Improvements of the Initialization Method of DAEs in OpenModelica

Lennart A. Ochel Bernhard Bachmann Willi Braun

FH Bielefeld - University of Applied Sciences

Outline

- ⇨ Mathematical Formulation of Hybrid DAEs
- ⇨ Symbolic Transformation Steps
- ⇨ Initialization in Modelica (Conventional)
- ⇨ Initialization in OpenModelica

Mathematical Formalism

general representation of hybrid DAEs:

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

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/constants

Principles of Numerical Integration Methods (Example: 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:

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

Convergence?

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

Here:
Explicit Euler integration method

Symbolic Transformation Steps

Transform to explicit state-space representation:

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

$$\underline{z} = \begin{pmatrix} \underline{\dot{x}} \\ \underline{y} \end{pmatrix} = \underline{g}\left(t, \underline{x}, \underline{u}, \underline{q}, \underline{q}_{pre}, \underline{c}, \underline{p}\right) \longrightarrow \begin{aligned} \underline{\dot{x}} &= \underline{h}\left(t, \underline{x}, \underline{u}, \underline{q}, \underline{q}_{pre}, \underline{c}, \underline{p}\right) \\ \underline{y} &= \underline{k}\left(t, \underline{x}, \underline{u}, \underline{q}, \underline{q}_{pre}, \underline{c}, \underline{p}\right) \end{aligned}$$

Implicit function theorem:

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

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

Initialization in Modelica

Initialization of “free” state variables

- same number of “free” states and additional equations

Initialization of “free” parameters

- same number of “free” parameters and additional equations

Initialization mechanism in Modelica

- initial equation/algorithm sections
- attribute fixed, start and nominal

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

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

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

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

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

Initialization in OpenModelica

Nonlinear system of equations

- m, number of equations
- n, number of variables
- $m \geq n$, over-determined
- $m \leq n$, under-determined

$$G_1(z_1, \dots, z_n) = 0$$
$$\vdots$$
$$G_m(z_1, \dots, z_n) = 0$$

Corresponding minimization problem

- solution solves the nonlinear system of equations

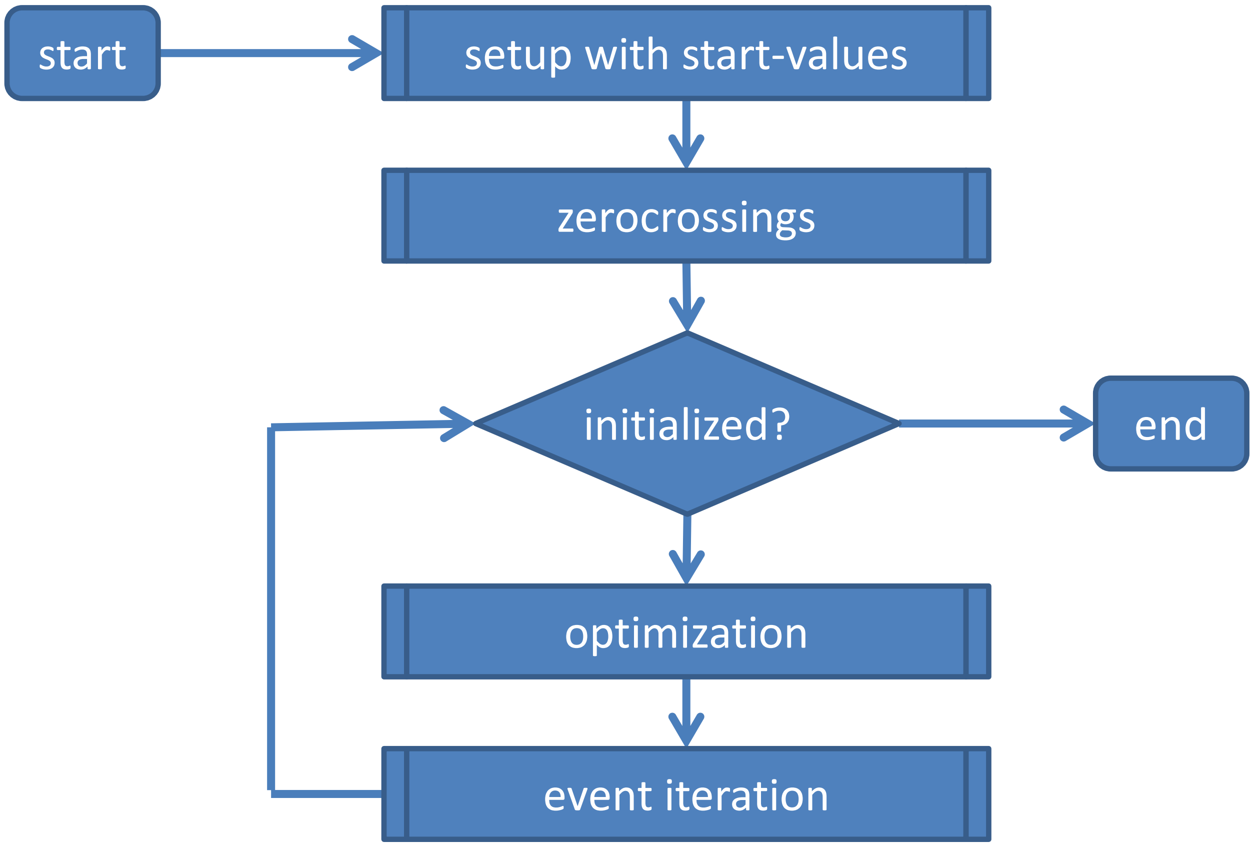
$$\sum_{i=1}^{m} G_i(z_1, \dots, z_n)^2 \to \min$$

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

Derivative-free method in OpenModelica

- based on simplex method of Nelder and Mead

Initialization in OpenModelica



Full Support of Start-Values

initial equation/algorithm

equations are valid at the end of the initialization

$$G_1(z_1, \ldots, z_n) = 0$$
$$\vdots$$
$$G_m(z_1, \ldots, z_n) = 0$$

attribute: start

guesses are valid at the beginning of the initialization

$$z_i - \mathrm{start}(z_i) = 0$$

resulting optimization:

$$\min\left\{\lambda \sum_{i=1}^{m} G_i^2 + (1-\lambda)\sum_{i}(z_i - \mathrm{start}(z_i))^2\right\}$$

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

Full Support of Start-Values

```
model StartValue
  Real x;
  Real y(start=-3);
initial equation
  x^2 = 10;
equation
  der(x) = time;
  y = x;
end StartValue;
```

var	expected solution	solution 2
x	$-\sqrt{10}$	$\sqrt{10}$
y	$-\sqrt{10}$	$\sqrt{10}$
	OpenModelica	Dymola

$$\min\left\{\lambda \sum_{i=1}^{m} G_i^2 + (1-\lambda)\sum_{i}(z_i - \text{start}(z_i))^2\right\}$$

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

Under-Determined Initialization

Fewer equations than unfixed states/parameters:

⇨ Mechanism to auto-fix free states/parameters

1. Symbolic analysis of dependencies

   ⇨ future work

2. Numerical analysis of dependencies

   ⇨ implemented

Scaling of Initial Residuals

$$\text{Initialization} \Leftrightarrow 0 = \min\{F(\underline{z})\}$$

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

$$\mathrm{F}(\underline{z}) = \lambda \sum_{i=1}^{m} G_i^2 + (1-\lambda) \sum_{i} (z_i - \mathrm{start}(z_i))^2$$

Scaling of Initial Residuals

Example: $$F(z_1, z_2) = \underbrace{(z_1 - 10^{-6})}_{G_1}^2 + \underbrace{(z_2 - 10^{6})}_{G_2}^2$$

- $F(10^{-6}, 10^{6}) = 0$
- $F(1.1 \cdot 10^{-6}, 10^{6}) = 10^{-14}$ 10% deviation in z_1
- $F(10^{-6}, 1.1 \cdot 10^{6}) = 10^{10}$ 10% deviation in z_2

Scaling of Initial Residuals

Example: $$F(z_1, z_2) = \underbrace{(z_1 - 10^{-6})}_{G_1}{}^2 + \underbrace{(z_2 - 10^{6})}_{G_2}{}^2$$

- $F(10^{-6}, 10^{6}) = 0$
- $F(1.1 \cdot 10^{-6}, 10^{6}) = 10^{-14}$ 10% deviation in z_1
- $F(10^{-6}, 1.1 \cdot 10^{6}) = 10^{10}$ 10% deviation in z_2

With Scaling: $$\tilde{F}(\underline{z}) = \lambda \sum_{i=1}^{m} K_i^{-1} \cdot G_i^2 + (1 - \lambda) \sum_i K_i^{-1} \cdot (z_i - \mathrm{start}(z_i))^2$$

- $\tilde{F}(10^{-6}, 10^{6}) = 0$
- $\tilde{F}(1.1 \cdot 10^{-6}, 10^{6}) = 0.1$ 10% deviation in z_1
- $\tilde{F}(10^{-6}, 1.1 \cdot 10^{6}) = 0.1$ 10% deviation in z_2

Scaling of Initial Residuals

Example: $\tilde{F}(z_1, z_2) = K_1^{-1} \cdot (z_1 - 10^{-6})^2 + K_2^{-1} \cdot (z_2 - 10^6)^2$

How to choose K_i^{-1}? (linear dependencies of one variable)

- $K_1 = \text{nominal}(z_1)$
- $K_1 = \text{nominal}(z_2)$

How to handle in general?

- nonlinear equations
- multiple dependencies

Scaling of Initial Residuals

How to choose scaling coefficients in general?

$$\tilde{F}(\underline{z}) = \lambda \sum_{i=1}^{m} K_i^{-1} \cdot G_i^2 + (1-\lambda) \sum_{i} K_i^{-1} \cdot (z_i - \mathrm{start}(z_i))^2$$

$$K_i = \max_j \left\{ \mathrm{nominal}(x_j) \cdot \left| \frac{\partial G_i(\mathrm{nominal}(x))}{\partial x_j} \right|, Kmin \right\}$$

(derived from differential error analysis)

Kmin, heuristically treatment for small K_i

Conclusions

- Reliable initialization of standard models (OMC test suite)
- Initialization of consistent over-determined systems
- Initialization of under-determined systems
- Full support of start values for all variables
    - see Modelica specification
- Numerical improvements by robust scaling techniques
- First tests with real-world-problems:
    - First tests with models from Siemens Power library successfull

Future Work

- Efficiency improvements
    - Implementation of more advanced optimization algorithms
    - Involve boundary conditions (min/max-values)
    - Symbolic preprocessing of initalization problem
- Initialization of under-determined systems
    - Symbolic analysis of dependencies between states and initial equations
- Real-World-Problems:
    - More advanced tests with models from Siemens Power library
    - Full support of Modelica Standard Library (OMC functionality)