted as $w = (J1*w1+J2*w2)/(J1+J2) = 8$.

Advanced Modia Constructs. Modia has many other features, such as inheritance, size and type inference, type declarations, component redeclarations, time events, state event, and nested simulations. More information can be found in [11,12]. It is possible to declare variables with specific SI units. As an example, an electric Pin connector can be defined as shown below using predefined variable constructors Voltage() and Current() with associated SI units (Volt and Ampere):

```
@model Pin begin
  v = Voltage()
  i = Current(flow=true)
end
```

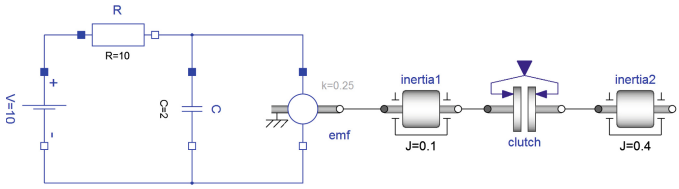


Fig. 3. Schematics of MotorAndLoad model.

Modia Multi-mode Modeling: Running Example. We model in Modia a multi-mode DAE system containing the idealized clutch introduced informally in Sect. 1. The model has an electric motor, two load inertia and a clutch, see Fig. 3. The electric motor model is based on a model of the electro-mechanical force (emf). It has two Pin connectors (p and v) and one Flange connector. The specific equations for an emf are:

```
k*flange.w = p.v - n.v
flange.tau = -k*p.i
```

Note that a Capacitor is connected across the emf.

A simple Clutch model is used which is either engaged or not engaged. It is modeled as two different set of equations:

```
@model Clutch begin
  flange1 = Flange()
  flange2 = Flange()
  engaged = Boolean()
@equations begin
    if engaged
      flange1.w = flange2.w
      flange1.tau + flange2.tau = 0
    else
      flange1.tau = 0
      flange2.tau = 0
    end
  end
end
```

When the clutch is engaged, the angular velocities of the two flanges are the same and the torques sum to zero. When not engaged, the torques are zero and there are no constraints on the angular velocities. The definition of the input `engaged` is made in the surrounding environment of the clutch:

```
clutch.engaged = time < 100 || time >= 300
```

The DAE index is changing depending on the state of the clutch. Furthermore, Dirac impulses occur initially and when the mode changes to engaged. The techniques needed to analyze and simulate such models are described in subsequent sections.

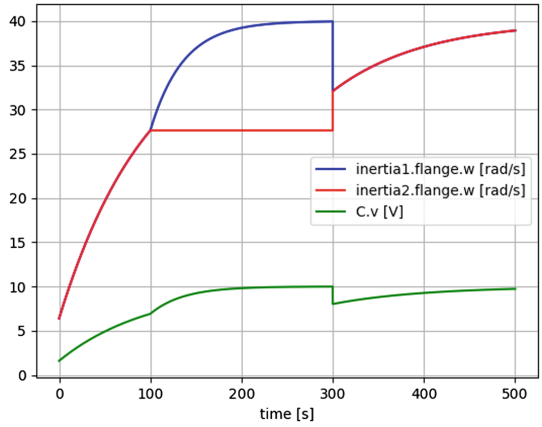


Fig. 4. Simulation results of the MotorAndLoad model.

The results of a simulation are shown in Fig. 4. The initial angular velocity of inertia1 is 0 rad/s and 10 rad/s for inertia2. The capacitor is initially uncharged. The upper two partly overlapping curves are the angular velocities of inertia1 and inertia2. When the clutch is disengaged, the angular velocity of inertia2 is constant. The lower curve shows the voltage over the capacitor. Since the clutch is engaged at initialization, Dirac impulses occur at time = 0 s. As a result, after initialization the angular velocities of the two inertia are identical. The common angular velocity at time = 0 s is not 8 rad/s as in the example TwoInertias, but 6.4 rad/s. The reason is that the capacitor acts in the same way as an additional moment of inertia to inertia1. The effective inertia is $J1' = J1 + k^2 * C = 0.1 + 0.25^2 * 2 = 0.225$. Thus, the angular velocity at time=0 s becomes $w = (J1' * w1 + J2 * w2)/(J1' + J2) = (0.225 * 0 + 0.4 * 10)/(0.225 + 0.4) = 6.4$ rad/s. There are no Dirac impulses at time = 100 s when the clutch disengages, but again Dirac impulses at time = 300 s when the clutch engages again.

4 Simulating a Restricted Class of Multi-Mode DAEs

In this section, we propose a method to simulate a restricted class of multi-mode DAE systems, encompassing in particular our example of Fig. 3 and its simulation result in Fig. 4.

Throughout this paper, we use the classical notations $\dot{x}$ and $\ddot{x}$ to denote the first and second time-derivatives of a function of continuous time $x : \mathbb{R} \mapsto \mathbb{R}$. When such a notation is not appropriate for readability reasons, we sometimes write instead x' and x''. The notation for higher order derivatives is indicated when needed. We write vectors and matrices in boldface. The transpose of a matrix $\boldsymbol{G}$ is written $\boldsymbol{G}^T$.

4.1 Problem Setting

The goal is to simulate the following class of multi-mode DAE systems:

$$\begin{aligned} \textbf{if } \gamma_1(\dot{\boldsymbol{x}}_\gamma^-, \boldsymbol{x}_\gamma^-, t) \textbf{ then } \boldsymbol{f}_1(\dot{\boldsymbol{x}}_1, \boldsymbol{x}_1, t) &= \boldsymbol{0} \\ \textbf{elseif } \gamma_2(\dot{\boldsymbol{x}}_\gamma^-, \boldsymbol{x}_\gamma^-, t) \textbf{ then } \boldsymbol{f}_2(\dot{\boldsymbol{x}}_2, \boldsymbol{x}_2, t) &= \boldsymbol{0} \\ &\vdots \\ \textbf{else } \boldsymbol{f}_m(\dot{\boldsymbol{x}}_m, \boldsymbol{x}_m, t) &= \boldsymbol{0} \end{aligned} \tag{1}$$

where

- vector $\boldsymbol{x}_\gamma$ of length n_γ collects variables that are present in all modes and in the predicates $\gamma_i(\dots)$,
- vector $\boldsymbol{x}_i$ of length n_i collects all variables that are used in mode i ($\boldsymbol{x}_\gamma$ is a subset of $\boldsymbol{x}_i$), and
- $\gamma_i \in \mathbb{R}^{n_\gamma} \times \mathbb{R}^{n_\gamma} \times \mathbb{R} \to \text{Boolean}, \quad \boldsymbol{f}_i \in \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \times \mathbb{R} \to \mathbb{R}^{n_i}$.

Hence the considered system switches between different modes, each described by a DAE having n_i equations and n_i unknowns. The switching conditions are predicates denoted by γ_i and depend on time and/or on the left limit of the DAE variables assuming they exist and are finite, that is

$$\boldsymbol{x}_\gamma^-(t) =_{\text{def}} \lim_{\varepsilon \searrow 0} \boldsymbol{x}_\gamma(t-\varepsilon) \text{ and } \dot{\boldsymbol{x}}_\gamma^-(t) =_{\text{def}} \lim_{\varepsilon \searrow 0} \dot{\boldsymbol{x}}_\gamma(t-\varepsilon).$$

When changing from mode i to mode j it is assumed that the initial conditions $\boldsymbol{x}_\gamma^-$ are known from the previous mode. A concrete example will be given in Sect. 4.3. we also refer to [13] for a more generic treatment. When changing from one mode to another the set of equations that govern the dynamics of the system may change. Thus, some variables may experience discontinuities or even impulses. Throughout this section we assume that the system spends a strictly positive duration in each mode in such a way that instants of mode changes are cleanly isolated.

Note that the class (1) of multi-mode systems is not compositional as the predicates $\boldsymbol{x}_\gamma$ are global. We restricted ourselves to this class in order to simplify the exposure below. This restriction is relaxed in the actual implementation in Modia. For instance, `@model Clutch` in Sect. 3 has a *local* definition of the predicate to engage or disengage the clutch and there is *no global* if-clause. In Sect. 5, where an alternative approach is discussed, such restrictions are not present.

4.2 Handling Mode Changes

In a mode i, an initial value problem $\boldsymbol{f}_i(\dot{\boldsymbol{x}}_i, \boldsymbol{x}_i, t) = \boldsymbol{0}$ has to be solved. It is well-known that only special classes of DAEs can be directly numerically integrated [9]. For this reason, general DAEs might be first transformed in to a DAE class where reliable integration methods exist. For nonlinear systems such

a transformation means that equations of the DAE might need to be (analytically) differentiated. The best integration methods exist for ODEs $\dot{\boldsymbol{x}} = \bar{\boldsymbol{f}}(\boldsymbol{x}, t)$. However, transforming a DAE to this form requires in general to solve nonlinear algebraic equation systems at every model evaluation and therefore the benefit of ODE integration methods might get lost. For this reason, another approach is used in this section where the transformation to a numerically solvable form is performed *without* solving algebraic equation systems. In Assumption 1 we define the target DAE class for this transformation:

Assumption 1. *For each mode i of System (1), the DAE $\boldsymbol{f}_i(\dot{\boldsymbol{x}}_i, \boldsymbol{x}_i, t) = \boldsymbol{0}$ has the following form (we omit the index i for better readability)*

$$\begin{aligned} \boldsymbol{f}_d(\dot{\boldsymbol{x}}, \boldsymbol{x}, t) &= \boldsymbol{0} \\ \boldsymbol{f}_c(\boldsymbol{x}, t) &= \boldsymbol{0} \end{aligned} \tag{2}$$

$$\textit{where } \boldsymbol{J} = \begin{bmatrix} \dfrac{\partial \boldsymbol{f}_d}{\partial \dot{\boldsymbol{x}}} \\ \dfrac{\partial \boldsymbol{f}_c}{\partial \boldsymbol{x}} \end{bmatrix} \quad \textit{is regular (that is, invertible).} \tag{3}$$

In (2), subscripts "$_d$" and "$_c$" are reminiscent of "dynamics" and "constraint", respectively. Condition (3) means that System (2) has differentiation index one.[2]

Transforming General DAEs to System (2, 3). If the model equations in a given mode are *not* in the form (2, 3), they are transformed to it. This transformation is non-trivial and it is beyond the scope of this paper to explain the details. Only a short overview is given here: Typically, the Pantelides algorithm [23], or a variant like [27], is used to determine which equations of the original DAE must be differentiated in order that the highest derivative variables can be uniquely determined from the highest derivative equations. By differentiating the corresponding equations analytically it is then possible to transform to ODE form. Hereby, algebraic equation systems might need to be solved. In [22] a new algorithm is proposed to transform every DAE that can be treated with the Pantelides algorithm to the form of Assumption 1 *without solving algebraic equation systems.* This algorithm is a generalization of [15] that was developed for single-mode multi-body systems. In Sect. 4.3 it is applied to multi-mode multi-body systems.

Simulating System (2, 3). A number of methods exist for solving system (2, 3) numerically. In particular, fixed or variable step-size BDF (Backward Differentiation Formula) methods can be used [9]. In all cases, consistent initial conditions have to be provided that fulfill (2) and the so-called *latent equations*

$$\frac{\partial \boldsymbol{f}_c}{\partial \boldsymbol{x}}(\boldsymbol{x}, t)\, \dot{\boldsymbol{x}} + \frac{\partial \boldsymbol{f}_c}{\partial t}(\boldsymbol{x}, t) = \boldsymbol{0} \tag{4}$$

[2] A DAE $\boldsymbol{f}(\dot{\boldsymbol{x}}, \boldsymbol{x}, t) = 0$ has *differentiation index* n if one or more equations must be differentiated n-times until the equations can be algebraically transformed to an ODE form $\dot{\boldsymbol{x}} = \bar{f}(\boldsymbol{x}, t)$, see for example [30].

obtained by differentiating the constraint $\boldsymbol{f}_c(\boldsymbol{x}, t) = \boldsymbol{0}$, see [9,23]. Integration methods, including BDF, assume that $\boldsymbol{x}$ is smooth. Therefore, standard integration methods cannot be applied if $\boldsymbol{x}$ is discontinuous at a mode change.

Computing Restart Values. We now propose a scheme for computing restart conditions. More precisely, let t_{ev} be an instant of mode change of System (1), meaning that one of the guards γ_i changes its value. The objective is to compute consistent restart values

$$\boldsymbol{x}^+ = \boldsymbol{x}(t_{ev}^+) =_{\text{def}} \lim_{s \searrow t_{ev}} \boldsymbol{x}(s) \text{ and } \dot{\boldsymbol{x}}^+ = \dot{\boldsymbol{x}}(t_{ev}^+) =_{\text{def}} \lim_{s \searrow t_{ev}} \dot{\boldsymbol{x}}(s) \tag{5}$$

for the index one system (2, 3), given values

$$\boldsymbol{x}^- = \boldsymbol{x}(t_{ev}^-) =_{\text{def}} \lim_{s \nearrow t_{ev}} \boldsymbol{x}(s) \tag{6}$$

just prior to entering the new mode. Note that, since $\boldsymbol{x}^-$ are variable values in the previous mode, they will not, in general, be *consistent* for the new mode, that is the second equation of System (2) might not be satisfied by $\boldsymbol{x}^-$. Still, these equations must be satisfied by the (yet unknown) $\boldsymbol{x}^+$. Since $\boldsymbol{x}(t)$ may be discontinuous at t_{ev}, the derivative of $\boldsymbol{x}(t)$ may be a Dirac impulse.

To derive our method for computing the restart values, we first restrict the class of systems with the following additional assumption. We then discuss the general case.

Assumption 2. *Assume DAE System (2) has the following special structure:*

$$\begin{aligned} 0 &= \boldsymbol{A}(\boldsymbol{x}_s, t)\dot{\boldsymbol{x}} + \boldsymbol{b}(\boldsymbol{x}, t) \qquad (= \boldsymbol{f}_d(\dot{\boldsymbol{x}}, \boldsymbol{x}, t)) \\ 0 &= \boldsymbol{f}_c(\boldsymbol{x}, t) \end{aligned} \tag{7}$$

where $\boldsymbol{x}_s$ collects the smooth elements of $\boldsymbol{x}$, that is those being continuous and of bounded variation around the mode change.[3] *Other elements of $\boldsymbol{x}$ might be discontinuous. Furthermore, we assume that $\boldsymbol{x}, \boldsymbol{A}(\dots), \boldsymbol{b}(\dots)$ are continuous functions of their arguments and remain bounded around the instant of mode change.*

Note that Assumption 2 does not forbid that the triple defining the dynamics (7) actually varies with the mode i, that is has the form $(\boldsymbol{A}_i, \boldsymbol{b}_i, \boldsymbol{f}_{c,i})$. Since elements of $\boldsymbol{x}$ may be discontinuous, $\dot{\boldsymbol{x}}$ may have Dirac impulses. In (7) it is therefore assumed that $\dot{\boldsymbol{x}}$ having potentially Dirac impulses appear linearly and the linear factors are continuous functions without discontinuities.

Under Assumption 2, the mathematical solution of the restart problem can be determined as follows: In a first step, since $\boldsymbol{x}_s$ is smooth at the instant t_{ev} of mode change, the matrix $\boldsymbol{A}(\boldsymbol{x}_s, t_{ev})$ is immediately known once a change has been detected at t_{ev}. To evaluate all the elements of $\boldsymbol{x}$ right after t_{ev} we

[3] A function $f : \mathbb{R} \mapsto \mathbb{R}$ is said to have bounded variation if it is the primitive of a Lebesgue integrable function [10]. As a consequence, $\int_t^{t+h} \dot{f}(s)ds \to 0$ when $h \to 0$.
.

proceed by integrating (7) over $[t_{ev}-\varepsilon, t_{ev}+\varepsilon]$ using the well known properties of Lebesgue integrals and Dirac measure. To this end, we decompose

$$\boldsymbol{x} \;=\; \begin{bmatrix} \boldsymbol{x}_s \\ \boldsymbol{x}_{ns} \end{bmatrix} \text{ and } \boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_s \; \boldsymbol{A}_{ns} \end{bmatrix}$$

where $\boldsymbol{x}_{ns}$ collects the nonsmooth entries of $\boldsymbol{x}$ that may experience discontinuities at the instant of mode change and $\boldsymbol{A}$ is decomposed accordingly. The following approximations hold, where the integrals $\int_{t_{ev}-\varepsilon}^{t_{ev}+\varepsilon}$ mean $\int_{[t_{ev}-\varepsilon, t_{ev}+\varepsilon]}$:

$$\begin{aligned}\int_{t_{ev}-\varepsilon}^{t_{ev}+\varepsilon} \boldsymbol{f}_d(\dot{\boldsymbol{x}}, \boldsymbol{x}, t)dt &= \underbrace{\int_{t_{ev}-\varepsilon}^{t_{ev}+\varepsilon} \left(\boldsymbol{A}_s(\boldsymbol{x}_s, t)\dot{\boldsymbol{x}}_s + \boldsymbol{b}(\boldsymbol{x}, t)\right) dt}_{\approx 0 \text{ by footnote 3}} + \underbrace{\int_{t_{ev}-\varepsilon}^{t_{ev}+\varepsilon} \boldsymbol{A}_{ns}(\boldsymbol{x}_s, t)\dot{\boldsymbol{x}}_{ns}dt}_{\dot{\boldsymbol{x}}_{ns}dt \text{ is a Dirac at } t_{ev}} \\ &\approx \boldsymbol{A}_{ns}(\boldsymbol{x}_s, t_{ev})\left(\boldsymbol{x}_{ns}(t_{ev}+\varepsilon) - \boldsymbol{x}_{ns}(t_{ev}-\varepsilon)\right) \\ &\approx \boldsymbol{A}(\boldsymbol{x}_s, t_{ev})\left(\boldsymbol{x}(t_{ev}+\varepsilon) - \boldsymbol{x}(t_{ev}-\varepsilon)\right)\end{aligned}$$

In the above two approximations, the first one follows from the property of the Dirac measure and the fact that $t \mapsto \boldsymbol{A}_{ns}(\boldsymbol{x}_s(t), t)$ is continuous, and the second one follows from the fact that $\boldsymbol{A}_s(\boldsymbol{x}_s, t_{ev})\left(\boldsymbol{x}_s(t_{ev}+\varepsilon) - \boldsymbol{x}_s(t_{ev}-\varepsilon)\right) \approx 0$.

This leads to proposing the following scheme for computing the restart values $\boldsymbol{x}^+$ from $\boldsymbol{x}^-$ at the instant t_{ev} of mode change, cf. Eqs. (5) and (6):

$$\begin{aligned} 0 &= \boldsymbol{A}(\boldsymbol{x}_s, t_{ev})(\boldsymbol{x}^+ - \boldsymbol{x}^-) \\ 0 &= \boldsymbol{f}_c(\boldsymbol{x}^+, t_{ev}) \end{aligned} \tag{8}$$

Since (7) may not hold at t_{ev}^-, and $\boldsymbol{x}^+ - \boldsymbol{x}^-$ need not to be small, the Implicit Function Theorem cannot be invoked to argue about the existence and uniqueness of a solution around $\boldsymbol{x}^-$. Since the Jacobian $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}^+}$ of (8) is regular due to (3) and Assumption 2, (8) can be numerically solved with a Newton-type method. Fortunately, in many practical cases (such as a mechanical impact) the nonlinear part of $\boldsymbol{f}_c(\boldsymbol{x}, t)$ is often satisfied by $\boldsymbol{x}^-$ and then only a linear equation system needs to be solved for $\boldsymbol{x}^+$, in which case a unique solution exists, see also Sect. 4.3. The physical interpretation of (8) is that we look for restart values that are consistent for the new mode (second equation) and meet the integral of $\boldsymbol{f}_d(\dots)$ across the mode change (first equation).

In a modeling language such as Modelica or Modia, the model code has the form (2), not the special form (7). It is therefore desirable to compute restart values only with the form (2), without being forced to generate special code that reveals the details of the equations as in (7). On the basis of Assumption 1, we can nevertheless consider the following two systems of equations, where $\boldsymbol{x}^+$ are the unknowns and $h > 0$ is small:

mixed explicit/implicit Euler	implicit Euler
$0 = h\boldsymbol{f}_d\left(\frac{\boldsymbol{x}^+ - \boldsymbol{x}^-}{h}, \boldsymbol{x}^-, t_{ev}\right)$	$0 = h\boldsymbol{f}_d\left(\frac{\boldsymbol{x}^+ - \boldsymbol{x}^-}{h}, \boldsymbol{x}^+, t_{ev}+h\right)$
$0 = \boldsymbol{f}_c(\boldsymbol{x}^+, t_{ev}+h)$	$0 = \boldsymbol{f}_c(\boldsymbol{x}^+, t_{ev}+h)$

(9)

Schemes (9) only require Assumption 1. Under Assumption 2, Schemes (9) take the form

mixed explicit/implicit Euler	implicit Euler
$0 = \boldsymbol{A}(\boldsymbol{x}_s, t_{ev})(\boldsymbol{x}^+ - \boldsymbol{x}^-) + h \cdot \boldsymbol{b}(\boldsymbol{x}^-, t_{ev})$	$0 = \boldsymbol{A}(\boldsymbol{x}_s, t_{ev} + h)(\boldsymbol{x}^+ - \boldsymbol{x}^-) + h \cdot \boldsymbol{b}(\boldsymbol{x}^+, t_{ev} + h)$
$0 = \boldsymbol{f}_c(\boldsymbol{x}^+, t_{ev} + h)$	$0 = \boldsymbol{f}_c(\boldsymbol{x}^+, t_{ev} + h)$

(10)

where we have used the fact that $\boldsymbol{x}_s^- \approx \boldsymbol{x}_s^+$ since $\boldsymbol{x}_s$ is smooth. These schemes reduce to (8) for $h \approx 0$.

Thus, if Assumption 2 holds, the form (2) for the system model at mode i can be used to numerically compute restart values for this mode: At the instant of the mode change integrate (2) with either mixed explicit/implicit Euler or implicit Euler schemes, from $\boldsymbol{x}^-$ over a small step-size h, thereby scaling the dynamic part of the model, $\boldsymbol{f}_d(..)$, with h. The solution of the nonlinear equation system (9) converges to the solution of the system (8).

Discussing the General Case. Schemes (9) can be applied to any index one system. Such a brute force use without Assumption 2, however, raises questions that we review now:

- Relaxing the continuity assumption on $\boldsymbol{A}(\dots)$, so that $\boldsymbol{A}(\boldsymbol{x}, t)$ might have discontinuous elements: As a result the first contribution in the decomposition of the integral $\int_{t_{ev}}^{t_{ev}+\varepsilon} \boldsymbol{f}_d(\dot{\boldsymbol{x}}, \boldsymbol{x}, t)dt$ is no longer negligible and it is unclear, from the literature, whether a well-defined meaning to this term exists.
- Removing the linearity assumption on $\dot{\boldsymbol{x}}$: The solution is no longer a Dirac and it is again unclear whether a well-defined solution exists. It is not even clear that h is the proper scaling factor for $\boldsymbol{f}_d$. Note that the linearity assumption quite often holds for physical system models, since balance equations in physics are linear in their derivatives.

To summarize, there is some evidence that precautions must be taken when relaxing Assumption 2.

Completing Consistent Restart. After the consistent restart values $\boldsymbol{x}^+$ have been computed, the corresponding consistent restart values of $\dot{\boldsymbol{x}}^+$ can be determined with (4) and the first equation of (2), by solving the following non-linear equation system in the unknowns $\dot{\boldsymbol{x}}^+$ (in case Assumption 2 holds, this is a linear equation system with a regular Jacobian, so a unique solution exists):

$$\begin{aligned} \boldsymbol{f}_d(\dot{\boldsymbol{x}}^+, \boldsymbol{x}^+, t_{ev}) &= 0 \\ \frac{\partial \boldsymbol{f}_c}{\partial \boldsymbol{x}}(\boldsymbol{x}^+, t_{ev})\dot{\boldsymbol{x}}^+ + \frac{\partial \boldsymbol{f}_c}{\partial t}(\boldsymbol{x}^+, t_{ev}) &= 0 \end{aligned} \tag{11}$$

Addressing Initialization. Note that the combined use of (9) and (11) provides a method for consistent initialization: The initial mode i and guesses for $\boldsymbol{x}_{\mathrm{i,init}}^-$ must be provided by the modeler. Afterwards, consistent values $(\dot{\boldsymbol{x}}_{\mathrm{i,init}}^+, \boldsymbol{x}_{\mathrm{i,init}}^+)$ for the initialization are computed with (9, 11).

4.3 A Class of Multi-mode Multi-body Systems Satisfying Assumptions 1 and 2

In this section we exhibit a practically useful class of systems satisfying Assumptions 1 and 2. We consider a multi-body system whose model at each mode i has the following structure:

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{v} && (m_1) \\ \boldsymbol{M}(\boldsymbol{q},t)\dot{\boldsymbol{v}} + \boldsymbol{G}_i^T(\boldsymbol{q},t)\boldsymbol{\lambda}_i &= \boldsymbol{h}(\boldsymbol{q},\boldsymbol{v},t) && (m_2) \\ \boldsymbol{0} &= \boldsymbol{g}_i(\boldsymbol{q},t) && (m_3) \end{aligned} \tag{12}$$

$$\text{where } \boldsymbol{M} = \boldsymbol{M}^T \text{ is positive definite, and } \boldsymbol{G}_i = \frac{\partial \boldsymbol{g}_i}{\partial \boldsymbol{q}} \text{ has full row rank.} \tag{13}$$

Equation (12) describes a multi-body system with generalized position coordinates $\boldsymbol{q}$, generalized velocity coordinates $\boldsymbol{v}$ and generalized constraint forces $\boldsymbol{\lambda}_i$ due to the constraints (m_3) of the ith mode. Both the constraints (m_3) and the constraint forces $\boldsymbol{G}_i^T(\boldsymbol{q},t)\boldsymbol{\lambda}_i$ can vary with the mode i. In particular the constraints can also be completely removed. $\boldsymbol{M}$, however, remains invariant through the different modes. Whenever a set of constraints is changing at a new mode, impulses might occur. In case an impulse is due to an impact, it is assumed that the impact is completely inelastic. We now show that Eq. (12) can be put to a form satisfying Assumptions 1 and 2.

Equation (12) is first transformed to an index *two* DAE with the method of Gear et al. [15]:

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{v} - \boldsymbol{G}_i^T(\boldsymbol{q},t)\boldsymbol{\mu}_i && (m_1^a) \\ \boldsymbol{M}(\boldsymbol{q},t)\dot{\boldsymbol{v}} + \boldsymbol{G}_i^T(\boldsymbol{q},t)\boldsymbol{\lambda}_i &= \boldsymbol{h}(\boldsymbol{q},\boldsymbol{v},t) && (m_2) \\ \boldsymbol{0} &= \boldsymbol{g}_i(\boldsymbol{q},t) && (m_3) \\ \boldsymbol{0} &= \boldsymbol{G}_i(\boldsymbol{q},t)\boldsymbol{v} + \boldsymbol{g}_i^{(1)}(\boldsymbol{q},t) && (\dot{m}_3) \end{aligned} \tag{14}$$

$$\text{with } \boldsymbol{g}_i^{(1)}(\boldsymbol{q},t) = \frac{\partial \boldsymbol{g}_i}{\partial t}\,. \tag{15}$$

In Eq. (14) the constraint equation (m_3) is differentiated once and new auxiliary variables $\boldsymbol{\mu}_i$ are introduced so that the number of equations and unknowns remain the same. (14) has the same solution as (12) because $\boldsymbol{\mu}_i = \boldsymbol{0}$ holds as we explain now: Inserting (m_1^a) in $(\dot{m}_3)$ yields:

$$\boldsymbol{0} = \boldsymbol{G}_i(\dot{\boldsymbol{q}} - \boldsymbol{G}_i^T\boldsymbol{\mu}_i) + \boldsymbol{G}_i^{(1)} \tag{16}$$

Subtracting the derivative of (m_3) from (16) results in equation $\boldsymbol{0} = \boldsymbol{G}_i\boldsymbol{G}_i^T\boldsymbol{\mu}_i$. Since $\boldsymbol{G}_i$ has full row rank according to assumption (13), $\boldsymbol{G}_i\boldsymbol{G}_i^T$ is regular and therefore $\boldsymbol{\mu}_i = \boldsymbol{0}$.

As proposed by Gear [14], by using the substitutions ($\boldsymbol{\lambda}_{int,i}$ is the integral of $\boldsymbol{\lambda}_i$ and $\boldsymbol{\mu}_{int,i}$ is the integral of $\boldsymbol{\mu}_i$):

$$\dot{\boldsymbol{\lambda}}_{int,i} = \boldsymbol{\lambda}_i, \quad \dot{\boldsymbol{\mu}}_{int,i} = \boldsymbol{\mu}_i \tag{17}$$

the DAE (14) which is index 2 in the variables $\boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\lambda}_i, \boldsymbol{\mu}_i$ is transformed to the following DAE which is index 1 in the variables $\boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\lambda}_{int,i}, \boldsymbol{\mu}_{int,i}$:

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{v} - \boldsymbol{G}_i^T(\boldsymbol{q},t)\dot{\boldsymbol{\mu}}_{int,i} && (m_1^b) \\ \boldsymbol{M}(\boldsymbol{q},t)\dot{\boldsymbol{v}} + \boldsymbol{G}_i^T(\boldsymbol{q},t)\dot{\boldsymbol{\lambda}}_{int,i} &= \boldsymbol{h}(\boldsymbol{q},\boldsymbol{v},t) && (m_2^b) \\ \boldsymbol{0} &= \boldsymbol{g}_i(\boldsymbol{q},t) && (m_3) \\ \boldsymbol{0} &= \boldsymbol{G}_i(\boldsymbol{q},t)\boldsymbol{v} + \boldsymbol{g}_i^{(1)}(\boldsymbol{q},t) && (\dot{m}_3) \end{aligned} \tag{18}$$

(18) has equation structure (7), with

$$\begin{aligned} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{v} \\ \boldsymbol{\lambda}_{int,i} \\ \boldsymbol{\mu}_{int,i} \end{bmatrix} \quad \boldsymbol{A} &= \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{G}_i^T(\boldsymbol{q},t) \\ \boldsymbol{0} & \boldsymbol{M}(\boldsymbol{q},t) & \boldsymbol{G}_i^T(\boldsymbol{q},t) & \boldsymbol{0} \end{bmatrix} \\ \boldsymbol{b} = \begin{bmatrix} -\boldsymbol{v} \\ -\boldsymbol{h}(\boldsymbol{q},\boldsymbol{v},t) \end{bmatrix} \quad \boldsymbol{f}_c &= \begin{bmatrix} \boldsymbol{g}_i(\boldsymbol{q},t) \\ \boldsymbol{G}_i(\boldsymbol{q},t)\boldsymbol{v} + \boldsymbol{g}_i^{(1)}(\boldsymbol{q},t) \end{bmatrix}. \end{aligned} \tag{19}$$

The Jacobian (3) of (18) is ($\boldsymbol{P}$ is a permutation matrix to exchange (m_3) and ($\dot{m}_3$) in order that the regularity of the Jacobian is at once visible):

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial \boldsymbol{f}_d}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial \boldsymbol{f}_c}{\partial \boldsymbol{x}} \end{bmatrix} = \boldsymbol{P} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{G}_i^T \\ \boldsymbol{0} & \boldsymbol{M} & \boldsymbol{G}_i^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{G}_i & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{G}_i & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{20}$$

As required by Assumption 1, this Jacobian is regular because $\boldsymbol{M}$ is positive definite and $\boldsymbol{G}_i$ has full row rank. With reference to Assumption 2, it remains to show that it is legitimate to take $\boldsymbol{x}_s = \boldsymbol{q}$, which amounts to requiring that $\boldsymbol{q}$ is continuous and has bounded variation around the instant of mode change. Before an impact occurs, the normal distance d_j to a constraint surface j is defined as $d_j = g_j(\boldsymbol{q},t)$ and $d_j > 0$ when the multi-body system is not in contact with the constraint surface. Contact occurs when $d_j = 0$ and therefore the constraint $g_j(\boldsymbol{q},t) = 0$ at the time instant of the impact and $\boldsymbol{q}$ is continuous. In the fully inelastic impact case, $\boldsymbol{q}$ has bounded variation because the multi-body system remains in contact with the contact surface after the impact and it is assumed that $g_j(\boldsymbol{q},t)$ is smooth. The scheme above can be easily extended to elastic impacts. In such a case, $\boldsymbol{q}$ has bounded variation provided instants of mode changes do not form a Zeno sequence.

4.4 Example: Ideal Clutch with Motor

We will show in this section how the example of Fig. 3 can be simulated with the method developed in the previous sections. The modular description with Modia is mapped to a set of about 25 equations that are first pre-processed (such as performing alias elimination). To avoid overloading the development with unnecessary details, we only show the results after this pre-processing is

performed (these equations can also be easily derived manually from the circuit). In the following model, C, k, J, R are constants; $u_0(t), \gamma(t)$ are given time functions; $\gamma = \mathtt{T}$ if the clutch is released and $\mathtt{F}$ if it is engaged:

$$\begin{aligned} R(i_1 + i_2) + u &= u_0 && \text{(sum of voltages in left loop)} \\ C\dot{u} &= i_1 && \text{(capacitor)} \\ k\omega_1 &= u && \text{(emf)} \\ \tau_1 &= ki_2 && \text{(emf)} \\ J_1\dot{\omega}_1 &= \tau_1 - \tau_2 && \text{(inertia1)} \\ J_2\dot{\omega}_2 &= \tau_2 && \text{(inertia2)} \\ \textbf{if } \gamma \textbf{ then } \tau_2 \textbf{ else } \omega_1 - \omega_2 &= 0 && \text{(idealized clutch)} \end{aligned} \tag{21}$$

All the following analysis could be performed with (21). To concentrate on the essential details, the equations are further simplified by using the following relationships from (21), as well as the substitution $\dot{\tau}_{int,2} = \tau_2$

$$\begin{aligned} \tau_1 &:= J_1\dot{\omega}_1 + \tau_2 \\ i_2 &:= \tau_1/k \;=\; (J_1\dot{\omega}_1 + \dot{\tau}_{int,2})/k \\ i_1 &:= C\dot{u} \end{aligned} \tag{22}$$

and the equation system (21) can be simplified to the following four equations:

$$\begin{aligned} J_1\dot{\omega}_1 + \dot{\tau}_{int,2} + kC\dot{u} - k(u_0 - u)/R &= 0 \\ J_2\dot{\omega}_2 - \dot{\tau}_{int,2} &= 0 \\ \textbf{if } \gamma \textbf{ then } \dot{\tau}_{int,2} \textbf{ else } \omega_1 - \omega_2 &= 0 \\ k\omega_1 - u &= 0 \end{aligned} \tag{23}$$

With $\boldsymbol{x} = [u\,;\,\omega_1\,;\,\omega_2\,;\,\tau_{int,2}]^T$, (23) is in both modes an index one DAE of the form (7):

$$\begin{aligned} &\gamma = \mathtt{T} \text{ (mode 1)}: \\ &\quad \boldsymbol{A}_1 = \begin{bmatrix} kC & J_1 & 0 & +1 \\ 0 & 0 & J_2 & -1 \\ 0 & 0 & 0 & +1 \end{bmatrix} \quad \boldsymbol{b}_1 = \begin{bmatrix} -k(u_0 - u)/R \\ 0 \\ 0 \end{bmatrix} \quad \boldsymbol{f}_{c,1} = k\omega_1 - u \\ &\gamma = \mathtt{F} \text{ (mode 2)}: \\ &\quad \boldsymbol{A}_2 = \begin{bmatrix} kC & J_1 & 0 & +1 \\ 0 & 0 & J_2 & -1 \end{bmatrix} \quad \boldsymbol{b}_2 = \begin{bmatrix} -k(u_0 - u)/R \\ 0 \end{bmatrix} \quad \boldsymbol{f}_{c,2} = \begin{bmatrix} k\omega_1 - u \\ \omega_1 - \omega_2 \end{bmatrix} \end{aligned} \tag{24}$$

With (8), or alternatively with (9) and $h \approx 0$, restart values can be computed:

$$\begin{aligned} &\gamma = \mathtt{F} \to \mathtt{T} \;\; (\text{assuming } \boldsymbol{x}^- \text{ satisfies mode 2 with } \boldsymbol{A}_2, \boldsymbol{b}_2, \boldsymbol{f}_{c,2}): \\ &\qquad \omega_1^+ = \omega_1^-, \quad \omega_2^+ = \omega_2^-, \quad u^+ = u^- \\ &\gamma = \mathtt{T} \to \mathtt{F} \;\; (\text{assuming } \boldsymbol{x}^- \text{ satisfies mode 1 with } \boldsymbol{A}_1, \boldsymbol{b}_1, \boldsymbol{f}_{c,1}): \\ &\qquad \omega_1^+ = \frac{J_1\omega_1^- + J_2\omega_2^- + kCu^-}{J_1 + J_2 + k^2C}, \quad \omega_2^+ = \omega_1^+, \quad u^+ = k\omega_1^+ \end{aligned} \tag{25}$$

When the clutch is disengaging, the variables are continuous at the mode change. When the clutch is engaging, the variables are discontinuous at the mode change and Dirac impulses occur.

4.5 Implementation of Multi-mode Features in Modia

The implementation of Modia is described in [12]. An extension of Modia to support a restricted class of multi-mode DAE systems along the lines developed in this section is currently under development. Since completely different sets of equations can be present in different modes, the implementation is based on just-in-time symbolic transformations and code generation of residue functions. A dictionary from a Boolean vector of mode flags to functions is used as a cache to avoid symbolic transformations and code generation in case the same modes have been active before and a corresponding function is already available to calculate the residues.

The final values of all variables from one simulation is extracted when a mode change event happens. These are inputs to one very short implicit Euler step with the residue function for the newly enabled modes in order to correctly simulate possible impulses. The new values of the state vector is used to start a new simulation until the next mode change.

5 Structural Analysis of Multi-Mode DAE Systems

In Sect. 4 we proposed an approach to analyze and simulate multi-mode DAE systems based on a generalization of DAE theory.

In this section we propose an alternative approach, more computer science oriented and detailed in [3] (as well as its companion technical report [5]), which works for general multi-mode systems and uses a small number of principles. The key ideas are as follows.

1. Mode changes may result in discontinuous jumps and, therefore, resets must be performed in discrete computation steps. Hence, we first map the original system to discrete time by using a first order Euler scheme. This brings to discrete time both the reset actions and the dynamics within each mode, hence the principles of index analysis uniformly apply, albeit *in discrete time*. Also, a new principle of *mode causality* is invoked.
2. Mapping the dynamics to discrete time results in approximations. No approximation, however, results if we interpret the Euler scheme via *nonstandard analysis* [18,28] by using an *infinitesimal* time step. The analysis is then performed over the nonstandard reals. A final *standardization* step is applied to recover an effective numerical scheme.

To keep the exposure simple, we develop this on the example of ideal clutch with motor, see Sect. 4.4. More precisely, we consider again (23) where, for simplicity, we substitute $\dot{\tau}_{int,2}$ by τ:

$$\begin{aligned} J_1\dot{\omega}_1 + \tau + kC\dot{u} - k(u_0 - u)/R &= 0 \\ J_2\dot{\omega}_2 - \tau &= 0 \\ k\omega_1 - u &= 0 \\ \textbf{if } \gamma \textbf{ then } \tau \textbf{ else } \omega_1 - \omega_2 &= 0 \end{aligned} \tag{26}$$

We first analyze separately the model for each mode of the clutch.

5.1 Separate Analysis of Each Mode, in Discrete Time

We begin by providing the model for each mode. We highlight in blue the equations that are unique to the considered mode. Other equations are shared. In the "released" mode, the two shafts are independent. In the "engaged" mode, the velocities of the two shafts are algebraically related.

Following key idea 1, we replace derivatives by their first order explicit Euler scheme, in discrete time with constant step size $\delta > 0$. Let

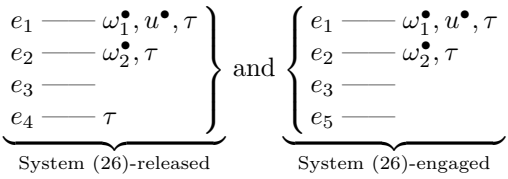
$$\omega^\bullet(t) =_{\text{def}} \omega(t+\delta) \tag{27}$$

be the forward time shift operator by an amount of δ. System (26) expands as:

$$\underbrace{\left.\begin{array}{ll}(e_1): & 0 = J_1(\omega_1^\bullet - \omega_1) + kC(u^\bullet - u) \\ & \quad - \delta.(\tau - k(u_0 - u)/R) \\ (e_2): & 0 = J_2(\omega_2^\bullet - \omega_2) - \delta.\tau \\ (e_3): & 0 = k\omega_1 - u \\ (e_4): & 0 = \tau \end{array}\right\}}_{\text{System (26)-released}} \text{ and } \underbrace{\left\{\begin{array}{ll}(e_1): & 0 = J_1(\omega_1^\bullet - \omega_1) + kC(u^\bullet - u) \\ & \quad - \delta.(\tau - k(u_0 - u)/R) \\ (e_2): & 0 = J_2(\omega_2^\bullet - \omega_2) - \delta.\tau \\ (e_3): & 0 = k\omega_1 - u \\ (e_5): & 0 = \omega_1 - \omega_2 \end{array}\right.}_{\text{System (26)-engaged}} \tag{28}$$

Let us first focus on System (26)-released. The state variables are u, ω_1, ω_2 and their respective values are known initially. The current step must determine the values of the leading variables $\tau, u^\bullet, \omega_1^\bullet, \omega_2^\bullet$.

Towards this, we first form the incidence graph $\mathcal{G}$ of each system, which is a nondirected bipartite graph having as vertices: the four leading variables, plus the two systems of equations $\{(e_1), (e_2), (e_3), (e_4)\}$ and $\{(e_1), (e_2), (e_3), (e_5)\}$. An edge from an equation to a leading variable exists if and only if this variable is involved in that equation. Incidence graphs for models (26-released) and (26-engaged) are:



Observe that, for both models, equation (e_3) involves no leading variable: it is a *consistency* equation, i.e., a constraint that must be satisfied as a result of the execution of previous time steps. Once initialization is performed, (e_3) is indeed satisfied and can thus be seen as a fact for both modes. The same holds for (e_5) in the engaged mode. In turn these equations cannot be used, when determining the leading variables from the state variables. In this case, for the two systems (26), we have 4 variables but only 3 and 2 equations, respectively: one cannot determine the leading variables as functions of the state variables by using System (26). Since the considered models are time-invariant, every solution has also satisfy the equations obtained by shifting forward any equation of the model. Shifting forward equations that do not bring variables that are shifted

more than originally present in the system, yields so-called *latent equations*. For our two models, we add the latent equations, highlighted in blue:

$$\underbrace{\left.\begin{array}{l}(e_1): 0 = J_1(\omega_1^\bullet - \omega_1) + kC(u^\bullet - u) \\ \qquad\qquad - \delta.(\tau - k(u_0 - u)/R) \\ (e_2): 0 = J_2(\omega_2^\bullet - \omega_2) - \delta.\tau \\ (e_3): 0 = k\omega_1 - u \\ (e_3^\bullet): 0 = k\omega_1^\bullet - u^\bullet \\ (e_4): 0 = \tau \end{array}\right\}}_{\text{Eq. (29)-released}} \text{ and } \underbrace{\left\{\begin{array}{l}(e_1): 0 = J_1(\omega_1^\bullet - \omega_1) + kC(u^\bullet - u) \\ \qquad\qquad - \delta.(\tau - k(u_0 - u)/R) \\ (e_2): 0 = J_2(\omega_2^\bullet - \omega_2) - \delta.\tau \\ (e_3): 0 = k\omega_1 - u \\ (e_3^\bullet): 0 = k\omega_1^\bullet - u^\bullet \\ (e_5): 0 = \omega_1 - \omega_2 \\ (e_5^\bullet): 0 = \omega_1^\bullet - \omega_2^\bullet \end{array}\right.}_{\text{Eq. (29)-engaged}} \tag{29}$$

The associated incidence graphs are augmented accordingly and we remove the consistency equations:

$$\underbrace{\left.\begin{array}{ll} e_1 & \omega_1^\bullet, u^\bullet, \tau \\ e_2 & \omega_2^\bullet, \tau \\ e_3^\bullet & \omega_1^\bullet, u^\bullet \\ e_4 & \tau \end{array}\right\}}_{\text{Eq. (29)-released}} \text{ and } \underbrace{\left\{\begin{array}{ll} e_1 & \omega_1^\bullet, u^\bullet, \tau \\ e_2 & \omega_2^\bullet, \tau \\ e_3^\bullet & \omega_1^\bullet, u^\bullet \\ e_5^\bullet & \omega_1^\bullet, \omega_2^\bullet \end{array}\right.}_{\text{Eq. (29)-engaged}} \tag{30}$$

In the two incidence graphs of (30), we show in red a *pairing function* $\mathbf{e}$, i.e., a bijection, from the set of variables, to the set of equations, such that $(\mathbf{e}(x), x)$ is a edge of $\mathcal{G}$ for every leading variables $\boldsymbol{x}$. This pairing function defines an orientation for $\mathcal{G}$ as follows: for each edge $(\mathbf{e}(x), x) \in \mathcal{G}$, we set $\mathbf{e}(x) \to x$ and $y \to \mathbf{e}(x)$ for every $y \neq x$ such that $(\mathbf{e}(x), y)$ is a edge of $\mathcal{G}$. The minimal cycles of the resulting directed graph form blocks of equations to be solved for their assigned variables. The blocks are partially ordered by the directed graph. According to [23], if a pairing function exists for its incidence graph, the system of equations possesses, in a generic sense, a unique solution for its leading variables, assuming consistent values for the state variables. "Generic" here means that the statement holds outside some exceptional numerical values when the non-zero coefficients of the Jacobian of this system of equations vary in a neighborhood.

The above reasoning applies in continuous time where the forward shift is substituted back using differentiation. As far as structural analysis is concerned, we can freely exchange continuous and discrete time using the correspondence $\dot{\omega} \leftrightarrow \frac{\omega^\bullet - \omega}{\delta}$.

5.2 Global Discrete-Time Analysis

In this section, we handle each of the two modes as well as the mode changes in a uniform way. Let's consider now the discretized version of our two modes System (26):

$$\begin{array}{ll} (e_1): & 0 = J_1(\omega_1^\bullet - \omega_1) + kC(u^\bullet - u) - \delta.(\tau - k(u_0 - u)/R) \\ (e_2): & 0 = J_2(\omega_2^\bullet - \omega_2) - \delta.\tau \\ (e_3): & 0 = k\omega_1 - u \\ (e_4): & \texttt{if } \gamma \texttt{ do } 0 = \tau \\ (e_5): & \texttt{if not } \gamma \texttt{ do } 0 = \omega_1 - \omega_2 \end{array} \tag{31}$$

Notice how model (31) encompasses both the released and engaged modes, as well as the mode changes, in the same uniform setting of discrete time systems. This calls for using similar reasoning to determine the new values of the leading variables regardless of whether the system is evolving in continuous mode or is at an event of mode change. We now detail the how and when the different involved equations are used.

1. Using $(e_1), \ldots, (e_3)$ Those three equations are active in all modes. Equation (e_3) is useless, so we are left with 2 equations and 4 leading variables and no subset of them can be evaluated using the 3 available equations only. We thus have to evaluate the guard γ in order to know which equation among (e_4) or (e_5) is active. We successively analyze the two cases below.

2-released. Case $\gamma = \text{T}$ Equation (e_4) is enabled and equation (e_5) is disabled. The reasoning proceeds exactly as for getting the model (29-released). The difference is that we take the consistency equation (e_3) as an assumption, since its satisfaction results from the execution of the previous time step. The resulting model is thus:

$$\begin{array}{l} \texttt{assuming } : \ (e_3): \ 0 = k\omega_1 - u \\ \texttt{if } \gamma = \text{T} \texttt{ do } (\text{29-released} \setminus \{(e_3)\}) \end{array} \tag{32}$$

where $(\text{29-released} \setminus \{(e_3)\})$ means that equation (e_3) is removed from system 29-released. Note that model (29-released) applies both within the "released" mode and at the instant of mode change $\gamma : \text{F} \to \text{T}$.

2-engaged. Case $\gamma = \text{F}$ Equation (e_4) is disabled and equation (e_5) is enabled. With reference to system (29-engaged), an important difference occurs: assuming that the consistency equation (e_5) results from having executed the previous time step is not always valid (equation (e_5) is guarded and can thus be disabled). Consequently the following two sub-cases need to be considered:

Case $\omega_1 = \omega_2$ *follows from previous time step.* This corresponds to the case in which the system was already in mode "engaged" at the previous time step. The separate analysis developed for the "engaged" mode in Sect. 5.1 applies with no change. The model, as augmented with the two latent equations $(e_3^\bullet, e_5^\bullet)$ is (29-engaged), in which (e_3, e_5) are taken as assumptions:

$$\begin{array}{l} \texttt{assuming } : \ \begin{cases} (e_3): \ 0 = k\omega_1 - u \\ (e_5): \ 0 = \omega_1 - \omega_2 \end{cases} \\ \texttt{if } \gamma = \text{F} \texttt{ do } (\text{29-engaged} \setminus \{(e_3), (e_5)\}) \end{array} \tag{33}$$

Case $\omega_1 = \omega_2$ *does not follow from previous time step.* This arises if the system is engaged at the current instant t, but was released at the immediate previous

time step, $t-\delta$, i.e., t is an instant of mode change $\gamma : \text{T} \to \text{F}$. This yields a new situation, not seen in Sect. 5.1. The engaged mode requires the consistency equation $\omega_1 = \omega_2$, whereas $\omega_1(t)$ and $\omega_2(t)$ were both evaluated at the previous time step $t-\delta$, at which $\omega_1^\bullet(t-\delta) = \omega_2^\bullet(t-\delta)$ was *not* enforced. As a consequence, equation (e_5) of System (31) is enabled. Unfortunately, (e_5) possesses no dependent variable and the values of the state variables ω_1 and ω_2 were set at previous time step $t-\delta$, with no guarantee that $\omega_1(t) = \omega_2(t)$ would result. System (31) is thus overdetermined at time t. A first idea would be to reject this kind of model. This would be unfortunate as our original model (26) was natural for our electromechanical system. To overcome this issue, we invoke the following

Principle 1 (Mode Causality Principle). *A guard must be evaluated before the equation it controls.*

This principle leads to *shifting forward* equation (e_5) in model (29-engaged). We expect this modification to be very mild since it consists in delaying the satisfaction of the constraint by a small amount of time. Performing this yields:

$$\begin{array}{l} \texttt{assuming} \ : \ (e_3) : \ 0 = k\omega_1 - u \\ \ \texttt{if} \ \gamma = \text{F} \ \texttt{do} \ (\text{29-engaged} \setminus \{(e_3), (e_5)\}) \end{array} \tag{34}$$

Systems (33) and (34) possess identical right hand sides but were obtained by different reasoning—the fact that identical right hand sides were obtained is incidental to this example. System (32, 33, 34) replaces the original System (31).

As a final remark, observe that the same analysis would work without changes if the guard γ was a predicate in the state variables u, ω_1, ω_2.

The corresponding complete execution scheme is shown in Fig. 5. In this figure, boxes are the *states* of the execution scheme and their content specifies the configuration of guard, variables, and equations. For the guard and the variables: v (resp. $\overline{v}$) means $v = \text{T}$ (resp. $v = \text{F}$). For equations, e (resp. $\overline{e}$) means that e is active (resp. disabled) and $\sharp e$ means that the body of e is assumed from previous time step. Not mentioning a variable or an equation in a box means that this variable is not evaluated yet and this equation is not solved yet; for shifted equations added by the algorithm, however, we mention them underlined. The *transitions* of the state machine indicate the actions performed when moving to the next state. FS(e) indicates that e is shifted. PR(e) indicates that e is known to be satisfied. LE($e^\bullet$) indicates that we add $e^\bullet$ as a latent equation. Blue (resp. black) transitions belong to a continuous-time (resp. discrete-time) dynamics. The red transition is impulsive. A semicolon is the sequential composition of computations, and the + sign denotes enabled blocks of equations, ready to be solved. The following comments are in order.

1. Observe first the parallel between the models sitting in the boxes of the diagram of Fig. 5 on the one hand, and the mixed explicit/implicit scheme (9) on the other hand. For this comparison, variables with superscript "+" in (9) correspond to shifted variables in Fig. 5 and variables with superscript "−" in (9) correspond to non shifted variables in Fig. 5.

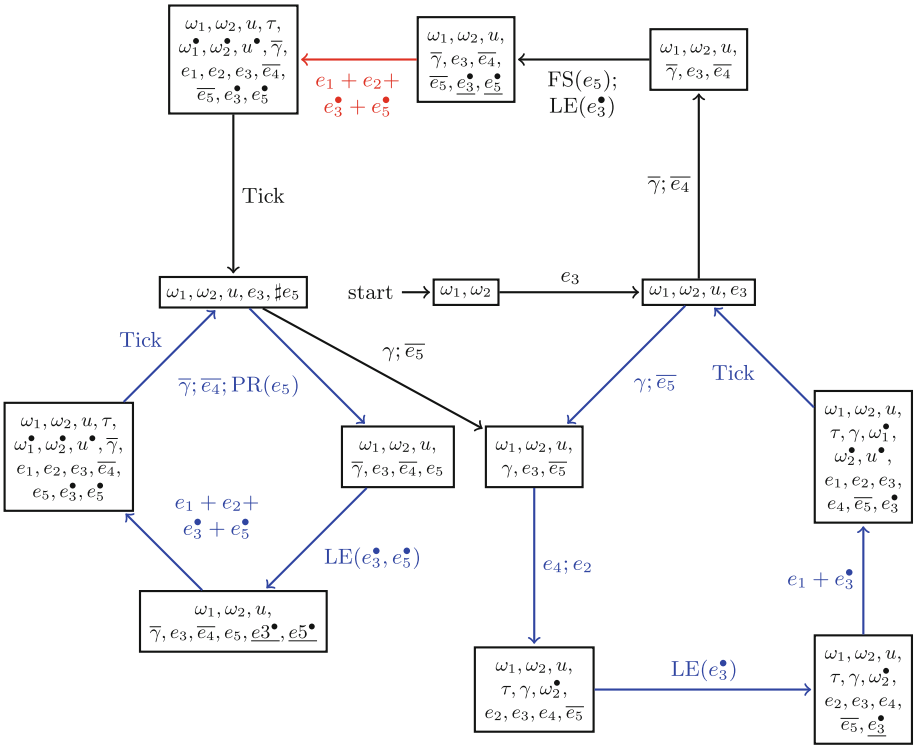


Fig. 5. Structural analysis of System (31): state machine describing the execution scheme of one time step of System (31).

2. Our development relies on a small set of principles:
   - We map the continuous time multi-mode System (26) to discrete time System (31) by mapping derivatives $\dot{x}$ to their explicit Euler scheme $\frac{x^\bullet - x}{\delta}$.
   - Our massaging of the equations only depends on the values taken by the guards, and the assumption regarding the satisfaction/violation of the consistency equations (here (e_3), and (e_5) for the engaged mode). Otherwise, we make no distinction between instants of mode changes and other instants: our treatment is uniform.
   - Our massaging of the equations has two objectives: finding latent equations if needed, and shifting forward equations when required by the principle of mode causality (Principle 1).

   This set of principles is small, clean, and powerful enough to encompass general systems with a structural analysis alike the one we developed here for the clutch example. See [3] for a presentation of this approach for general multi-mode DAE systems.

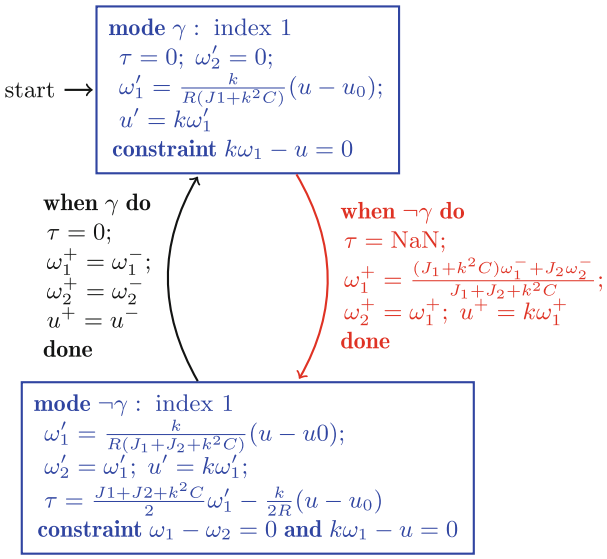


Fig. 6. Actual simulation code for the clutch.

5.3 Effective Simulation Code

So far the execution scheme of Fig. 5 is not satisfactory, as it uses an explicit first order Euler scheme. From the execution scheme of Fig. 5, however, the actual simulation code shown on Fig. 6 is derived. Observe that it consists of

- a DAE model of index zero or one for each mode, and
- code for resetting the state variables at mode changes.

The model for each mode can be simulated, e.g., using a BDF method as proposed in Sect. 4.2. Regarding the resets at mode changes, ω_i^- is the previous value of state variable ω_i, which is the left limit of ω_i when exiting a mode. Similarly, ω_i^+ is the reset value for state variable ω_i when entering the new mode. Continuous-time dynamics are colored blue; non-impulsive (resp. impulsive) resets are colored black (resp. red). The dynamics in each mode are defined by an over-determined index-1 DAE system consisting of an ODE system coupled to an algebraic constraint. In the transition from mode $\overline{\gamma}$ to mode γ, variable τ is impulsive, and its value is not computed—it is set to NaN (Not a Number). Let us explain how the code of Fig. 6 is deduced from the execution scheme of Fig. 5.

Nonstandard Analysis. First, we select the step size δ to be infinitesimal in the sense of nonstandard analysis [18,28], of which we informally recall the background we need.

In nonstandard analysis, the set $\mathbb{R}$ of real numbers is extended with infinite numbers, which are larger than any real number, and infinitesimal numbers,

which are smaller (in absolute value) than any nonzero real number. The resulting set ${}^\star\mathbb{R}$ is an extension of $\mathbb{R}$ that keeps its basic properties. In particular, it is an ordered field. We will be writing $x \approx y$ if $x - y$ is an infinitesimal. Any finite element $z \in {}^\star\mathbb{R}$ has a standard part, defined as the unique real number $st(z)$ such that $z \approx st(z)$. Any element of $\mathbb{R}$ is called standard.

If $\boldsymbol{x}(t)$ is a standard differentiable function of time, its derivative $\dot{\boldsymbol{x}}(t)$ satisfies the property that, for any infinitesimal nonzero time step δ,

$$\dot{\boldsymbol{x}}(t) \approx \frac{\boldsymbol{x}(t+\delta) - \boldsymbol{x}(t)}{\delta}. \tag{35}$$

That is, taking an explicit first order Euler scheme with infinitesimal step size yields an exact match for an ODE, up to an infinitesimal error. Formally, one says that this Euler scheme *standardizes* as the solution of the ODE.

DAE Model for Each Mode. Each (blue) mode in Fig. 6 corresponds to a blue cycle in Fig. 5. For instance, the equation

$$\begin{aligned} (e_2): \quad & 0 = J_2(\omega_2^\bullet - \omega_2) - \delta.\tau \\ \text{becomes } (e_2): \quad & 0 = J_2 \tfrac{\omega_2^\bullet - \omega_2}{\delta} - \tau, \end{aligned}$$

which, by Eq. (35), standardizes as the ODE $(e_2): 0 = J_2\dot{\omega}_2 - \tau$. The reader is referred to [3] for details about standardization.

Computing the Reset Values. As in the development of Sect. 4, one key contribution of our approach is the reset code for the mode transitions. Let us now explain how this part of Fig. 6 is derived from Fig. 5. Here the reasoning is different since we do not target a continuous time dynamics, but rather a finite sequence of discrete time steps implementing the reset actions.

Let's focus on the transition $\gamma : \text{T} \to \text{F}$ shown in red in Fig. 6, which originates from the path sitting at the top, from the right- to the left blue cycle in Fig. 5. The corresponding dynamics is (34) and we target a discrete time dynamics involving the forward shift $^\bullet$ and no differentiation. Hence, in Eq. (34), equations $(e_3^\bullet), (e_5^\bullet)$ have the convenient form. In turn, one can no longer interpret (e_1) or (e_2) as differential equations. We must rather regard them as difference equations. Since they involve the infinitesimal parameter δ, standardization must be performed with care. Indeed, due to the equation $(e_5^\bullet)$ of (34) and since (e_5) is not assumed, the velocities ω_i experience a discrete jump. By (e_1) and (e_2) and since δ is infinitesimal, we infer that τ must be impulsive. This propagates throughout the different equations of the system. For impulses, the *rescaling*

$$\tau' =_{\text{def}} \delta.\tau$$

of the system variables is applied, guided by the equations and the incidence graph (30-right) of the system, see the *impulse analysis* developed in [3]. Using this rescaling, the block of equations of (34) becomes

$$\begin{array}{ll} (e_1): & 0 = J_1(\omega_1^\bullet - \omega_1) + kC(u^\bullet - u) - \tau' + \delta.k(u_0 - u)/R \\ (e_2): & 0 = J_2(\omega_2^\bullet - \omega_2) - \tau' \\ (e_3^\bullet): & 0 = k\omega_1^\bullet - u^\bullet \\ (e_5^\bullet): & 0 = \omega_1^\bullet - \omega_2^\bullet \end{array} \tag{36}$$

The term shown in blue involves state variables and is multiplied by the infinitesimal δ. Zeroing this blue term in System (36) leaves us with a *structurally regular* system that we can solve for its dependent variables $\tau', u^\bullet, \omega_1^\bullet, \omega_2^\bullet$. Solving (36) yields in particular the reset values for the state variables $u^+ =_{\text{def}} u^\bullet$, and $\omega_i^+ =_{\text{def}} \omega_1^\bullet = \omega_2^\bullet$. We recover in particular the formulas (25) from Sect. 4.4.

5.4 Constructive Semantics

The essential step in getting the final code of Fig. 6 was the construction of the state machine of Fig. 5. This state machine is called the *Constructive Semantics* of the original System (31). The notion of constructive semantics was first introduced in the context of reactive synchronous programming languages [4,6,7], where it played an important role in grounding compilation on solid mathematical foundations. Essentially, a constructive semantics for a discrete time dynamical system consists of:

1. A specification of the set of *atomic actions*, which are effective, non interruptible, state transformation operations. Executing an atomic action is often referred to as performing a *micro-step*;
2. A specification of the correct scheduling of the set of micro-steps constituting a *reaction*, by which discrete time progresses, from the current instant to the next one.

The effect of atomic actions is to propagate knowledge regarding the statuses (*not evaluated*, *evaluated*) and values of variables. The computation of the constructive semantics of a synchronous program may succeed, and then the execution code is generated. Or it may fail, and then the program is rejected. The decision success/failure is formally sound, see [4,6,7].

For synchronous languages, atomic actions are restricted to either (*i*) the evaluation of a single expression, or (*ii*) control flow operations.

In contrast, the class of atomic actions for multi-mode difference algebraic equations systems comprises: (*i*) Evaluating a guard; (*ii*) Solving a block of numerical equations; (*iii*) Equation management operations, such as shifting an equation, adding a latent equation, or changing the status involved/not involved of an equation at a given mode.

In [3] we develop in detail the constructive semantics for multi-mode DAE systems. This allows us to formally define which model can be compiled and then simulated, and which cannot. Models that are under- or overdetermined are rejected. In addition, models which are not handled by the Principle 1 of mode causality, are rejected as well. This approach is implemented in the SunDAE proof of concept tool.

The standardization of the constructive semantics remains open in part although its main principles have been clarified. The Standardization, however, requires symbolic processing related to computer algebra.

6 Challenges in DAE Based Modeling Languages

DAE based modeling languages are essential to the design of CPS. The development of such languages raise a number of challenges.

The correct simulation of mode changes and the need for resetting state variables is a first—mostly open—difficulty, particularly when impulses occur. In this paper, we proposed two different approaches for addressing this issue. The first approach relies on a transformation of the system model to a special index one form, followed by the application of a new formula for resetting the state variables. Arguments supporting this formula were given under additional assumptions on the model. The second approach builds on the use of nonstandard analysis, combined with the heritage of synchronous languages in computer science, particularly on the concept of constructive semantics. The classes of accepted/rejected models are well defined and simulation code is always produced for accepted models. In turn, the physical interpretation is understood only for restricted classes of models.

A particular difficulty of DAE based modeling languages is the need for sophisticated symbolic preprocessing of the model, called structural analysis. The Pantelides algorithm for computing the latent equations of a (single-mode) DAE model gave birth to a large body of literature since 1988. Our paper shows that structural analyses are also essential to handle mode changes.

Structural analyses play also a central role in scaling-up to huge models involving millions of equations. The community of High Performance Computing has provided important related contributions, for single-mode DAE systems. Yet, modularity in compilation and simulation remains open.

Overall, we see as a grand challenge the development of a DAE based modeling language and tool with the following features: DAE models with a very large number of modes are supported; accepted/rejected models are formally characterized; huge models are supported and can be handled in a modular way. We see the new language Modia and the SunDAE library as good starting points for tackling such challenges.

Acknowledgements. The authors want to thank Toivo Henningsson, Lund, Sweden for the collaboration regarding Julia and the design of Modia, as well as the implementation of instantiation and flattening.

References

1. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.-H.: Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In: Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (eds.) HS 1991-1992. LNCS, vol. 736, pp. 209–229. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-57318-6_30
2. Barela, M.: A complementarity approach to modeling dynamic electric circuits. Ph.D. thesis, University of Iowa (2016)

3. Benveniste, A., Caillaud, B., Elmqvist, H., Ghorbal, K., Otter, M., Pouzet, M.: Structural analysis of multi-mode DAE systems. In: HSCC, pp. 253–263. ACM (2017)
4. Benveniste, A., Caillaud, B., Guernic, P.L.: Compositionality in dataflow synchronous languages: specification and distributed code generation. Inf. Comput. **163**(1), 125–171 (2000)
5. Benveniste,A., Caillaud, B., Pouzet, M., Elmqvist, H., Otter, M.: Structural analysis of multi-mode DAE systems. Research report RR-8933, Inria, July 2016
6. Benveniste, A., Caspi, P., Edwards, S.A., Halbwachs, N., Guernic, P.L., de Simone, R.: The synchronous languages 12 years later. Proc. IEEE **91**(1), 64–83 (2003)
7. Berry, G.: Constructive semantics of Esterel: from theory to practice (abstract). In: Wirsing, M., Nivat, M. (eds.) AMAST 1996. LNCS, vol. 1101, p. 225. Springer, Heidelberg (1996). https://doi.org/10.1007/BFb0014318
8. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: a fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017)
9. Brenan, K.E., Campbell, S.L., Petzold, L.R.: Numerical Solution of Initial Value Problems in Differential-Algebraic Equations. SIAM (1996)
10. Dunford, N., Schwartz, J.: Linear Operators, Part I, General Theory. Wiley-Interscience (1958)
11. Elmqvist, H., Henningsson, T., Otter, M.: Systems modeling and programming in a unified environment based on Julia. In: Margaria, T., Steffen, B. (eds.) ISoLA 2016. LNCS, vol. 9953, pp. 198–217. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47169-3_15
12. Elmqvist, H., Henningsson, T., Otter, M.: Innovations for future Modelica. In: Jiri Kofranek, F.C. (ed.) Proceedings of the 12th International Modelica Conference, May 2017. http://www.ep.liu.se/ecp/132/076/ecp17132693.pdf
13. Elmqvist, H., Mattsson, S.-E., Otter, M.: Modelica extensions for multi-mode DAE systems. In: Tummescheit, H., Arzèn, K.-E. (eds.) Proceedings of the 10th International Modelica Conference, Lund, Sweden. Modelica Association, September 2014. http://www.ep.liu.se/ecp/096/019/ecp14096019.pdf
14. Gear, C.W.: Differential-algebraic equation index transformations. SIAM J. Sci. Stat. Comput. **9**(1), 39–47 (1988)
15. Gear, C.W., Leimkuhler, B., Gupta, G.K.: Automatic integration of euler-lagrange equations with constraints. J. Comput. Appl. Math. **12**, 77–90 (1985)
16. Heemels, W.P.M.H., Camlibel, M.K., Schumacher, J.M.: On the dynamic analysis of piecewise-linear networks. IEEE Trans. Circ. Syst. I: Fundam. Theory Appl. **49**(3), 315–327 (2002)
17. Karnopp, D., Margolis, D., Rosenberg, R.: System Dynamics: A Unified Approach. Wiley, Hoboken (1990)
18. Lindstrøm, T.: An invitation to nonstandard analysis. In: Cutland, N. (ed.) Nonstandard Analysis and its Applications, pp. 1–105. Cambridge University Press, Cambridge (1988)
19. Mattsson, S.-E., Otter, M., Elmqvist, H.: Multi-mode DAE systems with varying index. In: Elmqvist, H., Fritzson, P. (eds.) Proceedings of the 11th International Modelica Conference, Versailles, France. Modelica Association, September 2015. http://www.ep.liu.se/ecp/118/009/ecp1511889.pdf
20. Mehrmann, V., Wunderlich, L.: Hybrid systems of differential-algebraic equations - analysis and numerical solution. J. Process Control **19**(8), 1218–1228 (2009). Special Section on Hybrid Systems: Modeling, Simulation and Optimization

21. Modelica: A unified object-oriented language for systems modeling. Language Specification, Version 3.4. Technical report, Modelica Association, April 2017. https://www.modelica.org/documents/ModelicaSpec34.pdf
22. Otter, M., Elmqvist, H.: Transformation of differential algebraic array equations to index one form. In: Kofranek, J., Casella, F. (eds.) Proceedings of the 12th International Modelica Conference, May 2017. http://www.ep.liu.se/ecp/132/064/ecp17132565.pdf
23. Pantelides, C.: The consistent initialization of differential-algebraic systems. SIAM J. Sci. Stat. Comput. **9**(2), 213–231 (1988)
24. Pepper, P., Mehlhase, A., Höger, C., Scholz, L.: A compositional semantics for Modelica-style variable-structure modeling. In: 4th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools (2011). http://www.ep.liu.se/ecp/056/006/ecp1105606.pdf
25. Pfeiffer, F.: On non-smooth multibody dynamics. Proc. Inst. Mech. Eng. Part K: J. Multi-body Dyn. **226**(2), 147–177 (2012). http://journals.sagepub.com/doi/pdf/10.1177/1464419312438487
26. Pfeiffer, F., Glocker, C.: Multibody Dynamics with Unilateral Contacts. Wiley, Hoboken (2008)
27. Pryce, J.D.: A simple structural analysis method for DAEs. BIT **41**(2), 364–394 (2001)
28. Robinson, A.: Nonstandard Analysis. Princeton Landmarks in Mathematics (1996). ISBN 0-691-04490-2
29. Schoeder, S., Ulbrich, H., Schindler, T.: Discussion of the Gear-Gupta-Leimkuhler method for impacting mechanical systems. Multibody Sys. Dyn. **31**, 477–495 (2013)
30. Campbell, S.L., Gear, C.W.: The index of general nonlinear DAEs. Numer. Math. **72**, 173–196 (1995)
31. Thoma, J.: Introduction to Bond Graphs and Their Applications. Pergamon International Library of Science, Technology, Engineering and Social Studies. Pergamon Press (1975)
32. Trenn, S.: Distributional differential algebraic equations. Ph.D. thesis, Technischen Universität Ilmenau (2009). https://www.db-thueringen.de/servlets/MCRFileNodeServlet/dbt_derivate_00018071/ilm1-2009000207.pdf
33. Zimmer, D.: Equation-based modeling of variable-structure systems. Ph.D. thesis, ETH Zürich, no. 18924 (2010). http://www.inf.ethz.ch/personal/fcellier/PhD/zimmer_phd.pdf