Robust Initialization of Differential Algebraic Equations

EOOLT'07
Berlin

Bernhard Bachmann
Fachhochschule Bielefeld
Bielefeld

Peter Aronsson
MathCore Engineering AB
Linköping

Peter Fritzson
Linköping University
Linköping

Outline

- Mathematical Formulation of Hybrid DAEs
- Symbolic Transformation Steps
- Initialization in Modelica (Conventional)
- Higher-Index DAEs
- System versus Component initialization
- Example (3-Phase System)

Mathematical Formalism

General representation of hybrid DAEs:

$$\underline{0} = \underline{f}\left(t,\ \underline{\dot{x}}(t), \underline{x}(t), \underline{y}(t), \underline{u}(t), \underline{q}(t_e), \underline{q}_{pre}(t_e),\ \underline{c}(t_e), \underline{p}\right)$$

t	time
$\underline{\dot{x}}(t)$	vector of differentiated state variables
$\underline{x}(t)$	vector of state variables
$\underline{y}(t)$	vector of algebraic variables
$\underline{u}(t)$	vector of input variables
$\underline{q}(t_e), \underline{q}_{pre}(t_e)$	vectors of discrete variables
$\underline{c}(t_e)$	vector of condition expressions
$\underline{p}$	vector of parameters and/or constants

Numerical Aspects for Simulation (Explicit Euler Method)

Integration of explicit ordinary differential equations (ODEs):

$$\dot{\underline{x}}(t) = \underline{f}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right), \qquad \underline{x}(t_0) = \underline{x}_0$$

Numerical approximation of the derivative and/or right-hand-side:

$$\dot{\underline{x}}(t_n) \approx \frac{\underline{x}(t_{n+1}) - \underline{x}(t_n)}{t_{n+1} - t_n} \approx \underline{f}\left(t_n, \underline{x}(t_n), \underline{u}(t_n), \underline{p}\right)$$

Iteration scheme:

$$\underline{x}(t_{n+1}) \approx \underline{x}(t_n) + \left(t_{n+1} - t_n\right) \cdot \underline{f}\left(t_n, \underline{x}(t_n), \underline{u}(t_n), \underline{p}\right)$$

Calculating an approximation of $\underline{x}(t_{n+1})$ based on the values of $\underline{x}(t_n)$

Here:
Explicit Euler integration method

Convergence?

Basic Transformation Steps Mathematical View

Transformation to explicit state-space representation:

$$\underline{0} = \underline{f}\left(t, \underline{\dot{x}}(t), \underline{x}(t), \underline{y}(t), \underline{u}(t), \underline{p}\right)$$

$$\downarrow$$

$$\underline{0} = \underline{f}\left(t, \underline{z}(t), \underline{x}(t), \underline{u}(t), \underline{p}\right), \quad \underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix}$$

$$\nearrow$$

$$\underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix} = \underline{g}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

$$\downarrow$$

$$\underline{\dot{x}}(t) = \underline{h}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

$$\underline{y}(t) = \underline{k}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

Implicit function theorem:

Necessary condition for the existence of the transformation is that the following matrix is regular at the point of interest:

$$\det\left(\frac{\partial}{\partial \underline{z}} \underline{f}\left(t, \underline{z}(t), \underline{x}(t), \underline{u}(t), \underline{p}\right)\right) \neq 0$$

Symbolic Transformation
Algorithmic Steps

$$\underline{0} = \underline{f}\left(t, \underline{\dot{x}}(t), \underline{x}(t), \underline{y}(t), \underline{u}(t), \underline{p}\right)$$

$$\downarrow$$

- Construct bipartite graph representation
  - Structural representation of the equation system

$$\underline{0} = \underline{f}\left(t, \underline{z}(t), \underline{x}(t), \underline{u}(t), \underline{p}\right), \quad \underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix}$$

- Solve the matching problem
  - Assign to each variable exact one equation
  - Same number of equations and unknowns

$$\downarrow$$

$$\underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix} = \underline{g}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

- Construct a directed graph
  - Find sinks, sources and strong components
  - Sorting the equation system

$$\downarrow$$

$$\underline{\dot{x}}(t) = \underline{h}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

$$\underline{y}(t) = \underline{k}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

Initialization of Dynamic Models Conventional

- Initialization of "free" state variables
  - Transformed DAE after index-reduction
  - States can be chosen at start time
    - same number of additional equations and "free" states
- Initialization of parameters
  - Determine parameter settings
  - Parameters can be calculated at start time
    - same number of additional equations and "free" parameters
- Initialization mechanism in Modelica
  - attribute start
  - initial equation section
    - attribute fixed for parameters

$$\underline{0} = \underline{f}\left(t, \underline{\dot{x}}(t), \underline{x}(t), \underline{y}(t), \underline{u}(t), \underline{p}\right)$$

$$\downarrow$$

$$\underline{0} = \underline{f}\left(t, \underline{z}(t), \underline{x}(t), \underline{u}(t), \underline{p}\right), \quad \underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix}$$

$$\downarrow$$

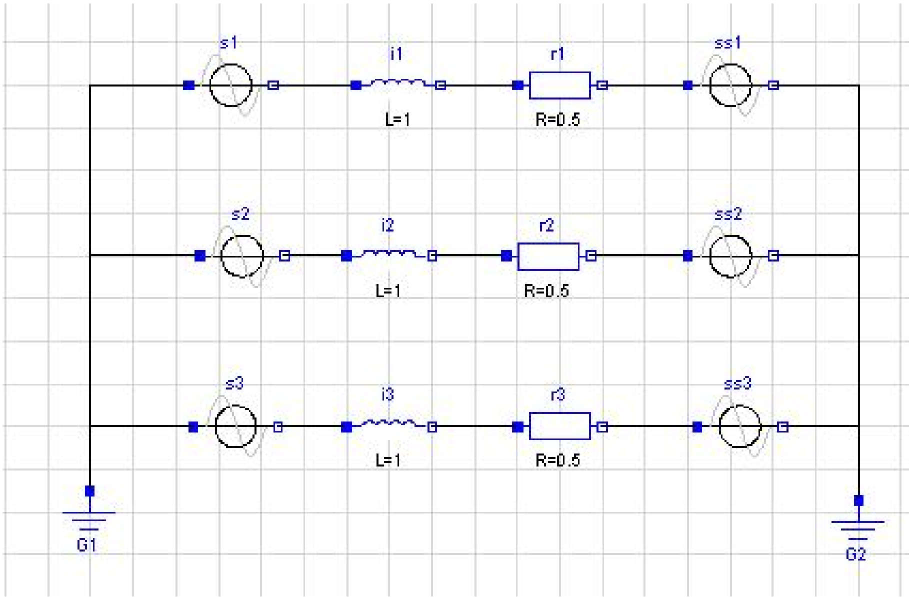
$$\underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix} = \underline{g}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

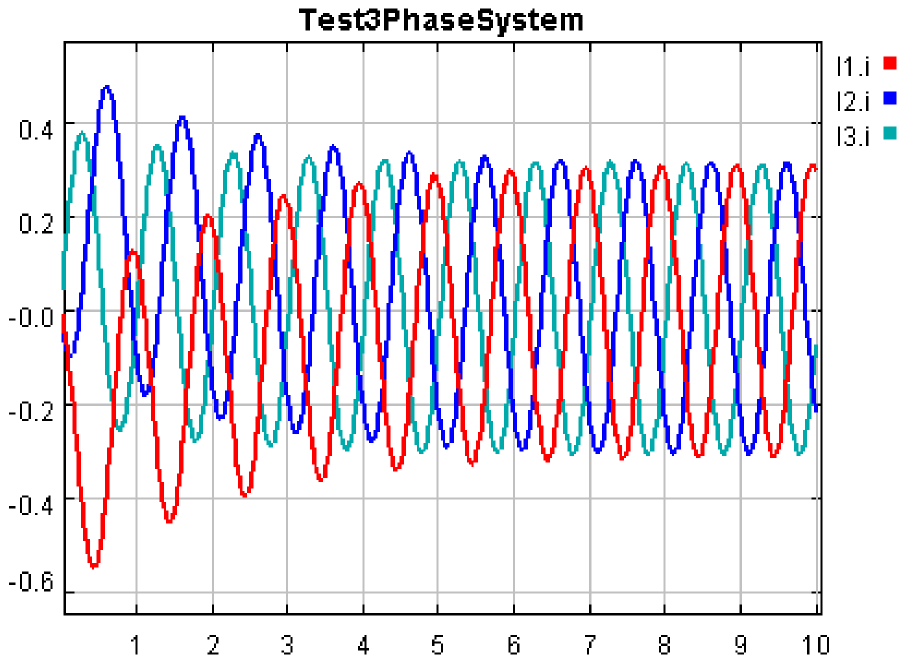
$$\downarrow$$

$$\underline{\dot{x}}(t) = \underline{h}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

$$\underline{y}(t) = \underline{k}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

Example: 3-Phase Electrical System





Steady-state initialization?

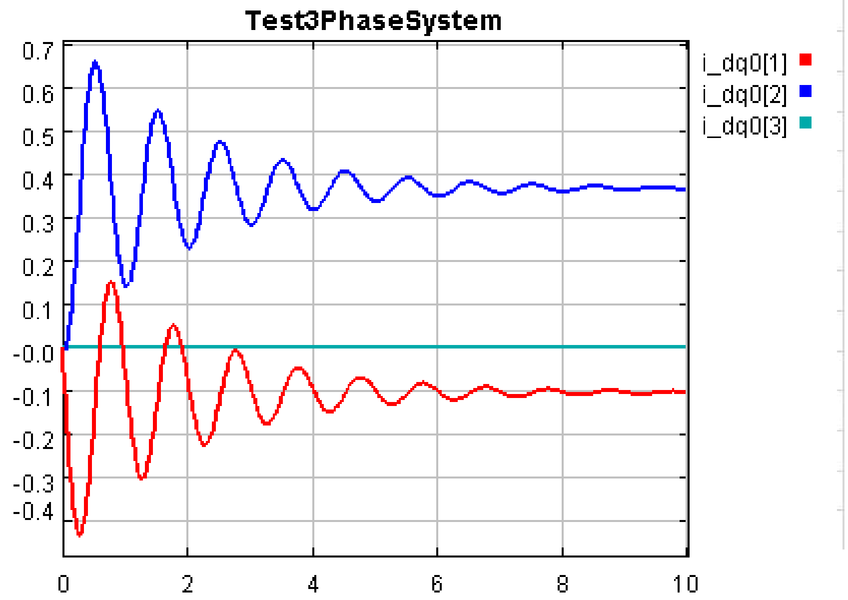
- States `I1.i, I2.i, I3.i` are not constant!
- Here: Initial values set to 0 (standard settings, attribute `start`)
- Transformation to rotating reference system necessary

Example: 3-Phase Electrical System

Park-Transformation to rotating reference system:

- Write states as vector `i_abc[3] = {I1.i, I2.i, I3.i}`
- Park-transformation to dq0-reference frame

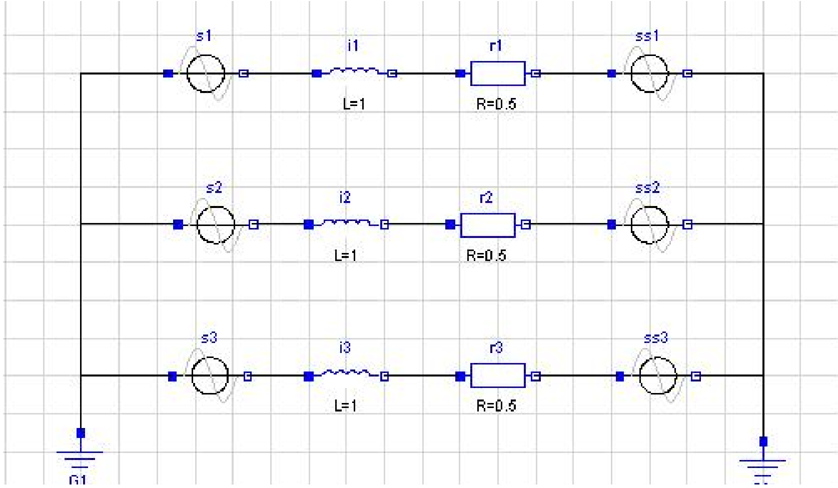
$$P = \frac{\sqrt{2}}{\sqrt{3}} \begin{pmatrix} \sin(\omega t) & \sin\left(\omega t + \frac{2\pi}{3}\right) & \sin\left(\omega t + \frac{4\pi}{3}\right) \\ \cos(\omega t) & \cos\left(\omega t + \frac{2\pi}{3}\right) & \cos\left(\omega t + \frac{4\pi}{3}\right) \\ 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}$$

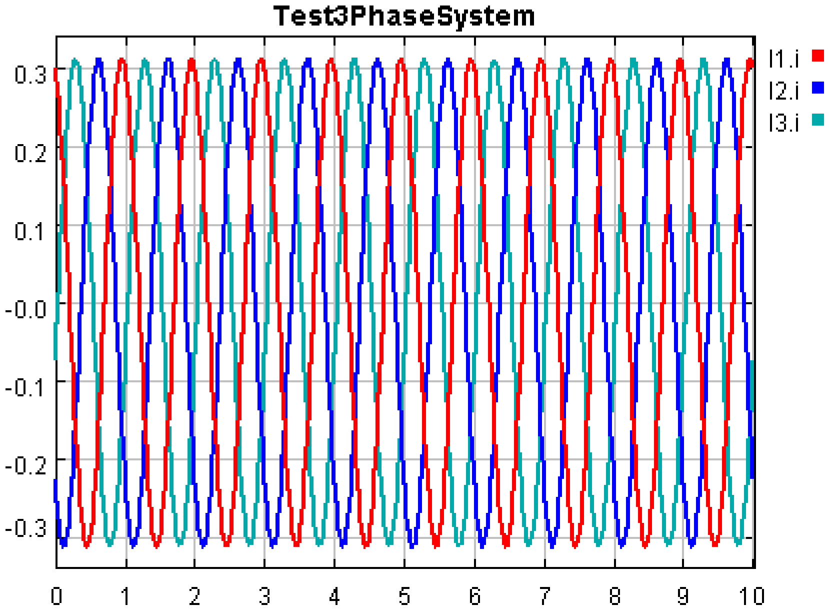


$$i\_dq0 = P \cdot i\_abc$$

Example: 3-Phase Electrical System
Initialize States

```
model Test3PhaseSystem
  parameter Real shift=0.4;
  Real i_abc[3]={I1.i,I2.i,I3.i};
  Real i_dq0[3];
  …
initial equation
  der(i_dq0)={0,0,0};
equation
  …
  i_dq0 = P*i_abc;
end Test3PhaseSystem
```



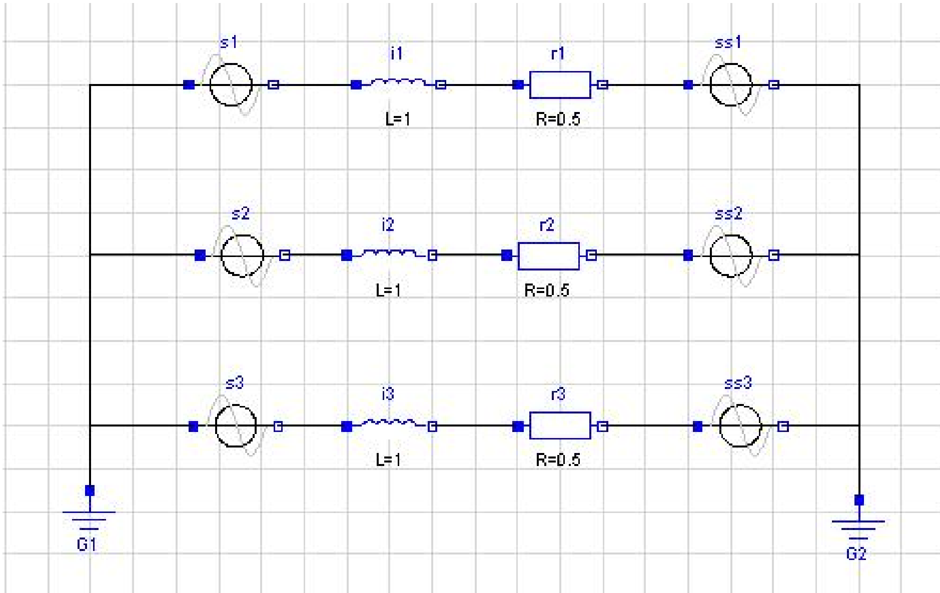


Steady-state initialization:

- Derivatives of `i_dq0` are only introduced during initialization
- Differentiation of `i_dq0 = P*i_abc` necessary
- No higher-index problem

Example: 3-Phase Electrical System Initialize Parameters

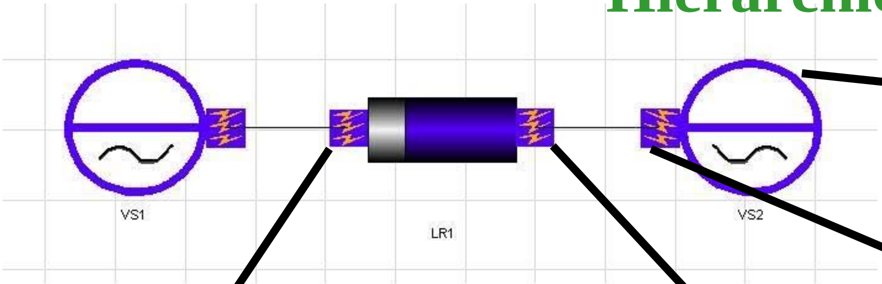
```
model Test3PhaseSystem
  parameter Real shift(fixed=false,start=0.1);
  Real i_abc[3]={I1.i,I2.i,I3.i}, u_abc[3]={S1.v,S2.v,S3.v};
  …
initial equation
  der(i_dq0)={0,0,0};
  power = -0.12865;
equation
  …
  u_dq0 = P*u_abc;
  i_dq0 = P*i_abc;
  power = u_dq0*i_dq0;
end Test3PhaseSystem
```



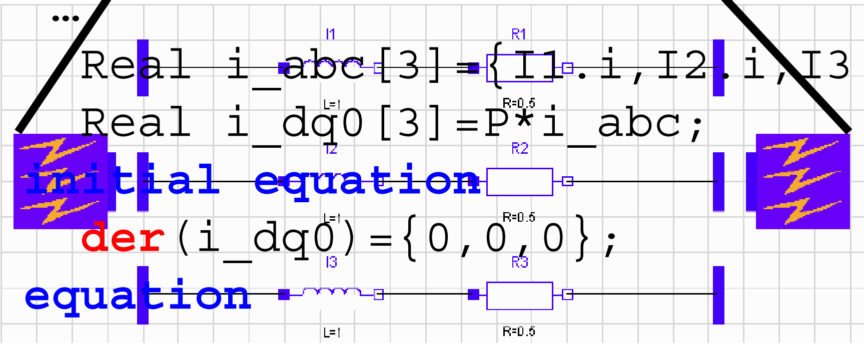
Parameter initialization:

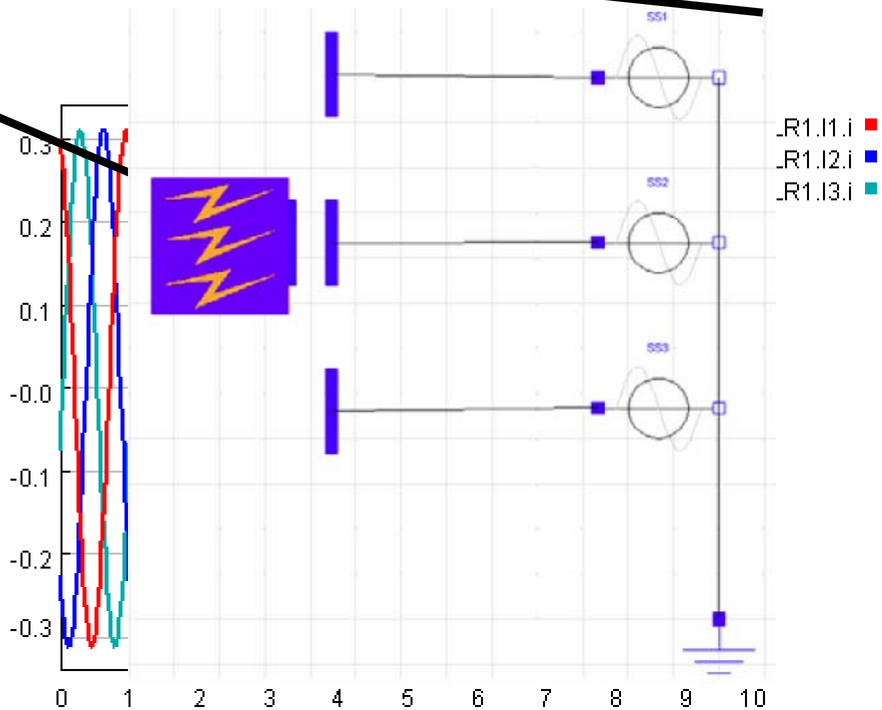
- Non-linear equation, no unique solution
  `power=u_dq0*i_dq0=-0.12865`
- Attribute `fixed` (default `true`)
- Attribute `start` (default `0`)

Example: 3-Phase Electrical System
Hierarchical Definition




```
model LR
  ...
   Real i_abc[3]={I1.i,I2.i,I3
   Real i_dq0[3]=P*i_abc;
initial equation
   der(i_dq0)={0,0,0};
equation
   ...
end LR
```





Define new LR – Component:

- States are `LR1.I1.i, LR1.I2.i, LR1.I3.i`
- Connectors based on rotating reference system
- Initial equations defined locally
- no higher-index problem

Higher-Index-DAEs

General representation of DAEs:

$$\underline{0} = \underline{f}\left(t,\ \underline{\dot{x}}(t), \underline{x}(t), \underline{y}(t), \underline{u}(t), \underline{p}\right)$$

t	time
$\underline{\dot{x}}(t)$	vector of differentiated state variables
$\underline{x}(t)$	vector of state variables
$\underline{y}(t)$	vector of algebraic variables
$\underline{u}(t)$	vector of input variables
$\underline{p}$	vector of parameters and/or constants

Differential index of a DAE:

The minimal number of analytical differentiations of the equation system necessary to extract by algebraic manipulations an explicit ODE for all unknowns.

Higher-Index-DAEs

DAE with differential index 0:

$$\underline{0} = \underline{f}\left(t,\ \dot{\underline{x}}(t), \underline{x}(t), \underline{u}(t), \underline{p}\right) \longrightarrow \dot{\underline{x}}(t) = \underline{g}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

DAE with differential index 1:

$$\underline{0} = \underline{f}\left(t,\ \dot{\underline{x}}(t), \underline{x}(t), \underline{y}(t), \underline{u}(t), \underline{p}\right)$$

$$\downarrow$$

$$\begin{aligned}\dot{\underline{x}}(t) &= \underline{h}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)\\ \underline{y}(t) &= \underline{k}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)\end{aligned} \longrightarrow \begin{aligned}\dot{\underline{x}}(t) &= \underline{h}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)\\ \dot{\underline{y}}(t) &= \frac{d}{dt}\underline{k}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)\end{aligned}$$

Higher Index Problems
Structurally Singular Systems

- Higher-index DAEs
  - Differential index of a DAE
  - Structural singularity of the adjacence matrix
  - Index reduction method using symbolic differentiation of equations
- Numerical issues
  - Consistent initialization
  - Drift phenomenon
  - Dummy derivative method
- State selection mechanism in Modelica
  - Attribute `stateSelect:`
    `never, avoid, default, prefer, always`

$$\underline{0} = \underline{f}\left(t, \underline{\dot{x}}(t), \underline{x}(t), \underline{y}(t), \underline{u}(t), \underline{p}\right)$$

$$\downarrow$$

$$\underline{0} = \underline{f}\left(t, \underline{z}(t), \underline{x}(t), \underline{u}(t), \underline{p}\right), \quad \underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix}$$

$$\downarrow$$

$$\underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix} = \underline{g}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

$$\downarrow$$

$$\underline{\dot{x}}(t) = \underline{h}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

$$\underline{y}(t) = \underline{k}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

Higher-Index-DAEs -- Numerical Problems

- Consistent initial conditions
  – Relation between states are eliminated when differentiating
  – Initial conditions need to be determined using the algebraic constrains
  – Automatic procedure possible using assign algorithm on the constrained equations

- Drift phenomenon
  – Algebraic constrained no longer fulfilled during simulation
  – Even worse when simulating stiff problems

=> Dummy-Derivative method

Principles of the Dummy-Derivative Method

- Matching algorithm fails
  - System is structurally singular
  - Find minimal subset of equations
    - more equations than unknown variables
  - Singularity is due to equations constraining states

- Differentiate subset of equations
  - Static state selection during compile time
    - choose one state and corresponding derivative as purely algebraic variable
      - so-called dummy-state and dummy derivative
    - by differentiation introduced variables are algebraic
    - continue matching algorithm
    - check initial conditions
  - Dynamic state selection during simulation time
    - store information on constrained states
    - make selection dynamically based on heuristic criteria
    - new state selection triggers an event (re-initialize states)

Initialization of Higher-Index Problems

- **Initialization (conventional)**
  - Transformed DAE after index-reduction
  - Define initial equations on system level
    - same number of additional equations and “free” states

$$\underline{0} = \underline{f}\left(t, \underline{\dot{x}}(t), \underline{x}(t), \underline{y}(t), \underline{u}(t), \underline{p}\right)$$

$$\downarrow$$

$$\underline{0} = \underline{f}\left(t, \underline{z}(t), \underline{x}(t), \underline{u}(t), \underline{p}\right), \quad \underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix}$$

$$\downarrow$$

- **Initialization (advanced)**
  - Transformed DAE after index-reduction
  - Define initial equations on component level
    - same number of additional equations and “free” states locally
    - consistent overdetermined system

$$\underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix} = \underline{g}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

$$\downarrow$$

$$\underline{\dot{x}}(t) = \underline{h}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

$$\underline{y}(t) = \underline{k}\left(t, \underline{x}(t), \underline{u}(t), \underline{p}\right)$$

Solving Overdetermined Systems

Nonlinear system of equations

- *m*, number of equations
- *n*, number of variables
- $m \geq n$

$$\begin{aligned} f_1(z_1,\ldots,z_n) &= 0 \\ &\vdots \\ f_m(z_1,\ldots,z_n) &= 0 \end{aligned}$$

Corresponding minimization problem

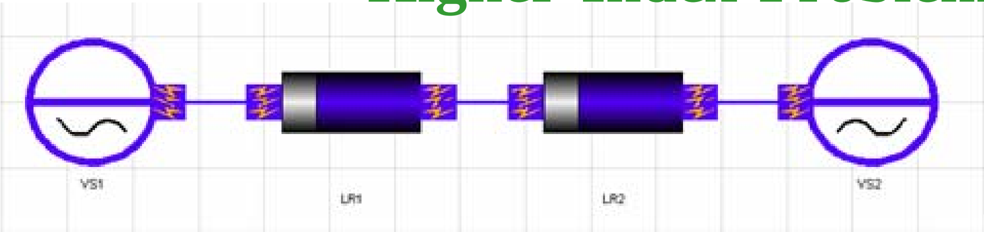
- solution solves the nonlinear system of equations

$$F\left(z_1,\ldots,z_n\right) = \sum_{i=1}^{m} f_i(z_1,\ldots,z_n)^2 \rightarrow \min$$

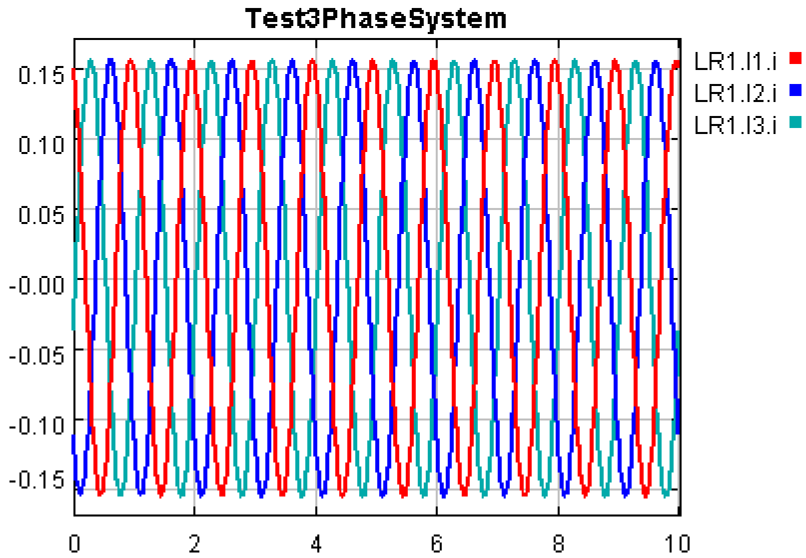
Derivative-free methods implemented in OpenModelica

- Simplex method of Nelder and Mead
- Minimization method of Brent

Example: 3-Phase Electrical System
Higher-Index-Problem (Conventional)



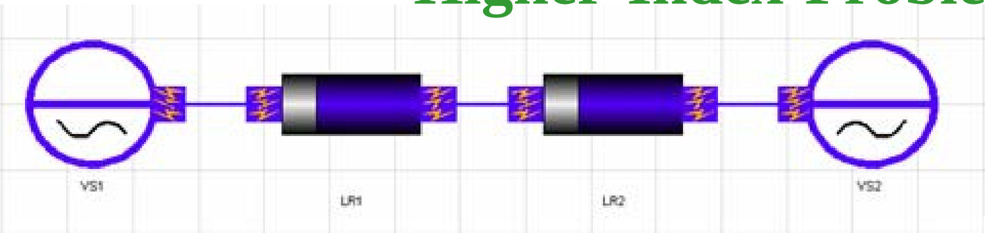

```
model Test3PhaseSystem
 …
initial equation
   der(LR1.i_dq0)={0,0,0};
equation
   …
end Test3PhaseSystem
```



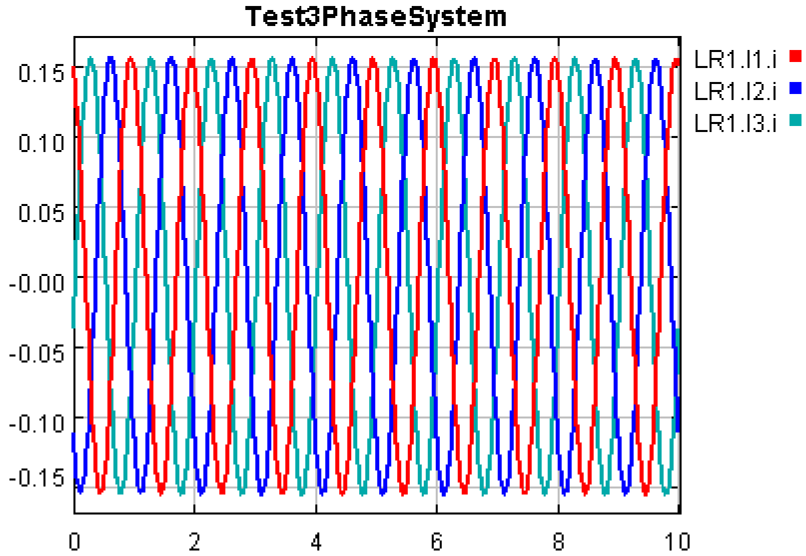
Higher Index problem:

- Algebraic dependency between differentiated variables `LR1.I1.i`, `LR1.I2.i`, `LR1.I3.i` and `LR2.I1.i`, `LR2.I2.i`, `LR2.I3.i`
- Only 3 states are left after index-reduction
- Define initial equations globally (Cumbersome)

Example: 3-Phase Electrical System Higher-Index-Problem (Advanced)




```
model LR
 …
initial equation
  der(i_dq0)={0,0,0};
equation
  …
end LR
```



Higher Index problem:

- Algebraic dependency between differentiated variables `LR1.I1.i`, `LR1.I2.i`, `LR1.I3.i` and `LR2.I1.i`, `LR2.I2.i`, `LR2.I3.i`
- Initial equations still defined locally
- Overdetermined equation system during initial time!

Conclusions and Future Work

- Advanced Initialization of DAEs
  - System versus component initialization
  - Well-posed overdetermined systems
- Prototype in OpenModelica
  - Implementation of concept
    - derivative-free minimization algorithms
  - Thorough testing necessary
  - Improve efficiency
  - Use advanced numerical minimization algorithms
    - Globally convergent methods
    - Calculation of Jacobian matrix of the equation system