Department of Applied Mathematics,
University of Applied Sciences Bielefeld,
33609 Bielefeld, Germany

Enhancement of the OpenModelica Compiler – Analytical calculation of the Jacobian matrix

Melanie Krems,
Bernhard Bachmann, Willi Braun

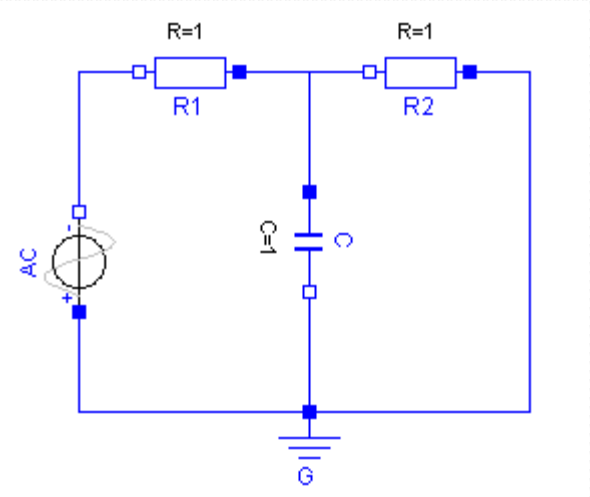
OpenProd - 01|SO9029C

Overview

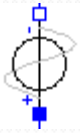
1. Differential Algebraic Equation System (DAE)
2. Numerical Integration - DASSL
3. Integration of the Jacobian Matrix
4. Test and Results
5. Outlook

1. Differential Algebraic Equation System (DAE)

```
model simpleCircuit
    Modelica.Electrical.Analog.Basic.Resistor R1
    Modelica.Electrical.Analog.Basic.Resistor R2
    Modelica.Electrical.Analog.Basic.Capacitor C
    Modelica.Electrical.Analog.Basic.Ground G
    Modelica.Electrical.Analog.Sources.SineVoltage AC
equation
    connect(R1.p, R2.n)
    connect(R1.p, C.p)
    connect(AC.n, R1.n)
    connect(AC.p, G.p)
    connect(C.n, G.p)
    connect(R2.p, G.p)
end simpleCircuit;
```



1. Differential Algebraic Equation System (DAE)

AC 	1.) 0 = AC.p.i + AC.n.i 2.) AC.v = AC.p.v-AC.n.v 3.) AC.i = AC.p.i 4.) AC.v = AC.VA*sin(2*AC.PI*AC.f*time)	R1 	1.) 0 = R1.p.i + R1.n.i 2.) R1.v = R1.p.v – R1.n.v 3.) R1.i = R1.p.i 4.) R1.v = R1.R * R1.i
C 	1.) 0 = C.p.i + C.n.i 2.) C.v = C.p.v – C.n.v 3.) C.i = C.p.i 4.) C.i = C.C*der(C.v)	R2 	1.) 0 = R2.p.i + R2.n.i 2.) R2.v = R2.p.v – R2.n.v 3.) R2.i = R2.p.i 4.) R2.v = R2.R * R2.i
G 	G.p.v = 0		
wires	1.) R1.p.v = R2.n.v 2.) R1.p.v = C.p.v 3.) AC.n.v = R1.n.v 4.) AC.p.v = G.p.v 5.) C.n.v = G.p.v 6.) R2.p.v = G.p.v	flow at node	1.) 0 = AC.n.i + R1.n.i 2.) 0 = R2.p.i + R2.n.i + C.p.i 3.) 0 = R1.p.i + G.p.i + C.n.i + AC.p.i

1. Differential Algebraic Equation System (DAE)

10 equations

(1) R1.v = (-R1.R) * R1.n.i

(2) R1.v = C.p.v - R1.n.v

(3) R2.v = (-R2.R) * R2.n.i

(4) R2.v = -C.p.v

(5) -C.n.i = C.C * **der(C.v)**

(6) C.v = C.p.v

(7) AC.y = A*sin(2*π*f*t+φ)

(8) AC.y = -R1.n.v

(9) 0.0 = R1.n.i + (C.n.i + G.p.i - R2.n.i)

(10) 0.0 = R2.n.i - C.n.i - R1.n.i

10 variables

(1) AC.y

(2) G.p.i

(3) C.n.i

(4) C.p.v

(5) C.v

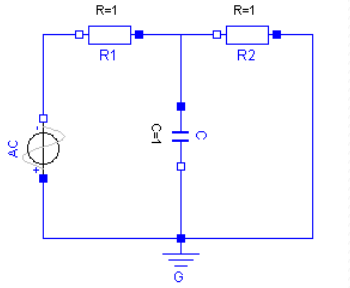
(6) R2.n.i

(7) R2.v

(8) R1.n.i

(9) R1.n.v

(10) R1.v



1. Differential Algebraic Equation System (DAE)

General representation of DAEs

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

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

1. Differential Algebraic Equation System (DAE)

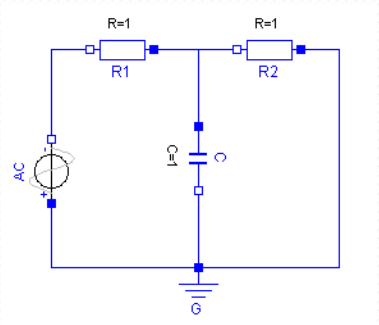
General representation - Example

$$\underline{p} = \begin{pmatrix} R1.R \\ R2.R \\ C.C \\ AC.offset \\ AC.startTime \\ AC.amplitude \\ AC.freqHZ \\ AC.phase \end{pmatrix}$$

$$\underline{y}(t) = \begin{pmatrix} R1.n.i \\ R1.v \\ R1.n.v \\ R2.n.i \\ R2.v \\ C.n.i \\ C.p.v \\ AC.y \\ G.p.i \end{pmatrix}$$

$$x(t) = (C.v)$$

$$\dot{x}(t) = (\dot{C}.v)$$



1. Differential Algebraic Equation System (DAE)

Symbolic Transformation to State Space Form

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

$$\downarrow$$

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

$$\downarrow$$

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

$$\downarrow$$

$$\dot{\underline{x}}(t) = \underline{h}(t, \underline{x}(t), \underline{u}(t), \underline{p})$$
$$\underline{y}(t) = \underline{k}(t, \underline{x}(t), \underline{u}(t), \underline{p})$$

1. Differential Algebraic Equation System (DAE)

State Space Form - Example

(1) AC.y = A*sin(2*π*f*t+φ)

(2) R1.n.v = - AC.y

(3) C.p.v = C.v

(4) R2.v = -C.p.v

(5) R2.n.i = - R2.v/R2.R

(6) R1.v = C.p.v - R1.n.v

(7) R1.n.i = -R1.R/ R1.v

(8) C.n.i = R2.n.i - R1.n.i

(9) Ċ.v = -C.n.i/C.C

(10) G.p.i = R2.n.i - C.n.i - R1.n.i

$$\underline{\dot{x}}(t) = (\dot{C}.v)$$

$$\underline{y}(t) = \begin{pmatrix} R1.n.i \\ R1.v \\ R1.n.v \\ R2.n.i \\ R2.v \\ C.n.i \\ C.p.v \\ AC.y \\ G.p.i \end{pmatrix}$$

2. Numerical Integration – DASSL

Solution Method – Explicit Euler (*forward Euler*)

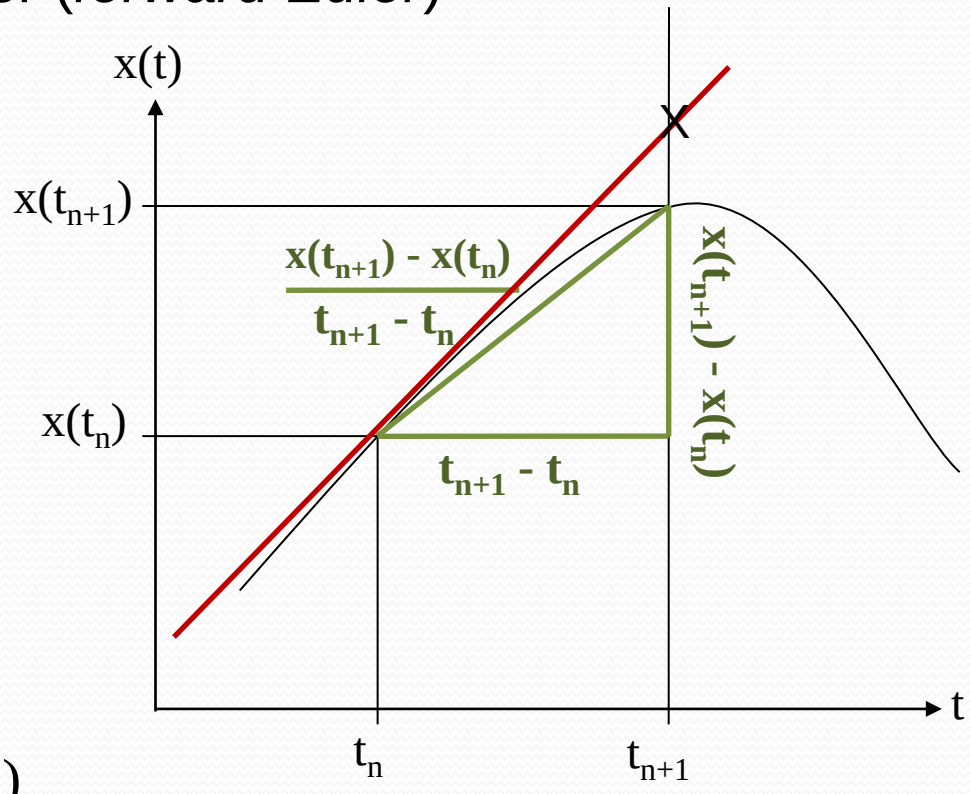
$$\dot{\underline{x}}(t) = \underline{f}(t, \underline{x}(t)) \quad \underline{x}(t_0) = \underline{x}_0$$

Numerical approximation for the derivative

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

giving an approximation of x at time t+1

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



2. Numerical Integration – DASSL

Solution Method – Implicit Euler (*backward Euler*)

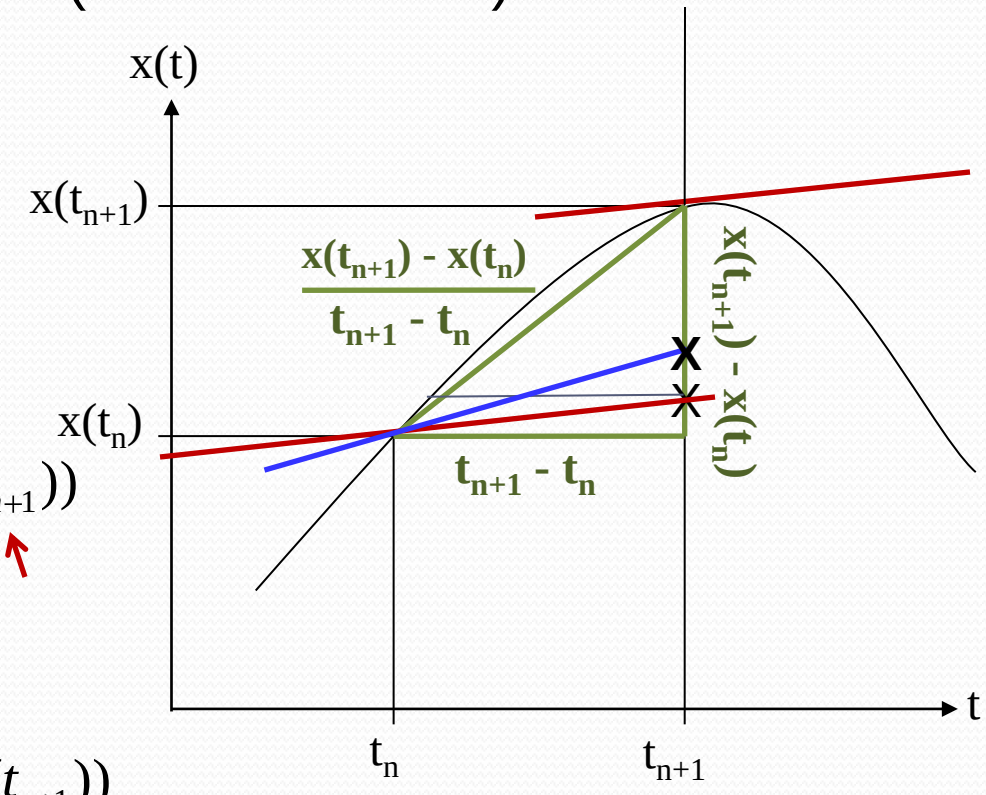
$$\dot{\underline{x}}(t) = \underline{f}(t, \underline{x}(t)) \quad \underline{x}(t_0) = \underline{x}_0$$

Numerical approximation for the derivative

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

Solve for $x(t_{n+1})$

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



2. Numerical Integration – DASSL

Using explixit and implicit Euler as a predictor-corrector pair

$$\underline{x}_{n+1}^{pred} \approx \underline{x}_n + (t_{n+1} - t_n) \cdot \underline{f}(t_n, \underline{x}_n)$$

$$\underline{x}_{n+1} \approx \underline{x}_n + (t_{n+1} - t_n) \cdot \underline{f}(t_{n+1}, \underline{x}_{n+1}^{pred})$$

Here: one iteration of the corrector

2. Numerical Integration – DASSL

Solution Method – DASSL

$$0 = \underline{f}(t, \underline{x}(t), \dot{\underline{x}}(t)) \qquad \underline{x}(t_0) = \underline{x}_0$$

Evalute a *predictor polynomial* for an initial guess for $\dot{x}_{n+1}^{(0)}, x_{n+1}^{(0)}$

Approximation for the derivative (*corrector equation*)

$$0 = \underline{f}\left(t_{n+1}, \underline{x}_{n+1}, \dot{x}_{n+1}^{(0)} - \frac{\alpha_s}{h_{n+1}} (\underline{x}_{n+1} - x_{n+1}^{(0)}) \right) \quad \alpha_s = -\sum_{j=1}^{k} \frac{1}{j}$$

„fixed leading coefficent“

2. Numerical Integration – DASSL

$$0 = \underline{f}\left(t_{n+1}, \underline{x}_{n+1}, \dot{x}_{n+1}^{(0)} - \frac{\alpha_s}{h_{n+1}} (\underline{x}_{n+1} - x_{n+1}^{(0)}) \right) \qquad \alpha_s = -\sum_{j=1}^{k} \frac{1}{j}$$

„fixed leading coefficent“

Simpler Notation

$$0 = \underline{f}(t_{n+1}, \underline{x}_{n+1}, \alpha \cdot \underline{x}_{n+1} + \beta)$$

$$\alpha = -\frac{\alpha_s}{h_{n+1}} \qquad \beta = \dot{x}_{n+1}^{(0)} - \alpha \cdot x_{n+1}^{(0)}$$

Solving the corrector equation

using a modified Newton iteration

$$\underline{x}^{(m+1)} = \underline{x}^{(m)} - cG^{-1} \underline{f}(t, \underline{x}^{(m)}, \alpha \cdot \underline{x}^{(m)} + \beta)$$

„Iteration matrix“

$$G = \alpha \frac{\partial f}{\partial \dot{x}} + \frac{\partial f}{\partial x}$$

2. Numerical Integration – DASSL

Calculation of the iteration matrix

$$G = \alpha \frac{\partial f}{\partial \dot{x}} + \frac{\partial f}{\partial x}$$

a) using finite differences

- numerical
- Calculation of all values in each time step

b) analytical calculation

- Evaluation at each time step
- Information about the sparsity pattern

3. Integration of the Jacobian Matrix

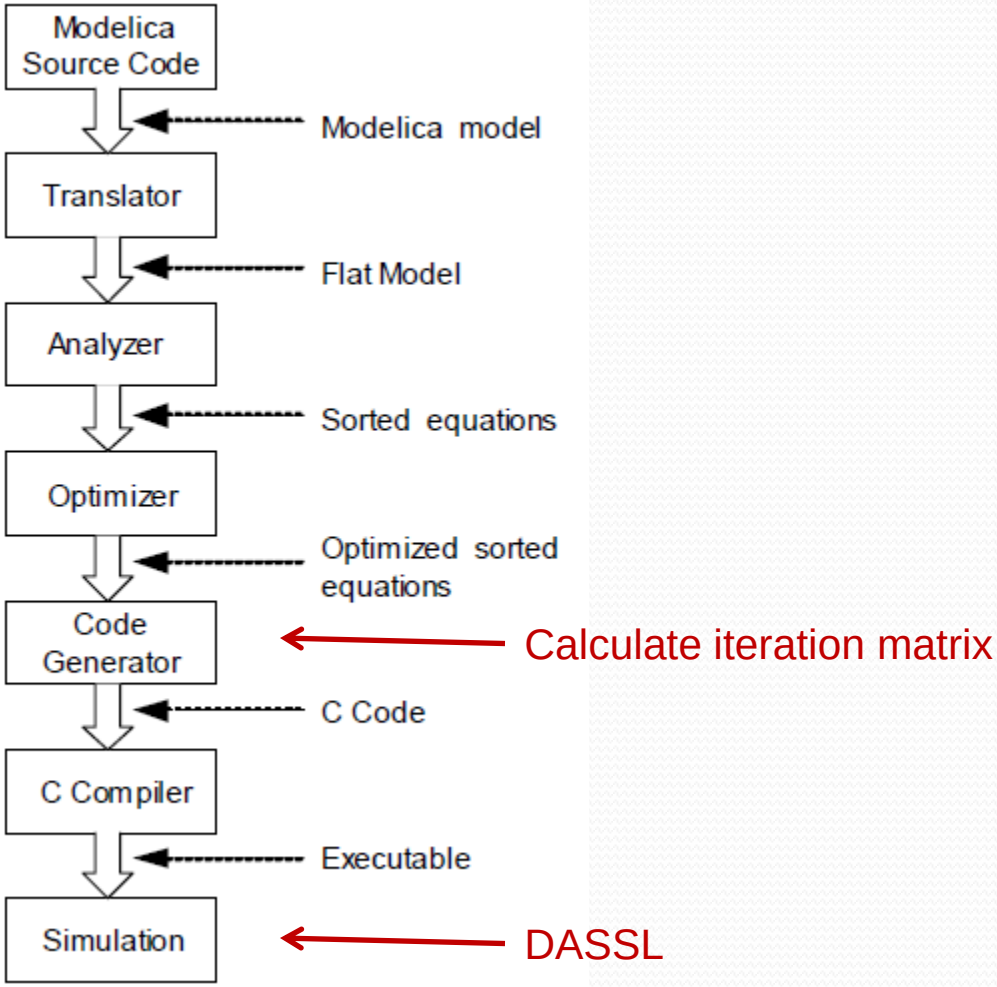
Calculation using Automatic Differentiation (AD)

Example:

$$
\begin{aligned}
&[y_1, y_2, b] = f(x, a) \{ \\
&\quad b = x \\
&\quad y_1 = a * \sin(b) \\
&\quad y_2 = b * y_1 \\
&\}
\end{aligned}
$$

$$
\begin{aligned}
&[y_1, y_2, y_1', y_2', b] = f_{AD}(x, x', a) \{ \\
&\quad b = x \\
&\quad b' = 1 \\
&\quad y_1 = a * \sin(b) \\
&\quad y_1' = a * \cos(b) * b' \\
&\quad y_2 = b * y_1 \\
&\quad y_2' = b' * y_1 + b * y_1' \\
&\}
\end{aligned}
$$

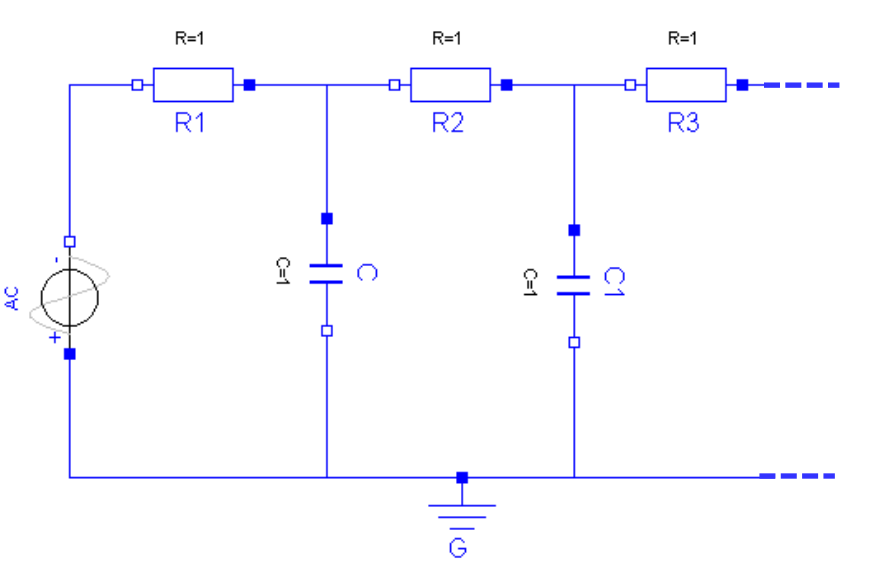
3. Integration of the Jacobian Matrix



4. Test and Results

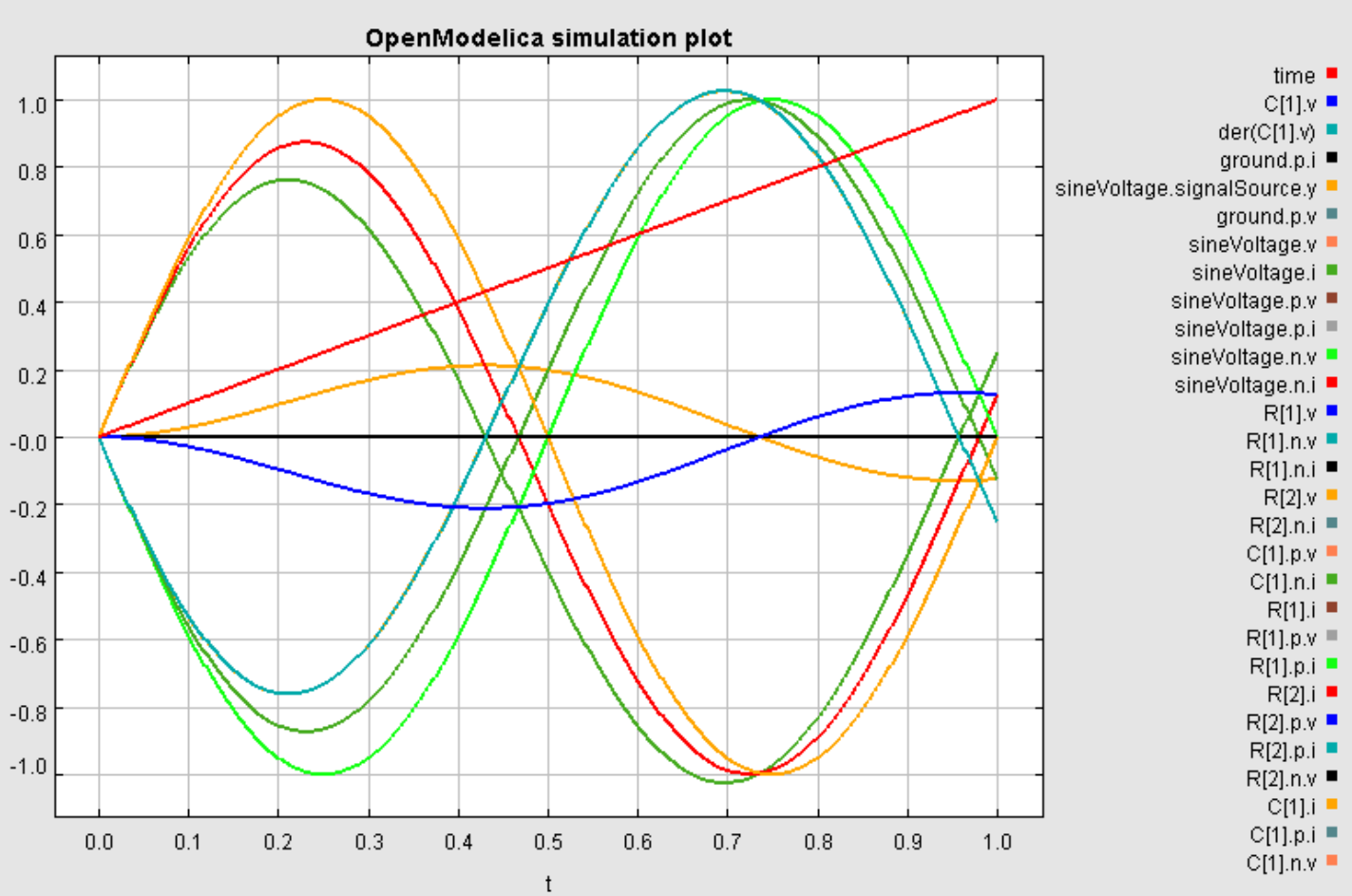
Testmodel:

```
model simpleCircuit
    parameter Integer n=2;
    Modelica.Electrical.Analog.Basic.Resistor R[n];
    Modelica.Electrical.Analog.Basic.Capacitor C[n-1];
    Modelica.Electrical.Analog.Basic.Ground ground;
    Modelica.Electrical.Analog.Sources.SineVoltage sineVoltage;
equation
    for i in 1:n-1 loop
            connect(R[i].p, R[i + 1].n);
            connect(R[i].p, C[i].p);
            connect(C[i].n, ground.p);
    end for;
    connect(sineVoltage.n, R[1].n);
    connect(R[n].p, ground.p);
    connect(sineVoltage.p, ground.p);
end simpleCircuit ;
```



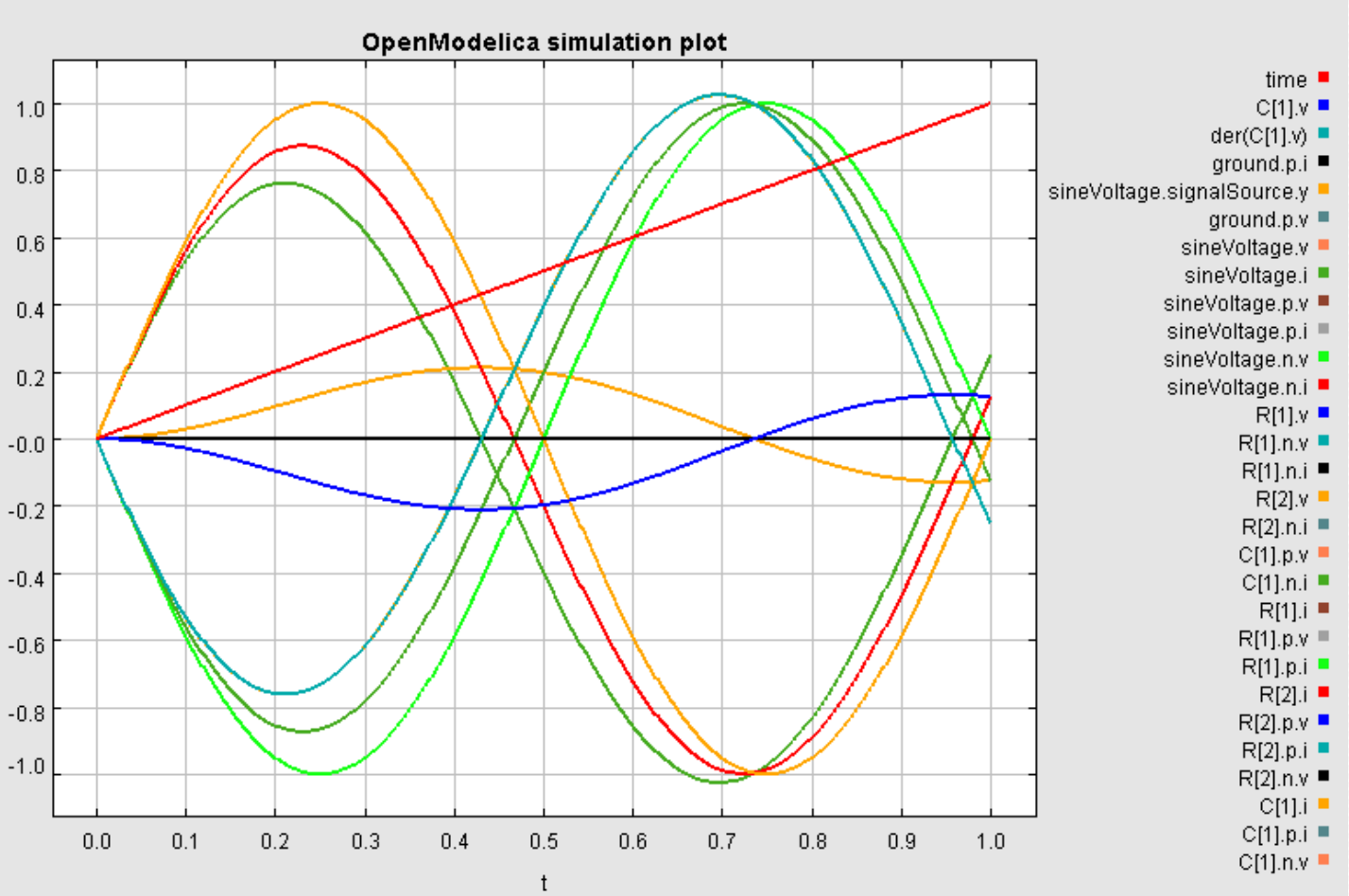
4. Test and Results

Result plot using Finite Differences

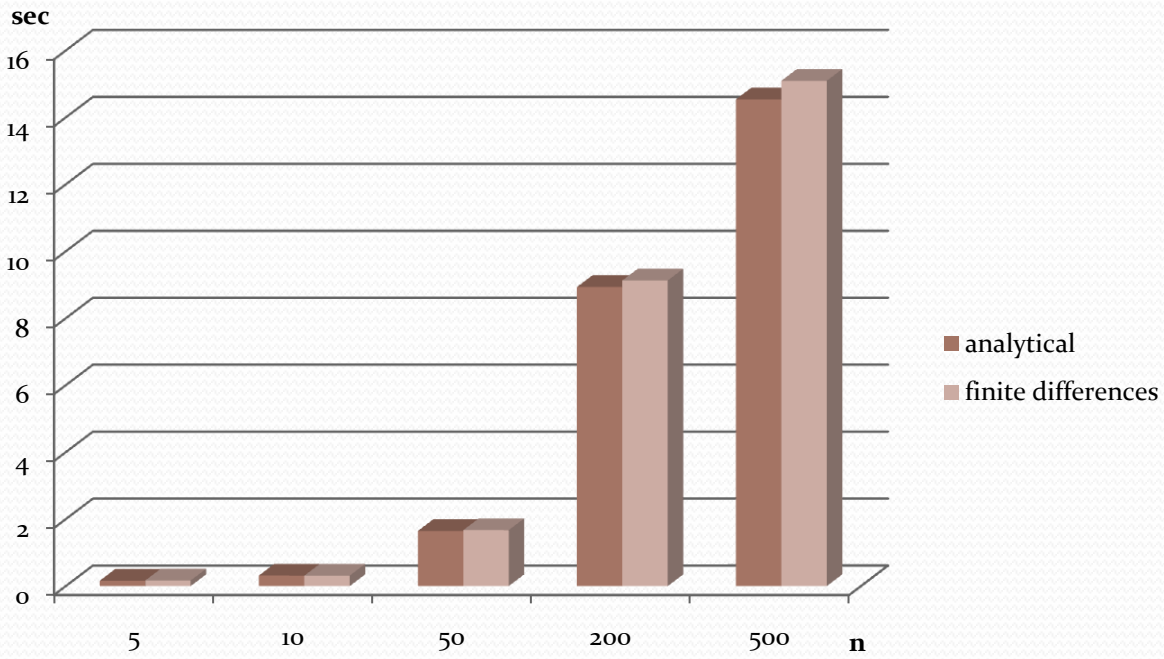


4. Test and Results

Result plot using analytical calculation



4. Test and Results



5. Outlook

- Use sparsity pattern information
- Handle algebraic loops
- In case of external functions: Finite differences only for the variables concerning these functions

Further advantages

- sensitivity analysis
- eigenvalue analysis
- parameter identification
- linearization of non-linear systems
- selection of efficient numerical solution methods (e.g. inline-integration)