FMI Layered Standard for DAE (FMI-LS-DAE)

Contents

This layered standard on top of FMI 3.0 defines how to exchange dynamic models in differential algebraic equation form.

Although the document refers to version 3.0 of the FMI standard, everything described in this document also applies to all subsequent minor versions. For further information on compatibility, see section Versioning and Layered Standards in the FMI 3.0 specification.

1. Introduction

1.1. Intent of this Document

A differential-algebraic system of equations (DAE) is a system of equations that either contains differential equations and algebraic equations, or is equivalent to such a system [1].

- Avoid the requirement of index reduction inside of FMUs.
  - This may improve accuracy due to better drift handling.
- Avoid local nonlinear equation solvers inside of FMUs.
  - This may improve accuracy and avoid problems with different local and global error tolerances.
- Preserve the sparseness of DAE systems which is lost for the corresponding reduced ODE systems.
  - This may improve the performance by usage of the sparseness.
- Allow connections between constraint FMUs.
  - Connecting reduced ODE FMUs may lead globally to a non-solvable (singular) system but not for unreduced DAE FMUs.

1.2. How to read this Document

1.3. Overview

2. Common Concepts

Mathematical Notation

Table 1. Additional symbols for specific variable types.

Variable	Description
$\mathbf{z}_c$	A vector of floating point continuous-time variables representing the continuous-time algebraic variables.

DAE representation

There are different forms of DAEs.

For simplicity discontinuous states are currently disregarded.

Fully-Implicit DAE

The generic *fully-implicit DAE*, or *implicit DAE*, formulation:

$$F(\dot{z}(t),\, z(t),\, t) = 0$$

where $z(t)$ represents state and additional variables and t is time.

A similar and sometimes more useful representation:

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

with dynamic variables $x(t)$, algebraic variables $y(t)$ and time t.

Note that $x(t)$ is not necessarily the state variables, more on this in section 2.1.1.

Semi-Explicit Index 1 / Hessenberg index 1 DAE

The *semi-explicit index 1*, also called *Hessenberg index 1*, DAE representation

$$\begin{aligned} \dot{x}(t) &= f(x(t),\, y(t),\, t) \\ 0 &= g(x(t),\, y(t),\, t) \end{aligned}$$

with g solvable for y, $det\left(\frac{\partial}{\partial y} g\right) \neq 0$.

Here f is called the *differential equations* and g the *algebraic equations*.

Relationship between fully-implicit and semi-explicit DAE

A fully implicit DAE

$$F(\dot{x}(t),\, x(t),\, t) = 0$$

can always be rewritten as a semi-explicit DAE by introducing a new algebraic variable x'

$$\begin{aligned} \dot{x}(t) &= x'(t) \\ 0 &= F(x'(t),\, x(t),\, t) \end{aligned}$$

Semi-Explicit DAE

The *semi-explicit DAE* representation

$$\begin{aligned} \dot{x}(t) &= f(x(t),\, y(t),\, t) \\ 0 &= g^1(x(t),\, y(t),\, t) \\ 0 &= g^2(x(t),\, y(t),\, t) \\ 0 &= g^3(x(t),\, y(t),\, t) \\ &\vdots \end{aligned}$$

with g^1 solvable for y, $det\left(\frac{\partial}{\partial y} g^1\right) \neq 0$ and $\left\{\frac{d}{dt} g_i^k\right\} \subseteq \{g_i^{k-1}\},\ k>1$.

Mass matrix DAE

$$M\dot{x}(t) = f(x(t), t)$$

where *mass matrix* M is singular.

An equivalent explicit formulation:

$$\begin{bmatrix} M_x & 0 \\ 0 & 0 \end{bmatrix} . \begin{bmatrix} \dot{x} \\ 0 \end{bmatrix} = \begin{bmatrix} \hat{f}(x(t), y(t), t) \\ g(x(t), y(t), t) \end{bmatrix}$$

See [cui2020mass].

Linear time-invariant DAE

$$E\dot{x}(t) = Ax(t)$$

where matrix E is singular.

Indices

There are multiple, sometimes closely related, definitions of indices for DAE systems. Usually these indices are a measure of how far from an ODE the DAE is.

Differential Index

The minimum number of times a DAE

$$F(\dot{x}(t), x(t), t) = 0$$

has to be differentiated with respect to time t to be able to determine $\dot{x}(t)$ as a function of t and x is called the *differential index*.

See [hairer2006numerical].

Perturbation Index

The *perturbation index* is a measure of the constraints among equations. An index-0 DAE contains neither algebraic loops nor structural singularities. An index-1 DAE contains algebraic loops, but no structural singularities. A DAE with a perturbation index greater than 1, a so-called *higher-index DAE*, contains structural singularities [2].

See [hairer2006numerical].

Example of Structural Singularity

[cellier2006continuous, chapter 7.7 Structural Singularity Elimination] provides a simple electric circuit model that is an higher-index DAE.

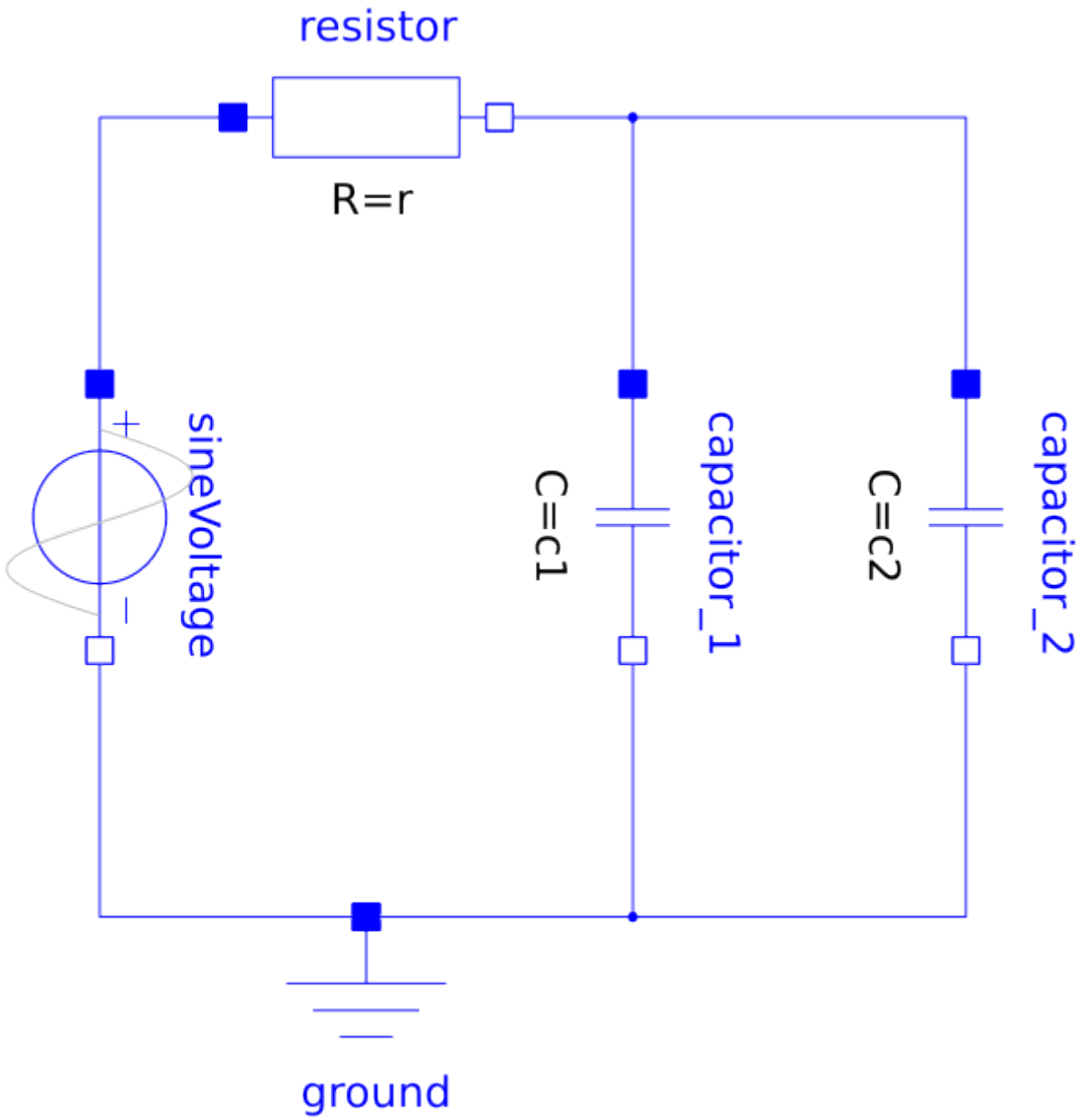

Figure 1. Modelica model of an electric circuit with two capacitors in parallel.

The circuit has two capacitors in parallel and can be represented with equations:

$$
\begin{aligned}
v_0 &= f(t) \\
v_R &= R \cdot i_0 \\
i_1 &= C_1 \cdot \dot{v}_1 \\
i_2 &= C_2 \cdot \dot{v}_2 \\
v_0 &= v_R + v_1 \\
v_2 &= v_1 \\
i_0 &= i_1 + i_2
\end{aligned}
$$

When both capacitive voltages v_1 and v_2 are chosen as state variables equation eqref $constraintEquationExampleCellier$ has no unknowns left, so it is a *constraint equation*.

There are different ways to solve this. The modeler could change the causality or exporting tools can try to reduce the perturbation index symbolically [3] to end up with a consistent initialization for the system.

3. Layered Standard Manifest File

This layered standard requires the use of a layered standard manifest file and it shall be stored inside the FMU at the following path: `/extra/org.fmi-standard.fmi-ls-dae/fmi-ls-manifest.xml`.

An example of a manifest file for this layered standard is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<fmiLayeredStandardManifest
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://fmi-standard.org/fmi-ls-manifest
../schema/fmi3LayeredStandardDaeManifest.xsd"
  xmlns="http://fmi-standard.org/fmi-ls-manifest"
  xmlns:fmi-ls="http://fmi-standard.org/fmi-ls-manifest"
  fmi-ls:fmi-ls-name="fmi-ls-dae"
  fmi-ls:fmi-ls-version="0.1"
  fmi-ls:fmi-ls-description="Layered standard for DAE support in FMI">
  <AlgebraicVariables>
    <AlgebraicVariable
      valueReference="889192448"/>
    <AlgebraicVariable
      valueReference="889192449"/>
  </AlgebraicVariables>
  <ModelStructure>
    <ContinuousStateDerivative
      valueReference="1124073472"
      dependencies="33554432 33554433 889192448 889192449 620756992"
      dependenciesKind="dependent dependent dependent dependent dependent"/>
    <ContinuousStateDerivative
      valueReference="1124073473"
      dependencies="33554432 33554433 889192448 889192449 620756992"
      dependenciesKind="dependent dependent dependent dependent dependent"/>
    <Residual
      valueReference="1409286144"
      dependencies="889192448 889192449"
      dependenciesKind="dependent dependent"
      index="1"/>
    <Residual
      valueReference="1409286145"
      dependencies="33554432 33554433 889192448 889192449"
      dependenciesKind="dependent dependent dependent dependent"
      index="1"/>
    <Output
      valueReference="603979776"
      dependencies="889192449"
      dependenciesKind="dependent"/>
    <Output
      valueReference="603979777"
      dependencies="889192448"
      dependenciesKind="dependent"/>
```

```
  </ModelStructure>
</fmiLayeredStandardManifest>
```

[Add figure here] shows the root element `fmiLayeredStandardManifest`.

Two additional elements - `<AlgebraicVariables>` and `<ModelStructure>` - are included in the layered standard manifest file to describe the DAE formulation.

Table 2. `fmiModelDescription` element details.

Element	Description
`<AlgebraicVariables>`	Ordered list of all algebraic variables exposed for the DAE formulation.
`<ModelStructure>`	Defines the structure of the DAE formulation for the model. Especially, the ordered lists of `outputs`, continuous-time states, `residuals`, and the initial unknowns (the unknowns during [InitializationMode]) are **re** -defined and overwrite the corresponding `<ModelStructure>` present in the `modelDescription.xml`. Furthermore, the dependency of the unknowns from the knowns (as described in [model-dependencies]) can be optionally defined for `outputs`, continuous-time states and initial unknowns.
[Annotations]	Optional annotations for the top-level element.

The XML attributes of `<fmiLayeredStandardManifest>` are:

Table 3. `<fmiLayeredStandardManifest>` attribute details.

Attribute	Namespace	Value	Description
`fmi-ls-name`	`http://fmi-standard.org/fmi-ls-manifest`	`org.fmi-standard.fmi-ls-dae`	Name of the layered standard in reverse domain name notation.
`fmi-ls-version`	`http://fmi-standard.org/fmi-ls-manifest`	`0.0.1`	Version of the layered standard. This layered standard uses semantic versioning, as defined in [PW13].
`fmi-ls-description`	`http://fmi-standard.org/fmi-ls-manifest`	`Layered standard for DAE support in FMI`	String with a brief description of the layered standard that is suitable for display to users.

3.1. Algebraic variables

The element `<AlgebraicVariables>` defines the ordered list of the algebraic variables.

Table 4. AlgebraicVariables elements.

Element	Description
`AlgebraicVariable`	The ordered list of all algebraic variables present in the `ModelVariables` element of the modelDescription.xml. This list must be the same size as the number of `<Residual>` elements.

Each [AlgebraicVariable] has only one attribute defining the value reference.

Table 5. Attributes to [AlgebraicVariable] element.

Attribute	Description
valueReference	The value reference for the algebraic variable present in the ModelVariables of the modelDescription.xml.

3.2. Model Structure

The structure of the model for the DAE-formulation is defined in element <ModelStructure>. It defines the dependencies between variables. The <ModelStructure> element is extended with an additional element - <Residual> - and the optional dependencies can now include algebraic variables.

An FMU that follows this layered standard must expose all residuals in order in <Residual>.

It should be noted that the ModelStructure below is not a straight extension of the ModelStructure from the core standard, since it e.g. does not specify how the elements ClockedState or EventIndicator are affected by the DAE-formulation. That is also why some elements in Table 6 are defined in a similar way to the core standard, with the exception that references to concepts in co-simulation, clocks, and events are excluded. Moreover, notes and examples that are already present in the core standard are omitted here.

Similar to the ModelStructure from the core FMI-standard, the optional part of the model structure defines in which way derivatives, outputs, and initial unknowns depend on inputs and/or parameters, and continuous-time states. Therefore the concepts from Model Exchange in the core-standard that are applicable to the DAE-formulation are repeated. The unknowns are extended with the <Residual> element, and each unknown can now also depend on <AlgebraicVariables>.

Table 6. ModelStructure elements.

Element	Description
Output	Ordered list of all outputs, in other words, a list of value references where every corresponding variable must have [causality] = [output] (and every variable with [causality] = [output] must be listed here). Attribute dependencies defines the dependencies of the outputs from the knowns after the [InitializationMode]. The functional dependency is defined as: $(\mathbf{y}_c, \mathbf{y}_d) := \mathbf{f}_{output}(\mathbf{x}_c, , \mathbf{z}_c, \mathbf{u}_c, \mathbf{u}_d, t, \mathbf{p})$
ContinuousStateDerivative	Ordered list of value references of all derivatives of continuous states. The order defined by this list defines the order of the elements for [fmi3GetContinuousStates], [fmi3SetContinuousStates], and [fmi3GetContinuousStateDerivatives]. The corresponding continuous-time states are defined by attribute [derivative] of the corresponding variable state-derivative element. The functional dependency is defined as: $\dot{\mathbf{x}}_c := \mathbf{f}_{der}(\mathbf{x}_c, \mathbf{z}_c, \mathbf{u}_c, \mathbf{u}_d, t, \mathbf{p})$

Element	Description
`Residual`	Ordered list of value references of all residual equations (constraints). The functional dependency is defined as: $0 := \mathbf{f}_{res}(\mathbf{x}_c, \mathbf{z}_c, \mathbf{u}_c, t, \mathbf{p})$
`InitialUnknown`	Ordered list of all exposed unknowns in [InitializationMode]. This list consists of all variables which: • are not clocked variables and have [causality] = [output] (with [initial] = [approx] or [calculated]), or • have [causality] = [calculatedParameter], or • are continuous-time states or state derivatives (defined with elements `<ContinuousStateDerivative>`) with [initial] = [approx] or [calculated], or • are algebraic variables that the FMU wishes to initialize in initialization mode. The resulting list is not allowed to have duplicates (for example, if a [state] is also an [output], it is included only once in the list). Attribute `dependencies` defines the dependencies of the unknowns from the knowns in [InitializationMode]. The functional dependency is defined as: $\mathbf{v}_{initialUnknowns} := \mathbf{f}_{init}(\mathbf{u}_c, \mathbf{u}_d, t_{start}, \mathbf{v}_{initial = exact})$ Since, `outputs`, continuous-time states and state derivatives are either present as knowns (if [initial] = [exact]) or as unknowns (if [initial] = [approx] or [calculated]), they can be inquired with `fmi3Get{VariableType}` in [InitializationMode].

Elements `<Output>`, `<ContinuousStateDerivative>`, `<Residual>`, and `<InitialUnknown>` have (partially) the following attributes:

Table 7. `<Output>`, `<ContinuousStateDerivative>`, `<Residual>`, and `<InitialUnknown>` common attributes details.

Attribute	Description
`valueReference`	The value reference of the unknown $v_{unknown}$.

Attribute	Description
dependencies	Optional attribute defining the algebraic dependencies as list of value references of the unknown $\mathbf{v}_{unknown}$ directly with respect to $\mathbf{v}_{known}$. For a real valued unknown and a real valued known, if the known is not listed among the dependencies then the partial derivative of the unknown with respect to that known is identically zero. If dependencies is not present, it must be assumed that the unknown depends on all knowns. If dependencies is present as empty list, the unknown depends on none of the knowns. Otherwise the unknown depends on the knowns defined by the given value references. Knowns $\mathbf{v}_{known}$ in [ContinuousTimeMode] (ME) for <Output>, <ContinuousStateDerivative> and <Residual> elements are: • inputs (variables with [causality] = [input]), • continuous states, • parameters (variables with [causality] = [parameter]), • algebraic variables (variables listed in the <AlgebraicVariables> element) • [independent] variable (usually time; [causality] = [independent]). Knowns $\mathbf{v}_{known}$ in [InitializationMode] (for elements <InitialUnknown>) are: • inputs (variables with [causality] = [input]), • variables with [initial] = [exact], • algebraic variables (variables listed in the <AlgebraicVariables> element) • [independent] variable (usually time; [causality] = [independent]).

Attribute	Description
dependenciesKind	If dependenciesKind is present, dependencies must be present and must have the same number of list elements. If dependenciesKind is not present, it must be assumed that the unknown $\mathbf{v}_{unknown}$ depends on the knowns $\mathbf{v}_{known}$ without a particular structure. Otherwise, the corresponding known $\mathbf{v}_{known,\,i}$ enters the equation as: = dependent: no particular structure, $\mathbf{f}(.., \mathbf{v}_{known,\,i}, ..)$ The following dependenciesKind is only allowed for floating point $\mathbf{v}_{unknown}$: = [constant]: constant factor, $\mathbf{c} \cdot \mathbf{v}_{known,\,i}$ where $\mathbf{c}$ is an expression that is evaluated before [fmi3EnterInitializationMode] is called. The following dependenciesKind is only allowed for floating point $\mathbf{v}_{unknown}$ in [ContinuousTimeMode] (ME), and not for <InitialUnknown> for [InitializationMode]: = [fixed]: fixed factor, $\mathbf{p} \cdot \mathbf{v}_{known,\,i}$ where $\mathbf{p}$ is an expression that is evaluated before [fmi3ExitInitializationMode] is called. = [tunable]: tunable factor, $\mathbf{p} \cdot \mathbf{v}_{known,\,i}$ where $\mathbf{p}$ is an expression that is evaluated before [fmi3ExitInitializationMode] is called and in [EventMode] or at a communication point (ME and CS) due to a change of a [tunable] parameter. = [discrete]: discrete factor, $\mathbf{d} \cdot \mathbf{v}_{known,\,i}$ where $\mathbf{d}$ is an expression that is evaluated before [fmi3ExitInitializationMode].

An additional attribute specifying the index must also be present for the <Residual> element.

Table 8. Unique attribute to <Residual> element.

Attribute	Description
index	The index of the residual.

References

- [cui2020mass] Cui, Hantao, Fangxing Li, and Joe H. Chow. "Mass-matrix differential-algebraic equation formulation for transient stability simulation." arXiv preprint arXiv:2008.03883, 2020.
- [cellier2006continuous] Cellier, François E., and Ernesto Kofman. "Continuous system simulation". Boston, MA: Springer US, 2006.
- [hairer2006numerical] Hairer, Ernst, Christian Lubich, and Michel Roche. "The numerical solution of differential-algebraic systems by Runge-Kutta methods". Vol. 1409. Springer, 2006.
- [cellier1993automated] Cellier, Francois E., and Hilding Elmqvist. "Automated formula manipulation supports object-oriented continuous-system modeling". IEEE Control Systems Magazine 13.2 (1993): 28-38.

[1] Source: https://en.wikipedia.org/wiki/Differential-algebraic_system_of_equations

[2] From [cellier2006continuous], p. 285.

[3] e.g. by using Pantelides algorithm, see [cellier1993automated]