Treatment of Discontinuities II

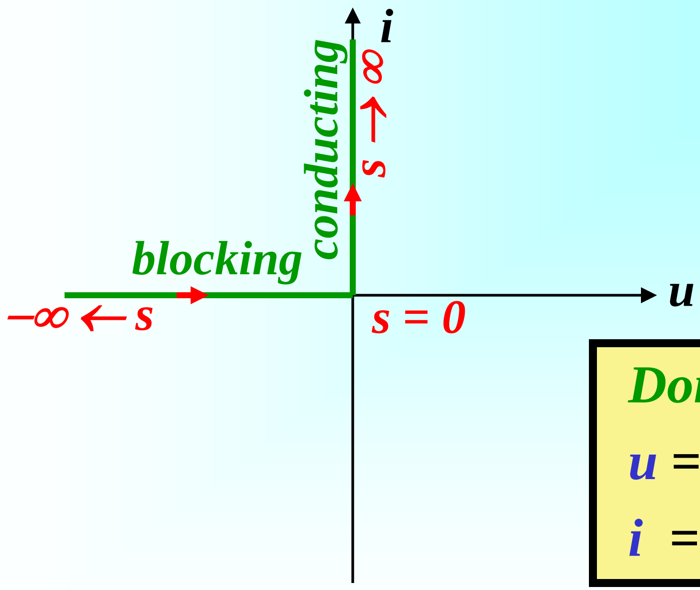
- We shall today once more look at the ***modeling of discontinuous systems***.
- First, an additional method to their mathematical description shall be discussed. This method makes use of a ***parameterized description of curves***.
- Subsequently, we shall deal with the problem of variable causality.
- Finally, a method shall be discussed that permits to solve causality problems elegantly.

Table of Contents

- Parameterized curve descriptions
- Causality of the switch equation
- Leaking diodes
- Singularity of switch equation
- Inline integration
- Causality of inline integration

Parameterized Curve Descriptions

- It is always possible to describe discontinuous functions by means of parameterized curves. This technique shall be illustrated by means of the diode characteristic.



Domain:	Condition:	Equations:
blocking:	$s < 0$	$u = s; i = 0$
conducting:	$s > 0$	$u = 0; i = s$

Domain = **if** *s* < *0* **then** *blocking* **else** *conducting*;
u = **if** *Domain* == *blocking* **then** *s* **else** *0* ;
i = **if** *Domain* == *blocking* **then** *0* **else** *s* ;

The Causality of the Switch Equation I

- Let us consider once more the switch equation in its algebraic form:

$$0 = s \cdot i + (1 - s) \cdot u$$

Switch open: $s = 1$
Switch closed: $s = 0$

- We can solve this equation either for u or for i :

	$u = \dfrac{s}{s-1} \cdot i$	$i = \dfrac{s-1}{s} \cdot u$
Switch open:	*Division by 0!*	$i = 0$
Switch closed:	$u = 0$	*Division by 0!*

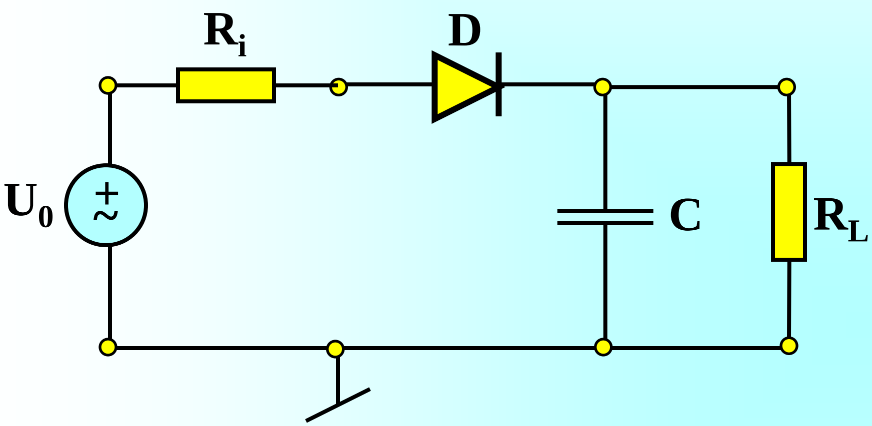
The Causality of the Switch Equation II

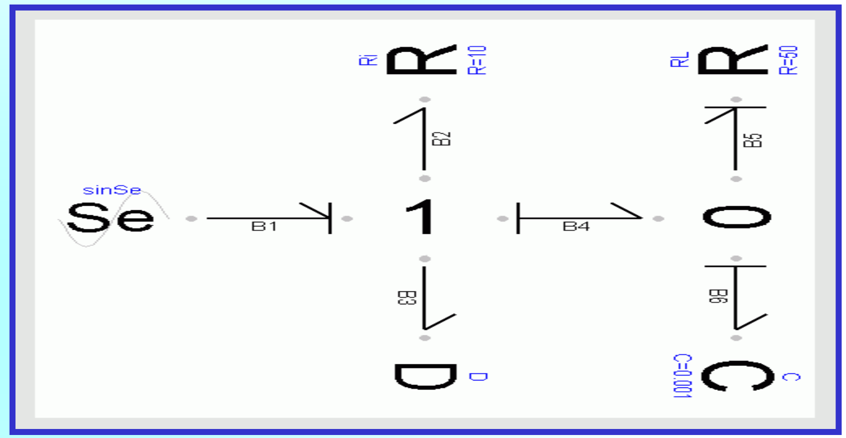
- Neither of the two causal equations can be used in both switch positions. Either one or the other switch position leads to a ***division by 0***.
- This is exactly what happens in the simulation, when the causality of the switch equation is fixed.

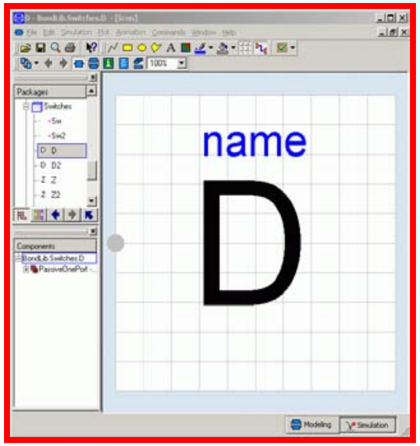
⇒ ***The causality of the switch equation must always be free.***

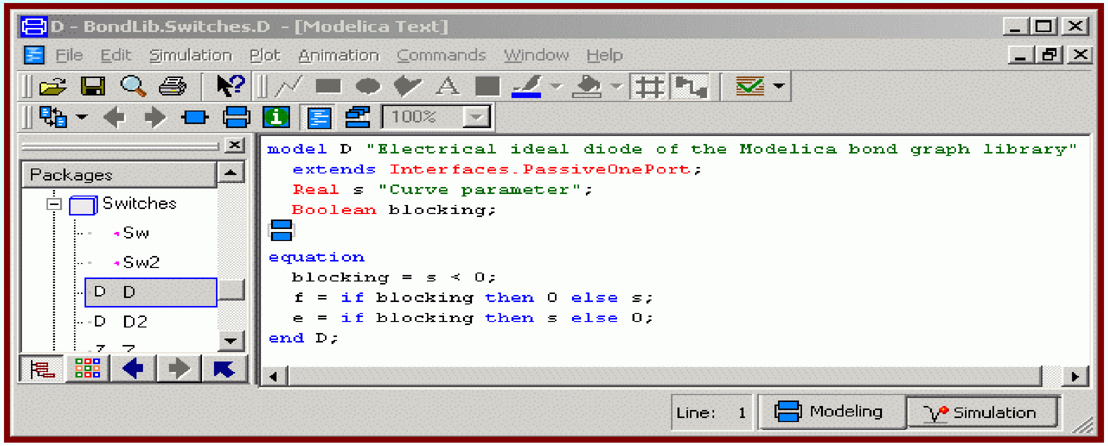
⇒ ***The switch equation must always be placed in an algebraic loop.***

An Example I

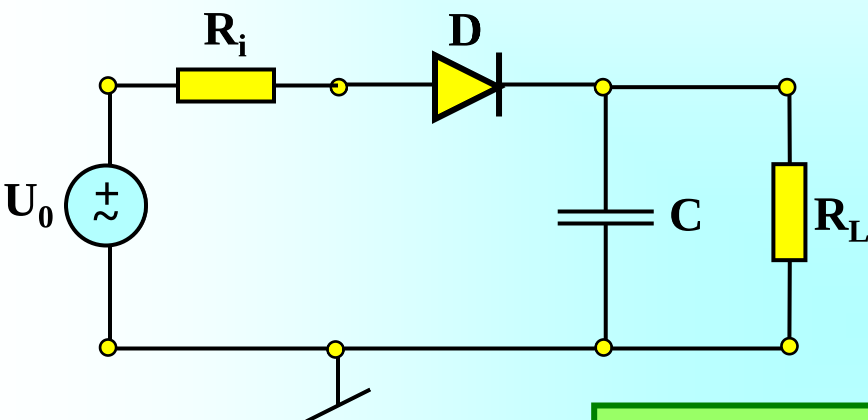

Ri
D
U0
C
RL

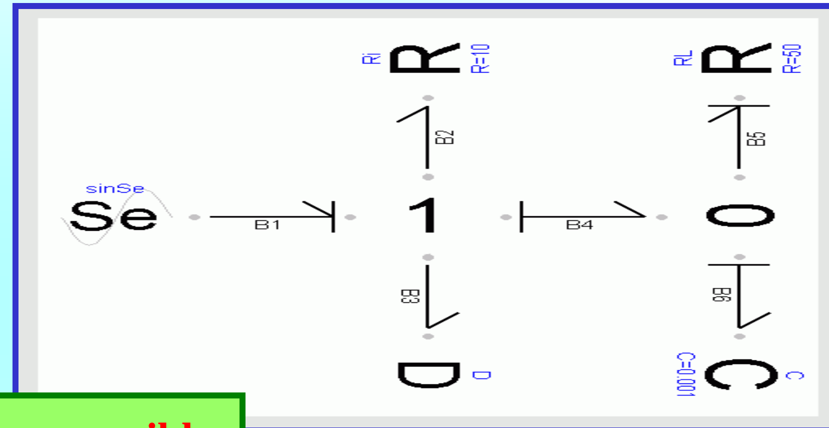

sinSe
Se
B1
1
B4
0
Ri
R
R=10
RL
R
R=50
B2
B5
B3
B6
D
D
C=0.001
C
C

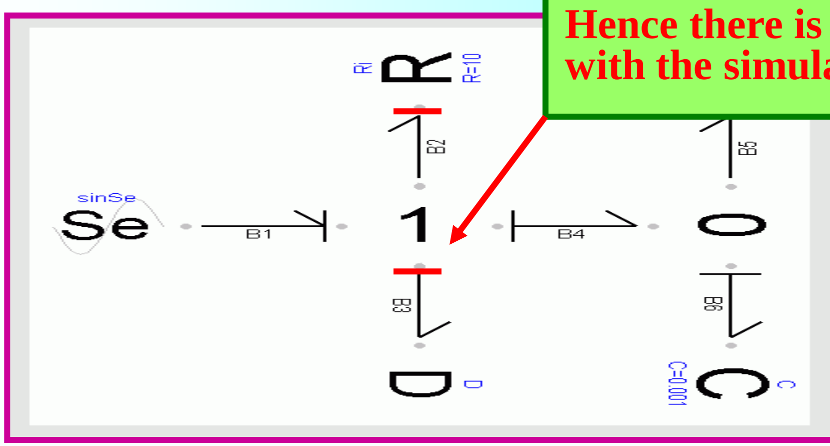

name
D

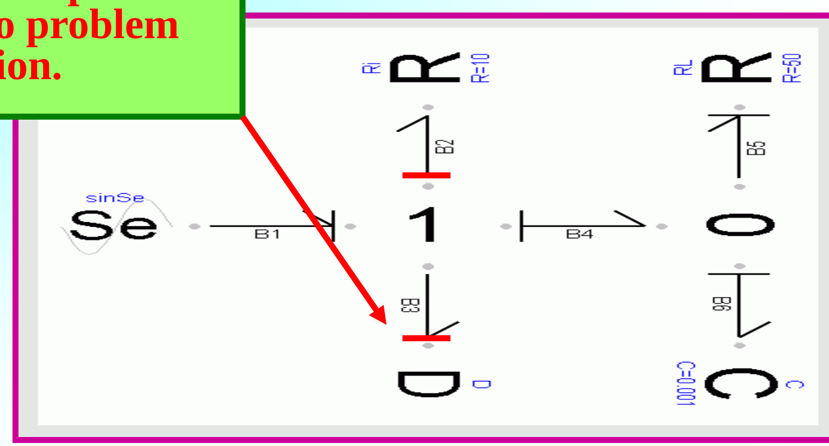

D - BondLib.Switches.D - [Modelica Text]
File Edit Simulation Plot Animation Commands Window Help
100%
Packages
Switches
Sw
Sw2
D D
D D2
model D "Electrical ideal diode of the Modelica bond graph library"
extends Interfaces.PassiveOnePort;
Real s "Curve parameter";
Boolean blocking;
equation
blocking = s < 0;
f = if blocking then 0 else s;
e = if blocking then s else 0;
end D;
Line: 1
Modeling
Simulation

An Example II



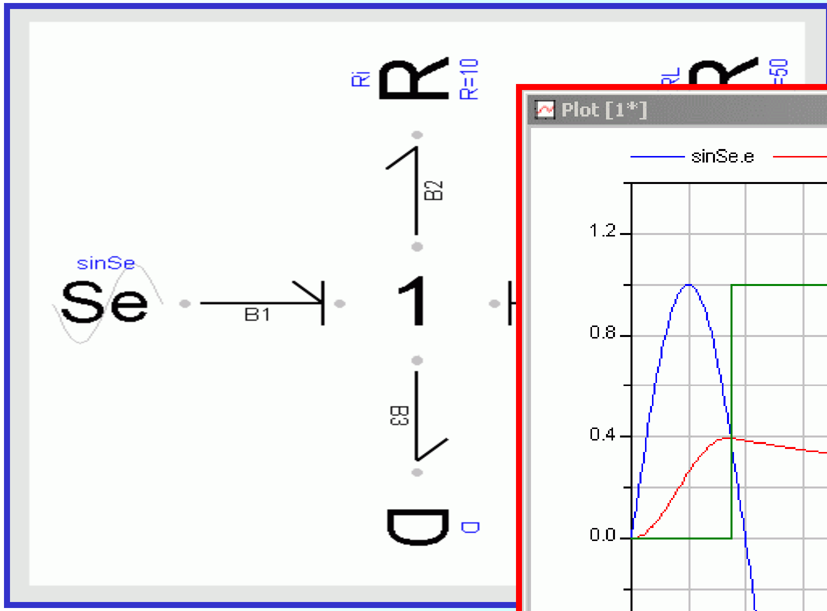


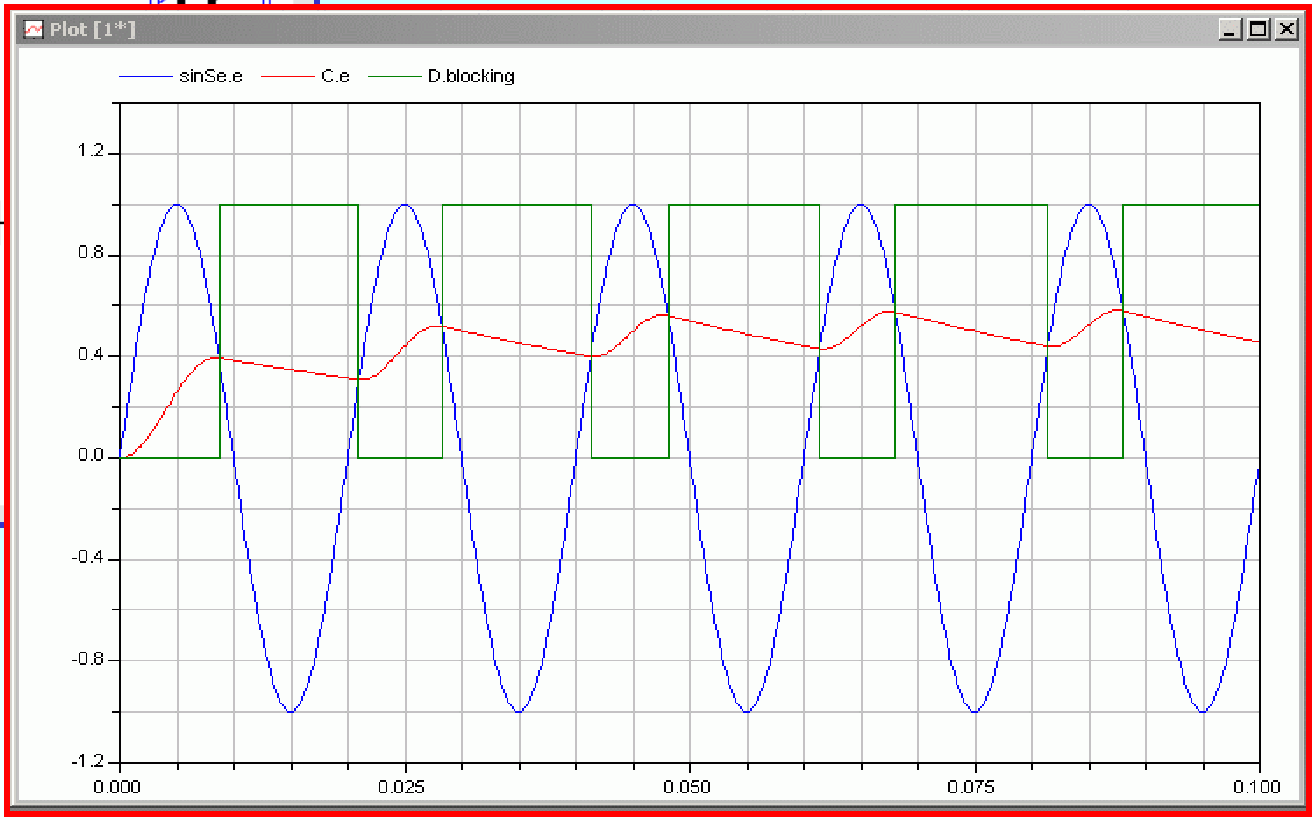




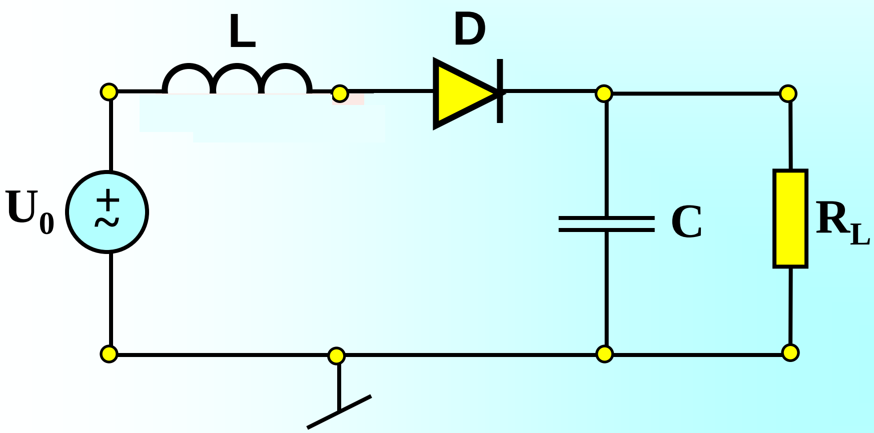
Both causalities are possible. Hence there is no problem with the simulation.

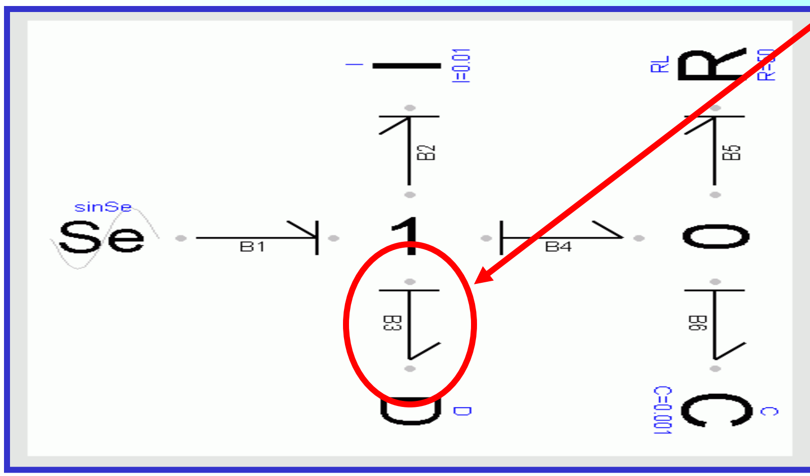
An Example III



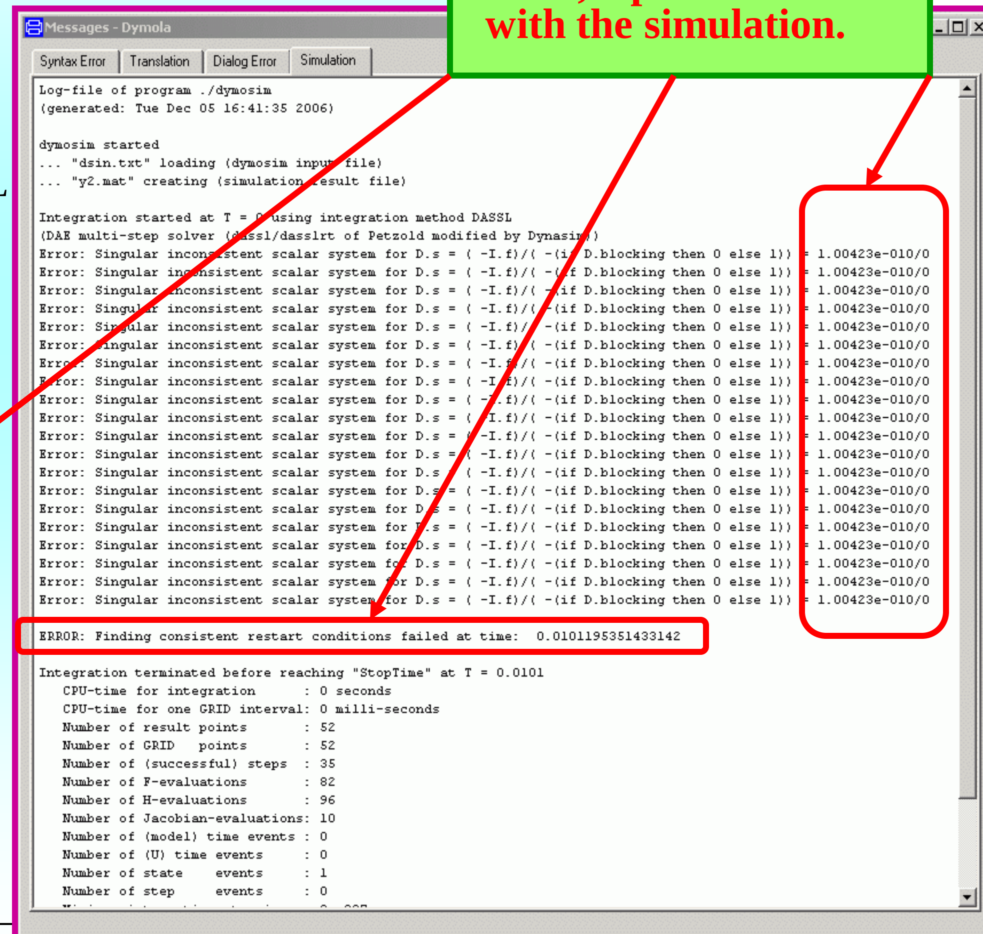


A Second Example





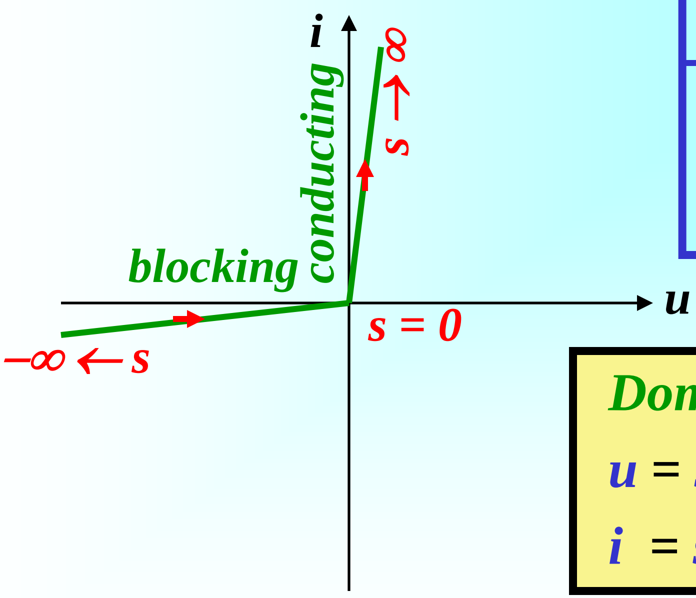
The causality is fixed. Thus, a problem exists with the simulation.



```
Messages - Dymola
Syntax Error | Translation | Dialog Error | Simulation

Log-file of program ./dymosim
(generated: Tue Dec 05 16:41:35 2006)

dymosim started
... "dsin.txt" loading (dymosim input file)
... "y2.mat" creating (simulation result file)

Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslrt of Petzold modified by Dynasim))
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0
Error: Singular inconsistent scalar system for D.s = ( -I.f)/( -(if D.blocking then 0 else 1)) = 1.00423e-010/0

ERROR: Finding consistent restart conditions failed at time:  0.0101195351433142

Integration terminated before reaching "StopTime" at T = 0.0101
   CPU-time for integration      : 0 seconds
   CPU-time for one GRID interval: 0 milli-seconds
   Number of result points       : 52
   Number of GRID   points       : 52
   Number of (successful) steps  : 35
   Number of F-evaluations       : 82
   Number of H-evaluations       : 96
   Number of Jacobian-evaluations: 10
   Number of (model) time events : 0
   Number of (U) time events     : 0
   Number of state     events    : 1
   Number of step      events    : 0
```

Not So Ideal Diode I

- One possibility for circumventing the causality problem consists in defining a ***leakage resistance*** R_{on} for the closed switch, as well as a ***leakage conductance*** G_{off} for the open switch.



Domain:	Condition:	Equations:
blocking:	$s < 0$	$u = s;\ i = G_{off} \cdot s$
conducting:	$s > 0$	$u = R_{on} \cdot s;\ i = s$

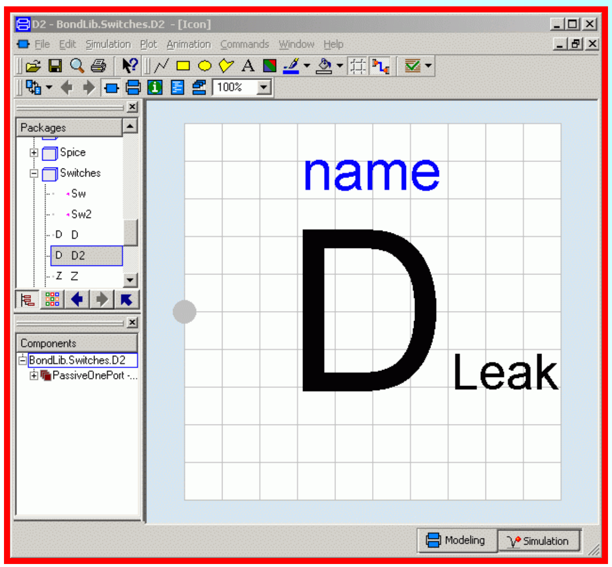
Domain = **if** $s < 0$ **then** ***blocking*** **else** ***conducting***;

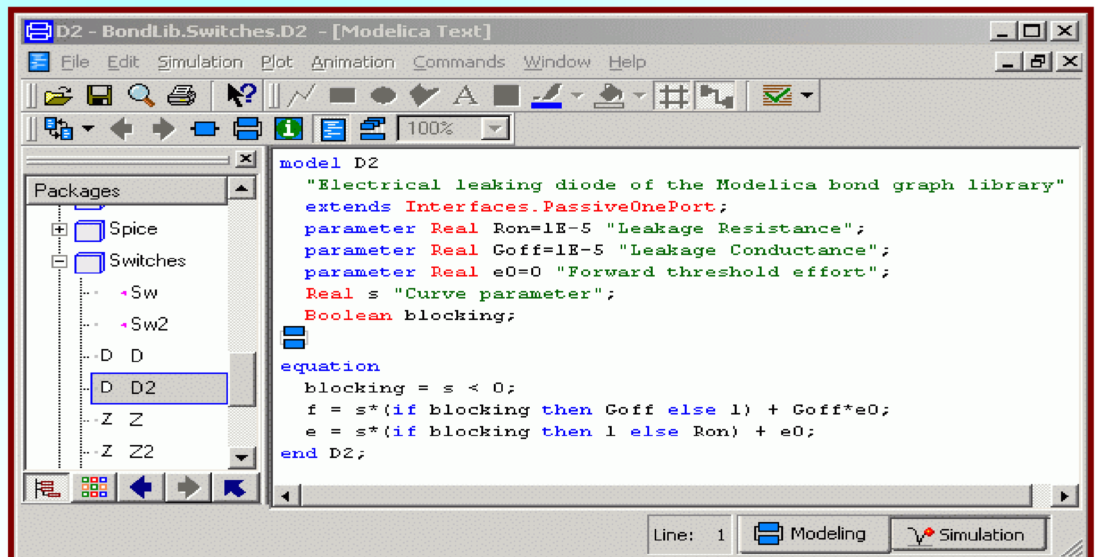
$u = s$*(**if** ***Domain*** == ***blocking*** **then** 1 **else** R_{on});

$i = s$*(**if** ***Domain*** == ***blocking*** **then** G_{off} **else** 1);

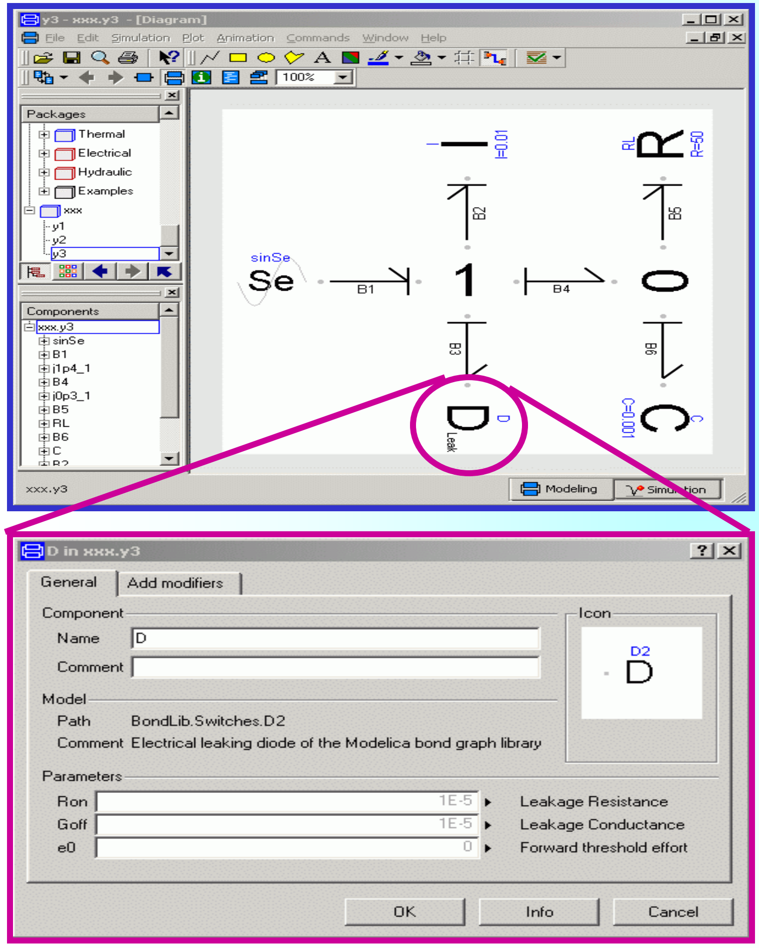
Not So Ideal Diode II

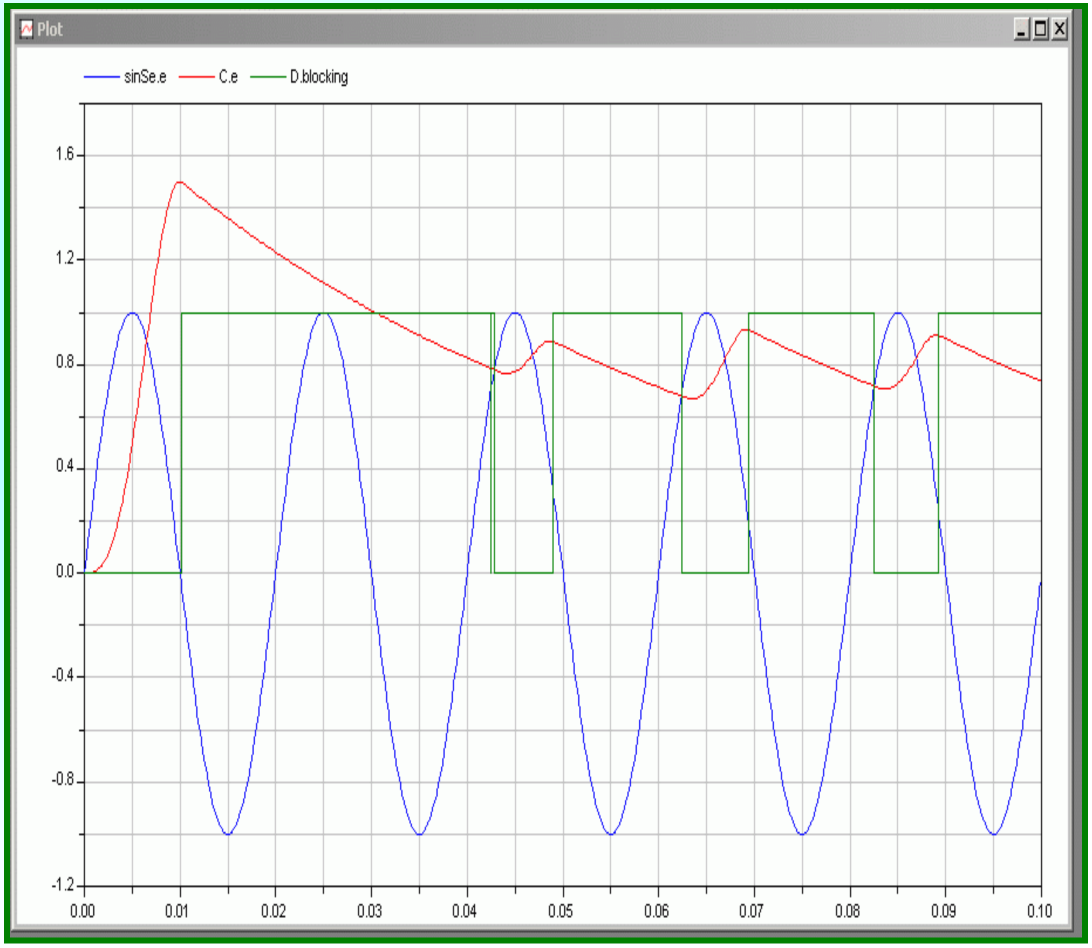
- This is the solution that was chosen in the standard library of ***Modelica***.
- The same solution is also offered in ***BondLib*** in the form of a "leaky" diode model.





Not So Ideal Diode III





Problems I

- For ***electrical applications***, the solution with the leaking diode is frequently acceptable.
- One problem has to do with the numerics. When a circuit using the ideal diode is plagued by division problems, the circuit with the leaking diode leads invariably to a ***stiff system***.
- Stiff systems can be integrated in ***Modelica*** by means of the (standard) ***DASSL integration algorithm***.
- However, this is time consuming and may not be suitable, at least for real-time applications.

Problems II

- In the case of ***mechanical applications***, the method is less suitable, since for example friction characteristics must frequently be computed rather accurately, and since in mechanical applications, the causalities are almost invariably fixed.
- The masses (and inertias) determine all velocities, and the friction as well as spring forces (and torques) must therefore be determined by the ***R***- and ***C***-elements in a pre-set causality.
- Consequently, another solution approach should be sought for these applications.

"Inline" Integration Algorithm

- When using ***Inline Integration***, the integration algorithm is directly substituted into the model equations (or inversely: the model equations are being substituted into the integration algorithm).
- Let us consider an inductor integrated by means of the ***implicit Euler algorithm***.

$$u_L = L \cdot di_L/dt$$
$$i_L(t) = i_L(t-h) + h \cdot di_L(t)/dt$$

$$\Rightarrow \quad i_L(t) = i_L(t-h) + (h/L) \cdot u_L(t)$$

The Causality of Inline Integration

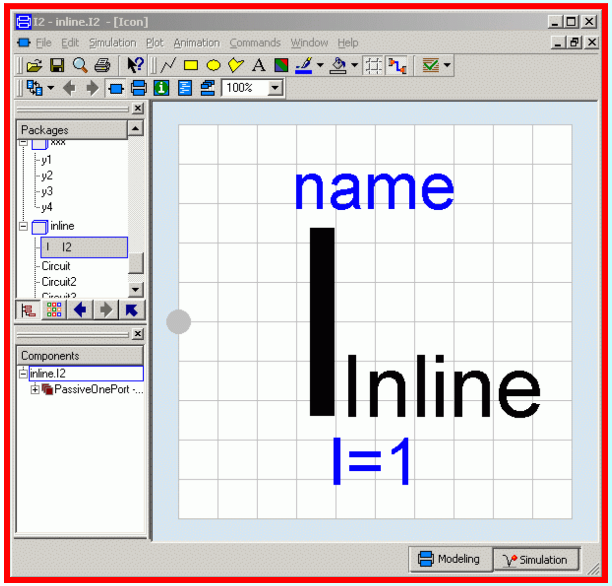
$$i_L(t) = i_L(t-h) + (h/L) \cdot u_L(t)$$

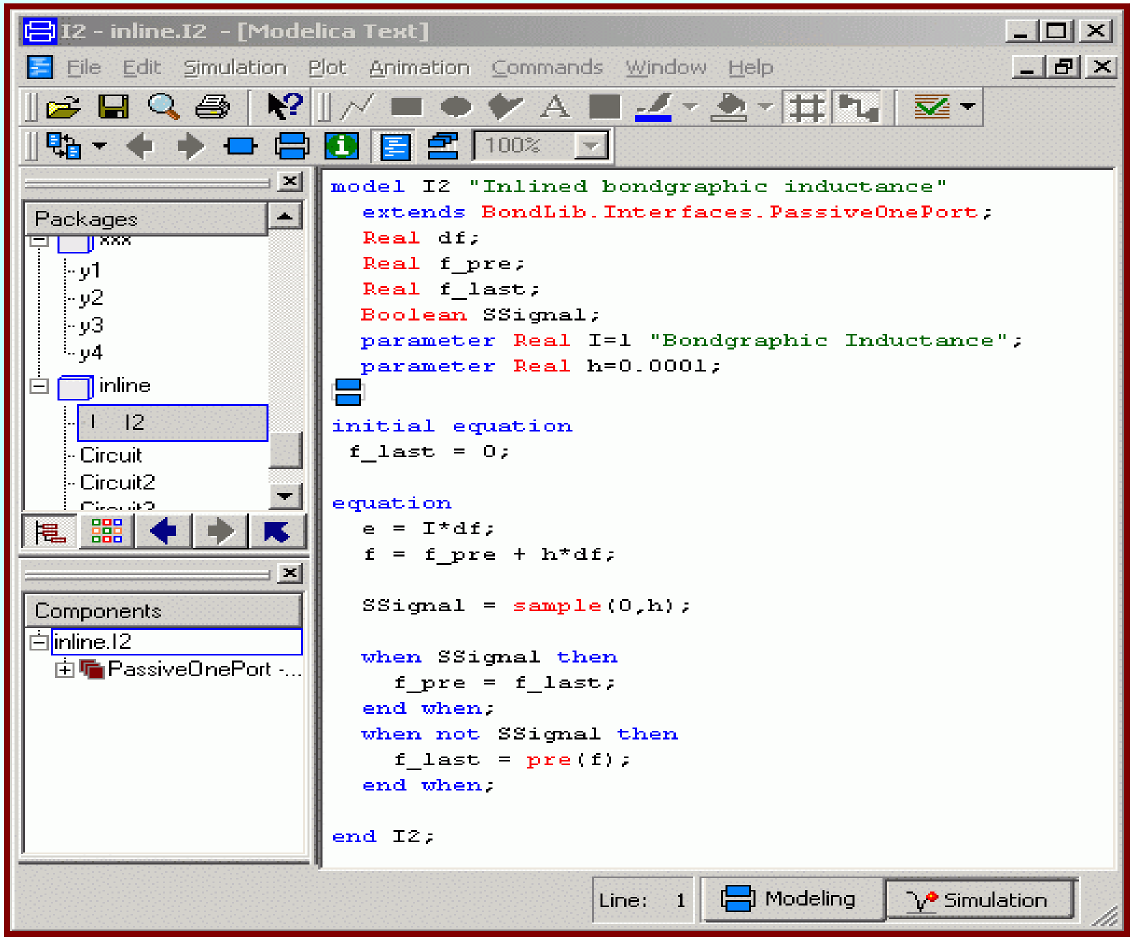
Known, since computed in the past.

This constitutes an algebraic relation between *i* and *u*. This now looks like a resistor. Hence the causality is now free.

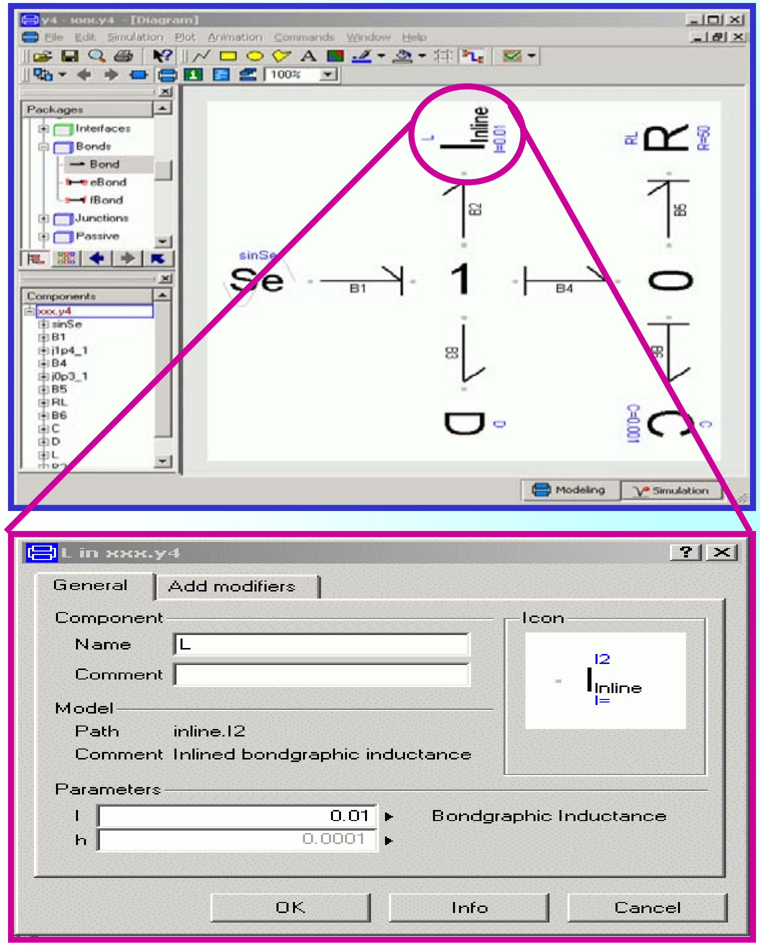
When using the inline integration algorithm, the causalities of the so integrated storage elements are being freed up. Consequently, the division by zero problem disappears.

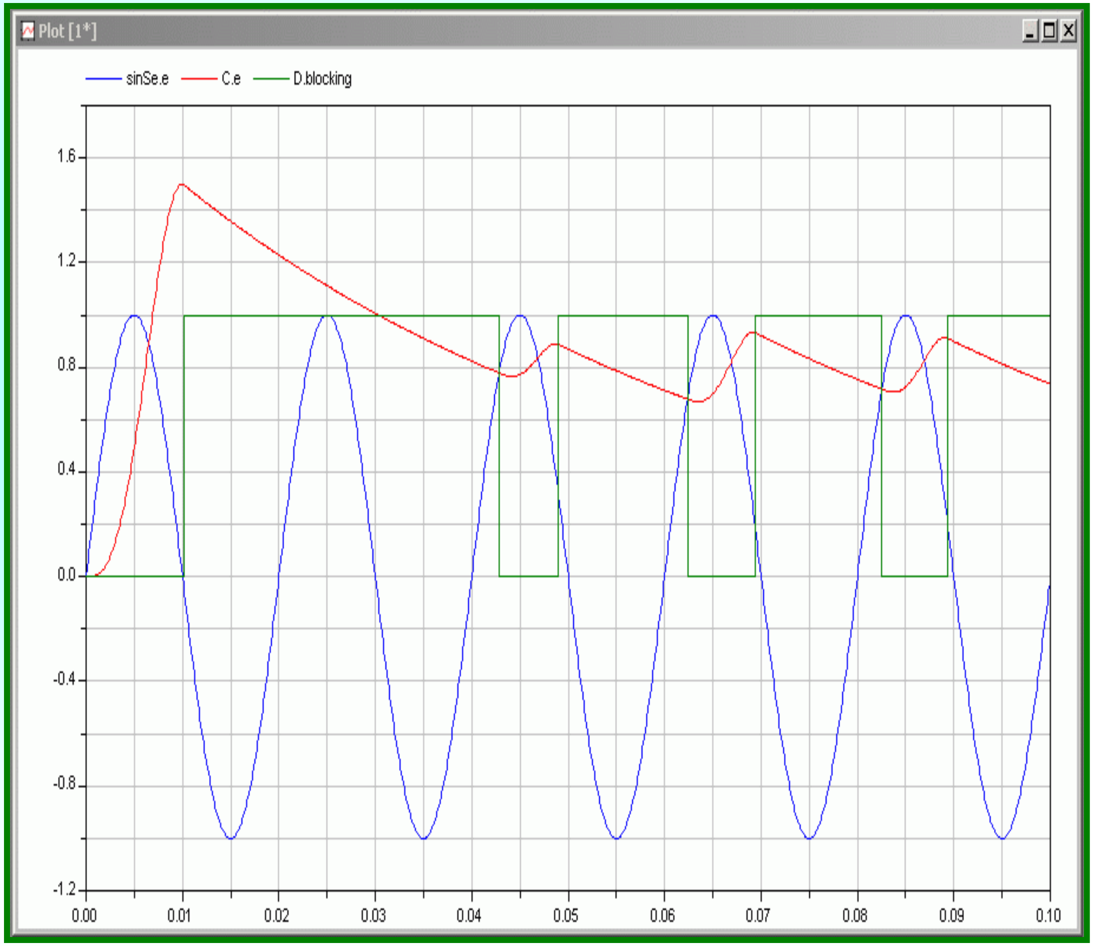
Ideal Diode With Inline Integration I





Ideal Diode With Inline Integration II





References I

- Elmqvist, H., M. Otter, and F.E. Cellier (1995), "Inline integration: A new mixed symbolic/numeric approach for solving differential-algebraic equation systems," *Proc. ESM'95, European Simulation Multi-conference*, Prague, Czech Republic, pp. xxiii – xxxiv.

- Otter, M., H. Elmqvist, and S.E. Mattsson (1999), "Hybrid modeling in Modelica based on the synchronous data flow principle," *Proc. CACSD'99, Computer-Aided Control System Design*, Hawaii.

References II

- Krebs, M. (1997), *Modeling of Conditional Index Changes*, MS Thesis, Dept. of Electr. & Comp. Engr., University of Arizona, Tucson, AZ.
- Cellier, F.E. and M. Krebs (2007), "Analysis and simulation of variable structure systems using bond graphs and inline integration," *Proc. ICBGM'07, 8th Intl. Conf. Bond Graph Modeling and Simulation*, San Diego, CA, pp. 29-34.