TEARING IN BOND GRAPHS WITH DEPENDENT STORAGE ELEMENTS

Wolfgang Borutzky
Dept. of Computer Science
Cologne Polytechnic
D-51643 Gummersbach
Germany
wolfgang.borutzky@uni-koeln.de

Francois Cellier
Dept. of Electr. & Comp. Engr.
University of Arizona
Tucson, AZ 85721
U. S. A.
Cellier@ECE.Arizona.Edu

Abstract

In a previous paper [1] we have shown that information about possible tearing variables added to a Bond Graph (BG) with dependent storage elements can be exploited by a program like Dymola [6], if the stores are replaced by a so-called resistive companion model originally introduced in integrated circuit analysis [13].

In this paper it is shown how tearing information added to a BG with causal paths between stores of the same type can be exploited directly by a program like Dymola allowing for a mixed symbolic, numerical solution of the underlying Differential Algebraic Equation (DAE) system. For didactic reasons, the method is explained by means of some fairly small examples. Nevertheless, it is quite general.

Keywords: Bond Graphs, dependent storage elements, DAE systems, residual sinks, tearing.

1 Introduction

For many engineering systems, e. g. for multibody systems (MBS) reasonable modeling assumptions lead to algebraic dependencies between some of the state variables such that only a smaller number of them is independent. Consequently, the mathematical model has the form of a Differential Algebraic Equations (DAE) system. Its numerical solution requires an appropriate package like the DASSL code [3]. Before DAE solvers have become popular, a commonly used method was to turn the DAE system into an explicit Ordinary Differential Equations (ODE) system by adding small resistors in appropriate locations such that causal paths between storage ports of the same type vanish. The disadvantage of that approach is that equations may become stiff. A different option we advocate in this paper is to add information to the Bond Graph (BG) such that a program like Dymola [6] can apply its formulae manipulation capabilities to the model equations to allow for a subsequent efficient numerical solution. The key issue is tearing that is, the partitioning of an overall large system of equations into a number of smaller systems. The method was introduced by G. Kron [11] already in the early 60's and is not limited to linear equations. In order to be able split a system into smaller subsystems, so-called tearing variables must be determined somehow. Once these tearing variables are known, all other non-state variables can be computed by means of the tearing variables and the independent state variables. Of course, for general non-linear systems a (modified) Newton iteration will be needed, while for systems that are linear, at least in regard to the tearing variables, the original DAEs can be transformed into an explicit ODE system.

The objective we are aiming at is to give information to a program at the BG level such that tearing based formulae manipulation can take place automatically. For a system of hundreds of equations and variables derived automatically from a BG it is almost impossible to inspect the generated equations and to decide which variables can be used to tear the overall set of equations into a number of smaller sets. For fairly small systems the modeler might represent variables and their functional relations by a so-called dependency graph in order to see which variables are involved in which algebraic loops. However, since it is an NP complete problem to find a *minimal* set of tearing variables that cut all loops [12], the modeler might be seeking not for a minimal but *small* number of tearing variables that reduce the computational amount, at least to some extent and a method that is generally applicable. That is the issue we are addressing in the sequel. First, for a better understanding, the principle of tearing is briefly recalled. More information on tearing may be found in [11], [8].

2 Tearing

If an index-one DAE system is to be reduced symbolically to an explicit ODE system, then after analytical differentiation of the constraint equations a system must be solved that may be of considerable size. For simplicity reasons we assume a set of linear equations, although tearing is not limited to the linear case. Suppose that it has been decided which components of the vector $\mathbf{z}$ of unknowns including the dependent state variables are the tearing variables. If they are grouped into a subvector $\mathbf{z}_2$, then permutation matrices **P** and **Q** can be determined such that the system can be written in the following way:

$$\begin{pmatrix} \mathbf{L} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} \tag{1}$$

If **L** is non-singular, the symbolic solution of the equation

$$\mathbf{L}\,\mathbf{z}_1 \;+\; \mathbf{A}_{12}\,\mathbf{z}_2 \;=\; \mathbf{b}_1 \tag{2}$$

with respect to $\mathbf{z}_1$ is not expensive, since the submatrix **L** is least lower block triangular. If its solution is substituted into

$$\mathbf{A}_{21}\,\mathbf{z}_1 \;+\; \mathbf{A}_{22}\,\mathbf{z}_2 \;=\; \mathbf{b}_2\;, \tag{3}$$

then we obtain an equation that determines the subvector $\mathbf{z}_2$ of tearing variables. Once these are known, the other subvector $\mathbf{z}_1$ can be computed at low cost. Obviously, if we are lucky to find a *small* number of tearing variables, then only a *small* subsystem must be solved, and all other unknowns $\mathbf{z}_1$ are obtained essentially by evaluation of the expressions in eq. (2). The question is, how such a set of tearing variables can be found. In the following we will show that information about tearing can be expressed already in the BG.

3 The underlying idea

Let us start with an example by considering a Bond Graph with a tree structure with five capacitances as shown in Fig. 1. Application of the standard Sequential Causality Assignment Procedure (SCAP) [10] reveals that two of the five capacitors must have derivative causality. Consequently, their co-energy variables are not state variables. This is indicated to the program Dymola by the command

- **variable unknown** C3.e C5.e

(C3.e denotes the effort of capacitance C3.) Since there are more unknown variables than equations, the right hand side of the ODE's for the three independent state variables cannot be computed. For that reason, by means of the command

- **differentiate**

we invoke an implementation of Pantelides' algorithm [14], which provides the missing equations. The equations derived from the BG can be written in the DAE form

$$\begin{pmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} \dot{\mathbf{e}}_i \\ \dot{\mathbf{e}}_d \end{pmatrix} + \begin{pmatrix} \mathbf{G}_{11} & \mathbf{0} \\ \mathbf{G}_{21} & \mathbf{G}_{22} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{e}_i \\ \mathbf{e}_d \end{pmatrix} \;=\; \begin{pmatrix} \mathbf{F} \\ \mathbf{0} \end{pmatrix} \tag{4}$$

with $\mathbf{e}_i = (e_1, e_2, e_4)^T$, $\mathbf{e}_d = (e_3, e_5)^T$ and $\mathbf{F} = (E/R, 0, 0)^T$. As the matrix

$$\mathbf{C} \;=\; \mathbf{C}_{11} \;-\; \mathbf{C}_{12}\mathbf{G}_{22}^{-1}\mathbf{G}_{21} \tag{5}$$

```
! SORTED AND SOLVED EQUATIONS
  one1.       [R.e] = Cnet.E - C1.e
  R.          [one1.f1] = R.e/R.R

! SYSTEM OF 9 SIMULTANEOUS EQUATIONS
! UNKNOWN VARIABLES
! C4.dere
! C5.dere
! one3.f1
! C2.dere
! C1.f
! C1.dere
! C3.dere
! C3.f
! one2.f1
!
! EQUATIONS
! C4.         C4.C*[C4.dere] = one3.f1
! one3.       [C5.dere] = C3.dere - C4.dere
! C5.         C5.C*C5.dere = [one3.f1]
! C2.         C2.C*[C2.dere] = one2.f1
! Cnet.       [C1.f] + one2.f1 = one1.f1
! C1.         C1.C*[C1.dere] = C1.f
! one2.       [C3.dere] = C1.dere - C2.dere
! C3.         C3.C*C3.dere = [C3.f]
! Cnet.       C3.f + one3.f1 = [one2.f1]
!
! INCIDENCE MATRIX
!
! X X
! XX    X
!  XX
!    X    X
!     X   X
!     XX
!    X XX
!       XX
!   X    XX
!
```

Figure 2: Dymola output for the BG with five capacitances

is non-singular, the index of that DAE system is one. Hence, the sum of all efforts at the second and the third one junction

$$e_1 \;=\; e_2 + e_3 \tag{6.a}$$

$$e_3 \;=\; e_4 + e_5 \tag{6.b}$$

is differentiated only once.

Now, after causality assignment Dymola comes up with a system of nine simultaneous equations and solves them symbolically without exploiting the system structure in order to produce the state equations for the independent capacitances C_1, C_2, C_4. The corresponding Dymola output file including the incidence matrix of that system of simultaneous equations is given partly in Fig. 2. In that figure a variable in square brackets means that the variable is considered the unknown in that equation.

By looking at the BG it can be seen immediately that if the flows f_3 and f_5 would be known, all other algebraic variables could be computed from these flow variables and the state variables. Of course, this is expressed by the incidence matrix as well (cf. Fig. 2) , but it is less ap-

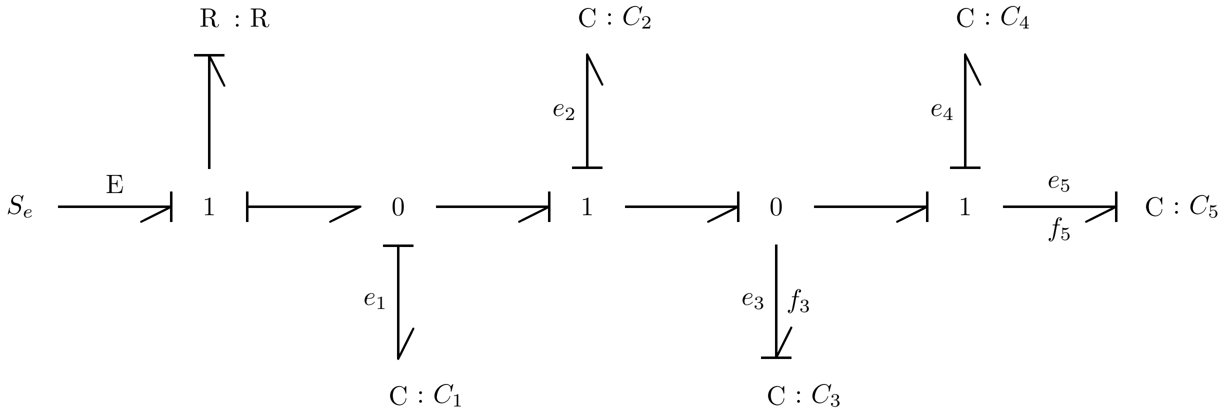

Figure 1: Bond Graph with a tree structure with dependent storage elements

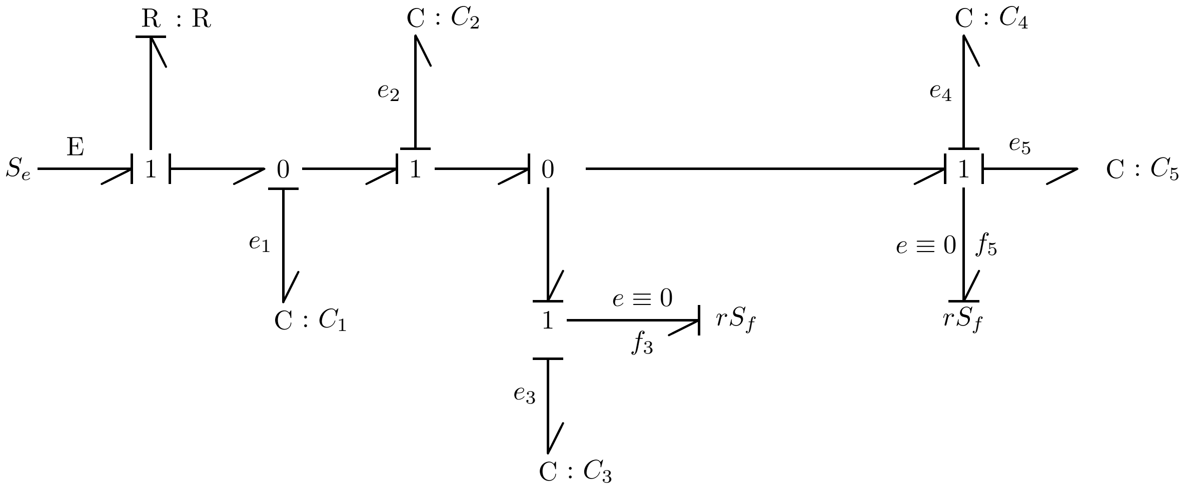

Figure 3: BG with residual flow sinks added

parent. In regard to the BG the condition becomes even more evident, if we add residual flow sinks rS_f to the BG (cf. Fig. 3). Residual sinks have been introduced in [1]. They simply represent a Lagrange multiplier λ. There are two types of residual sinks. Residual flow sinks provide a flow such that their corresponding effort vanishes, while residual effort sinks are the dual elements. Apparently, by adding such sinks to the BG we do not modify the model. Such elements have been used as well by Gawthrop and Smith [9] in order to make algebraic variables associated with algebraic loops explicit on the BG. They have termed them sensing sources and label them SS.

In Dymola the role of a residual flow sink is expressed by means of a residue() operator:

$$e = residue(f) \ , \tag{7}$$

which means that f is a possible tearing variable, while residue(f) is made to be zero. By following an idea of H. Elmqvist [7] we modify eq. (7) for a reason that will become obvious shortly:

$$e = Time * residue(f) \tag{8}$$

From the BG in Fig. 3 we see that the sum of efforts in equations (6.a), (6.b) become:

$$\begin{aligned} e_1 &= e_3 + e_2 + Time * residue(f_3) && (9.a) \\ e_3 &= e_5 + e_4 + Time * residue(f_5) && (9.b) \end{aligned}$$

Apparently, since residue(...) vanishes, two capacitances are still dependent. When Pantelides' algorithm differentiates these equations, we obtain

$$\begin{aligned} der(e_1) &= der(e_3) + der(e_2) + \\ & \quad residue(f_3) && (10.a) \\ der(e_3) &= der(e_5) + der(e_4) + \\ & \quad residue(f_5) && (10.b) \end{aligned}$$

The last term in the differentiated equations is the reason why we slightly changed the constitutive equation of the residual flow sink. Apparently, the algebraic constraints have just been differentiated. However, for Dymola the meaning is that f_3 and f_5 are possible tearing variables and equations (10.a), (10.b) are the associated residual equations that determine the tearing variables. That is, after solving the residual equations for f_3 and f_5, the right hand side of the ODE system for e_1, e_2, e_4 can be computed. As can be seen from the corresponding Dymola output (cf. Fig. 4), adding the residual sinks to the BG, enables Dymola to identify tearing variables and their associated residual equations. With that information it solves symbolically for the tearing variables, and after that step determines all remaining non-state variables by means of the tearing variables and the state variables (C1.e, C2.e, C4.e).

By writing the simultaneous equations in Fig. 4 as a matrix equation, it can be seen that the system indeed, has the form of equation (1)

4 The general case

In the previous section we considered a small and fairly simple example. The method however, can be well applied to more general cases. In the above example a dependent capacitance has been replaced by a C-element and a residual flow sink, both attached to a one junction. Correspondingly, if dependent inertances occur as they do for instance, in BGs of rigid multibody systems, they are replaced by an inertance and a residual effort sink, both connected to a zero junction (cf. Fig. 5).

First, for simplicity let us exclude the case of kinematic loops in mechanical systems. Now, if $\mathbf{x}_i$ denotes the vector of state variables and $\mathbf{x}_d$ the vector of output variables of the dependent stores, then assuming that the state variables depend linearly on the outputs $\boldsymbol{\lambda}$ of the residual sinks, equations can be derived straightforward from a BG in the following form:

$$\begin{aligned} \mathbf{D}_i\,\dot{\mathbf{x}}_i &= \mathbf{G}\cdot\boldsymbol{\lambda} + \mathbf{g}(\mathbf{x}_i, \mathbf{x}_d, \mathbf{u}) && (11.a)\\ \mathbf{D}_d\,\dot{\mathbf{x}}_d &= \boldsymbol{\lambda} && (11.b)\\ \mathbf{x}_d &= \mathbf{B}\,\mathbf{x}_i + \mathit{Time} * \mathit{residue}(\boldsymbol{\lambda}) && (11.c) \end{aligned}$$

In many cases the storage elements are not directly coupled such that the matrices $\mathbf{D}_i$ and $\mathbf{D}_d$ are diagonal. Equation (11.a) represents just the state equations of all independent storage elements, in which the Lagrange multiplier $\boldsymbol{\lambda}$ is treated like a known input. True external system inputs are denoted by the vector $\mathbf{u}$. Equation (11.b) is due to the above mentioned replacement of dependent storage elements (cf. Fig. 5) and eq. (11.c) is just a summation of efforts at one junctions, to which residual flow sinks are attached, or a summation of flows at zero junctions respectively, to which a residual effort sink is connected. Since the last term vanishes, the equation expresses the algebraic dependencies between storage elements. Differentiation of that constraint equation gives the residual equation for the tearing variable $\boldsymbol{\lambda}$

$$\dot{\mathbf{x}}_d = \dot{\mathbf{B}}\,\mathbf{x}_i + \mathbf{B}\,\dot{\mathbf{x}}_i + \mathit{residue}(\boldsymbol{\lambda}) \qquad (12)$$

Substitution of $\dot{\mathbf{x}}_i$ and $\dot{\mathbf{x}}_d$ by means of equations (11.a), (11.b) yields the linear system determining the Lagrange

```
! SORTED AND SOLVED EQUATIONS
  one1.      [R.e] = Cnet.E - C1.e
  R.         [one1.f1] = R.e/R.R

! SYSTEM OF 9 SIMULTANEOUS EQUATIONS
! UNKNOWN VARIABLES
! C2.dere
! C4.dere
! RSf1.f
! C3.f
! RSf2.f
! C5.dere
! C3.dere
! C1.dere
! C1.f
!
! EQUATIONS
! C2.        C2.C*[C2.dere] = RSf1.f
! C4.        C4.C*[C4.dere] = RSf2.f
! Cnet.      C1.f + [RSf1.f] = one1.f1
! C3.        C3.C*C3.dere = [C3.f]
! Cnet.      C3.f + [RSf2.f] = RSf1.f
! C5.        C5.C*[C5.dere] = RSf2.f
! one3.      [C3.dere] = C5.dere + C4.dere +
!                    RSf2.residuef
! one2.      [C1.dere] = C3.dere + C2.dere +
!                    RSf1.residuef
! C1.        C1.C*C1.dere = [C1.f]
!
! TEARING VARIABLES AND RESIDUES
! RSf1.f                            RSf1.residuef
! RSf2.f                            RSf2.residuef
!
! SOLVED SYSTEM OF EQUATIONS
             Q101 = 1/C3.C
             Q102 = 1/C1.C
             Q103 = 1/C2.C
             Q104 = 1/C4.C
             Q105 = 1/C5.C
             Q106 = Q102 + Q103 + Q101
             Q107 = Q101 + Q104 + Q105
             Q109 = Q106*Q107 - Q101*Q101
             Q108 = Q102*one1.f1
             Q110 = - Q108
             Q111 = Q107*Q110
             RSf1.f = - Q111/Q109
             Q113 = Q101*Q110
             RSf2.f = - Q113/Q109

! NON-TORN VARIABLES
  Cnet.      [C1.f] = one1.f1 - RSf1.f
  C1.        [C1.dere] = C1.f/C1.C
  C2.        [C2.dere] = RSf1.f/C2.C
  Cnet.      [C3.f] = RSf1.f - RSf2.f
  C3.        [C3.dere] = C3.f/C3.C
  C4.        [C4.dere] = RSf2.f/C4.C
  C5.        [C5.dere] = RSf2.f/C5.C

! END OF SYSTEM OF SIMULTANEOUS EQUATIONS

  RSf1.      [RSf1.e] = Cnet.Time*RSf1.residuef
  one2.      [C3.e] = C1.e - (C2.e + RSf1.e)
  RSf2.      [RSf2.e] = Cnet.Time*RSf2.residuef
  one3.      [C5.e] = C3.e - (C4.e + RSf2.e)
! END OF SORTED AND SOLVED EQUATIONS
```

Figure 4: Dymola output for the BG with residual sinks

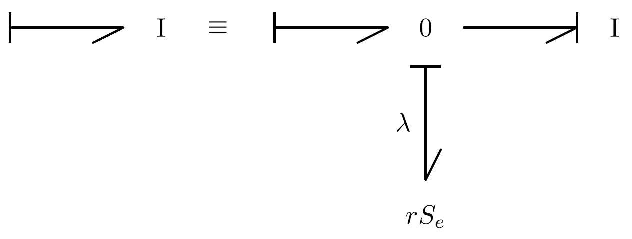

Figure 5: Replacement of a dependent inertance

multipliers

$$[\mathbf{I} - \mathbf{D}_d \mathbf{B} \mathbf{D}_i^{-1} \mathbf{G}] \boldsymbol{\lambda} = \mathbf{D}_d \dot{\mathbf{B}} \mathbf{x}_i + \mathbf{D}_d \mathbf{B} \mathbf{D}_i^{-1} \mathbf{g} \quad (13)$$

Once $\boldsymbol{\lambda}$ is known, the derivatives of the state variables $\dot{\mathbf{x}}_i$ can be easily computed.

If there are kinematic loops in a mechanical system, they introduce an algebraic dependency between the components of the vector $\mathbf{x}_i$. If we add residual sinks representing Lagrange multipliers $\boldsymbol{\lambda}_{kl}$ to the kinematic loops, and if we partition the vector $\mathbf{x}$ into a subvector $\mathbf{x}_{ii}$ of state variables and a dependent subvector $\mathbf{x}_{id}$, then a weighted sum of both subvectors vanishes at those junctions, to which the additional residual sinks have been attached:

$$\mathbf{C}_1 \mathbf{x}_{ii} + \mathbf{C}_2 \mathbf{x}_{id} = Time * residue(\boldsymbol{\lambda}_{kl}) , \quad (14)$$

which can be written in the form

$$\mathbf{C} \mathbf{x}_i = residue(\boldsymbol{\lambda}_{kl}) \quad (15)$$

with $\mathbf{C} = (\mathbf{C}_1 \,|\, \mathbf{C}_2)$. In order to distinguish between those residual sinks added to the dependent storage elements and those inserted into the kinematic loops, we replace $\boldsymbol{\lambda}$ in eq. (12) by $\boldsymbol{\lambda}_d$ and partition $\boldsymbol{\lambda}$ in eq. (11.a) into subvectors $\boldsymbol{\lambda}_d$ and $\boldsymbol{\lambda}_{kl}$. Correspondingly, the matrix $\mathbf{G}$ can be partitioned into two submatrices: $\mathbf{G} = (\mathbf{G}_1 \,|\, \mathbf{G}_2)$. Now, if the above eq. (15) is differentiated

$$\dot{\mathbf{C}} \mathbf{x}_i + \mathbf{C} \dot{\mathbf{x}}_i = residue(\boldsymbol{\lambda}_{kl}) , \quad (16)$$

then equations (12) and (16) are the residual equations for the tearing variables $\boldsymbol{\lambda}_d$ and $\boldsymbol{\lambda}_{kl}$. Substituting $\dot{\mathbf{x}}_i$ and $\dot{\mathbf{x}}_d$ gives lengthy expressions for the unknown tearing variables $\boldsymbol{\lambda}_d$ and $\boldsymbol{\lambda}_{kl}$ similar to that in eq. (13). As a result, systems with kinematic loops can be treated by the tearing approach in the same manner as those without kinematic loops. In case of kinematic loops only additional residual sinks are needed.

5 Applications of the method

The presented tearing approach has been applied to a number of examples ranging from BGs of rather small systems to those of medium size, including mechanical systems with kinematic loops like the well known slider crank mechanism. Due to the lack of space not all BGs of those examples along with explanations can be given here. Nevertheless, statistics of the program Dymola for a number of examples are listed in table 1. The input to the program has been a description of the BG in the language Dymola. By means of a library containing model classes for the BG elements Dymola generates the model equations. These are differentiated by Pantelides' algorithm. After indicating to Dymola which outputs of storage elements are no state variables, and after assigning variables to equations, Dymola can produce an input file for a number of simulation programs. In our experiments we produced ACSL code. If Dymola's tearing capability is activated by the command

```
- set tear on ,
```

the equations are automatically torn. Of course, for convenience all necessary commands and settings of switches can be collected in a command input file.

Table 1 clearly shows two results. If a symbolic reduction of an index-one DAE system is wanted, then after differentiation, tearing of the system of simultaneous equations can significantly reduce the computational amount in terms of arithmetic operations. Even in the case of a small system with two capacitances connected in parallel the number of arithmetic operations is reduced by 40 %.

The second result is that for larger linear systems the number of algebraic operations increases considerably. As has been experienced by Bos [2] and others as well, the symbolic solution of larger linear systems indeed, requires much memory. Consequently, for some systems a symbolic reduction without tearing turns out to be impossible. In that case a possible remedy is to calculate the coefficients of the right hand side of the ODE system numerically. One disadvantage of course, is that the impact of system parameters cannot be directly analyzed any longer.

For modeling of the 2D mechanical systems the basic building block of a free rigid body given by Bos [2] has been used accounting for simplifications due to planar motion. The ring structure listed in table 1 is a simple BG that involves a causal path between two stores of the same type, as well as an algebraic loop between resistors R_1 and R_2 (cf. Fig. 6). In each causal path a residual sink has been added as it is shown in Fig. 7. Of course, different types of dependent storage elements can be treated as well.

6 Conclusions

In this paper we have shown that by adding residual sinks to a BG with dependent stores, information specifying tearing variables and corresponding residual equations can be given to a program like Dymola. As a result, the formulae manipulation capabilities of that program enable it to use that information directly to reduce without any further help an initial DAE system (of index-one) sym-

System	No. of storage ports	No. of simultaneous equations	No. of arithmetic operations without tearing	with tearing	Reduction
two parallel Cs	2	4	20	12	40 %
ring structure (cf. Fig. 7)	2	6 + 6	99	29	70.7 %
C network (cf. Fig. 3)	5	9	166	32	80.7 %
mechanical lever	4	12	233	58	75.1 %
2D pendulum	3	21	1636	109	93.3 %
2D double pendulum	6	53	—	604	
slider crank	8	9 + 61	—	2253	

Table 1: Numbers of arithmetic operations for some examples

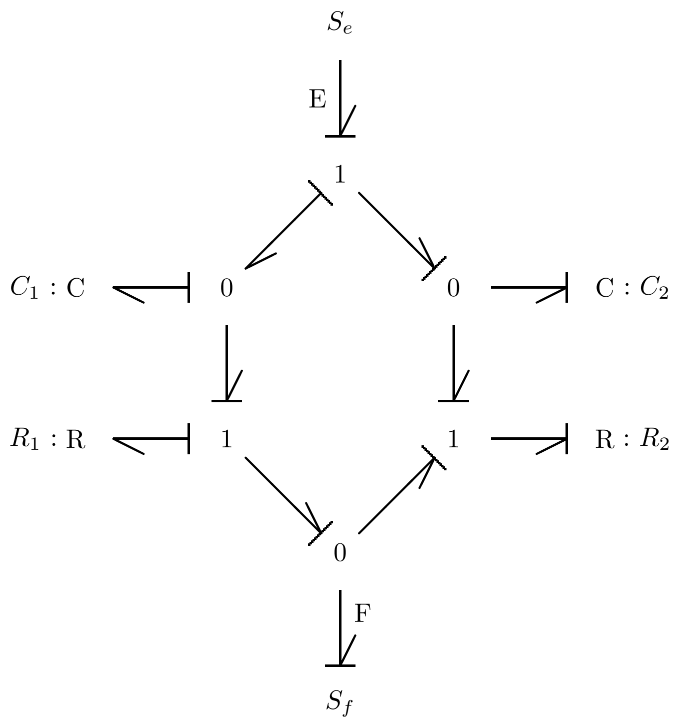

Figure 6: BG with dependent capacitances and an algebraic loop

bolically to an explicit ODE system. That is, the modeler can support symbolic processing of the equations generated from a BG description without having to inspect them, which obviously, becomes impossible for larger sets of equations. The method presented is quite general and has been applied to a number of BGs of electrical and mechanical systems, including those with kinematic loops. Considering the general case, we have only assumed that the equations are linear in the tearing variables.

With our approach smaller non torn subsystems may occur in the reduction phase for instance, if the equations of a coordinate transformation need to be inverted, or in case kinematic loops are present. Since the symbolic solution of larger linear systems needs much memory, it is important to exploit tearing even more comprehensively. One possible way is to exploit information given by the modeler at the BG level and in addition to tear emerging smaller subsystems formally at an embedded level matrix. Whatever additional approaches are used, the major goal should be that once equations have been generated, the modeler should not be concerned with their inspection.

Acknowledgment

This paper is based on research work carried out during the first author's sabbatical leave at the University of Arizona in Tucson, in 1995, who gratefully acknowledges

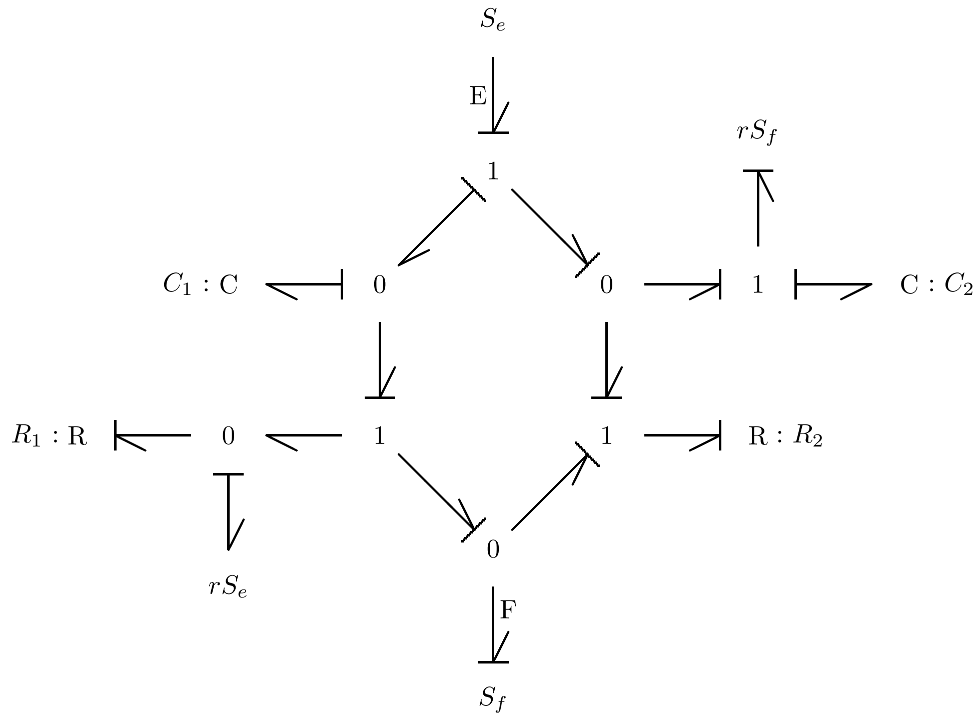

Figure 7: BG with residual sinks added

a grant from the German Research Council "Deutsche Forschungsgemeinschaft" (DFG) in Bonn, Germany.

References

[1] *Borutzky, W.; Cellier, F.:* "Tearing Algebraic Loops in Bond Graphs", submitted for publication

[2] *Bos, A. M.:* "Modelling Multibody Systems in Terms of Multibond Graphs with application to a motorcycle", Ph.D. Thesis, Twente University, Enschede, 1986

[3] *Brenan, K. E.; Campbell, S. L.; Petzold, L. R.:* "Numerical Solution of Initial-Value Problems in Differential Algebraic Equations", North-Holland, New York, 1989

[4] *Cellier, F:* "Hierarchical non-linear bond graphs: a unified methodology for modeling complex physical systems", SIMULATION, April 1992, pp. 230-248

[5] *van Dijk, J.; Breedveld, P. C.* (1991): "Simulation of System Models Containing Zero-order Causal Paths – I. Classification of Zero-order Causal Paths", J. Franklin Institute 328(5/6), 959-980

[6] *Elmqvist, H.:* "Dymola - User's Manual", Dynasim AB, Park Ideon, Lund, Sweden, 1995

[7] *Elmqvist, H.:* Private communication, June 12, 1995

[8] *Elmqvist, H.; Otter, M.:* "Methods For Tearing Systems of Equations in Object-Oriented Modeling", Proc. 1994 ESM, Barcelona, Spain, June 1-3, 1994, pp. 326-332

[9] *Gawthrop, P. J. and L. P. S. Smith* (1992): "Causal augmentation of bond graphs", Journal of the Franklin Institute, 329(2), pp. 291–303

[10] *Karnopp, D. C.; Rosenberg, R. C.:* "Analysis and Simulation of Multiport Systems", M. I. T. Press, Cambridge, MA, U. S. A., 1968

[11] *Kron, G.:* "Diakoptics - The Piecewise Solution of Large-Scale Systems", MacDonald & Co., London 1962

[12] *Mah, R. S. H:* "Chemical Process Structures and Information Flows", Butterworths, 1990

[13] *Nagel, L. W.:* "SPICE2: A computer program to simulate semiconductor circuits", Univ. of Cal., Electronic Research Lab., ERL-M 520, 1975

[14] *Pantelides, C. C.:* "The consistent initialization of differential-algebraic systems", SIAM, J. Scientific and Statistic Computing, September 1988, pp. 213-231