MODELING OF CONDITIONAL INDEX CHANGES

by

Matthias Krebs

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 9 7

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the Department of Electrical and Computer Engineering or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interest of scholarship. In all other instances, however, permission must be obtained by the author.

SIGNED: Matthias Krebs

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

François E. Cellier — 10/21/97

Dr. François E. Cellier
Professor of Electrical and Computer Engineering

Date

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor François E. Cellier, for his great support, guidance, and encouragement throughout this thesis. This degree would have been impossible for me without his support. His willingness to spend so many hours in the lab working on the completion of this project was greatly appreciated. I am truly fortunate to have worked on a project with such an extremely knowledgeable and helpful individual. Furthermore I am deeply grateful for the possibility to hear his courses that extended my knowledge in an extraordinary way.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

An electrical circuit containing switch elements represents a variable structure system. Ideal switch element can be described by a switch equation using a discrete variable to specify the switch position. The causality of an ideal switch element cannot be fixed. However non-ideal switches used to prevent the causality problem often cause artificial stiffness in the resulting differential equation model.

The idea for resolving the causality assignment problem was to modify Pantelides' index reduction algorithm to a form suitable for conditional index changes. However, a fairly simple counterexample shows that pure modifications of switch equations cannot solve the causality problem.

However, the previous analysis resulted in a new idea, the use of *implicit difference formulae* that are widely used in commercial DAE solvers. The new approach solves the problems associated with index changes. Yet, the concept still has remaining problems caused by the ideal nature of switches.

PREFACE

The topic of this thesis is to find an algorithm, probably graph–theoretical, that recognizes conditional index changes in a model and performs the necessary formulae manipulations in extension of the Pantelides algorithm. The first part of this thesis is concerned with graphical methods that were used. The second part uses these graphical methods to find restrictions for the parameter set used to describe the modification problem. These restrictions result in contradictions for two identified possibilities of seemingly promising solutions in a simple example. The analysis proves that there cannot exist any solution to the problem using an extended Pantelides algorithm. The idea for a different approach was then borne, and the subsequent part of the thesis shows how the problem can indeed be solved by using Difference Formulae. Then follows the description of the complete solution for the simple example, as well as a more complicated example containing six switches that is characterized by 64 possible switch combinations. This second example exhibits some cases where the simulation still does not work. In the sequel these cases are examined to unveil the reasons that explain the singularities.

CHAPTER 1

Introduction

1.1 The Problem Statement

Find an algorithm, probably graph–theoretical, that recognizes conditional index changes in a model and performs the necessary formulae manipulation in extension to the Pantelides [5] algorithm. Attacking this problem will start with a simple example that cannot be solved by the usual Pantelides algorithm. Then, one can find the modified equations for the example shown in Fig. 1.1, and derive from these equations underlying basic rules. Working with these basic rules, they can be embodied in an extended algorithm for arbitrarily complex circuits.

1.2 Basic Concepts

This section describes briefly basic concepts that are important to understand the subsequent work. Model descriptions, equation systems, differential algebraic equation systems, and switch equations are the four parts that are explained in detail. The first part on model descriptions elaborates on the two most frequently used model descriptions, the differential algebraic equation system and the ordinary

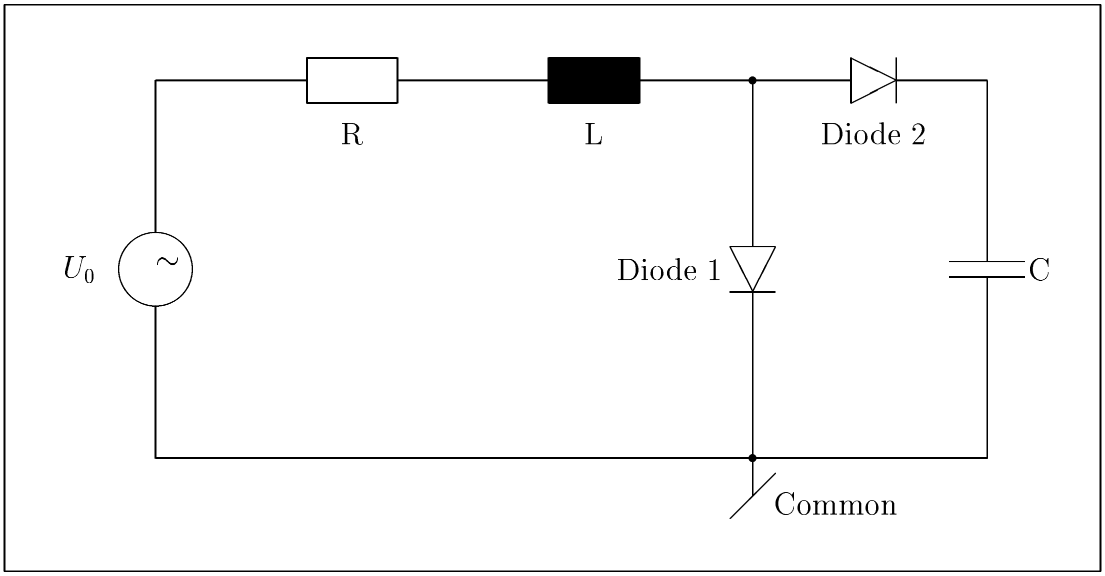


Figure 1.1: Example circuit

differential equation system. Subsequently, general equation system concepts that are necessary to understand algebraic loops are introduced. Differential algebraic equation systems are the focus of the third part that includes the definition of the DAE index, the explanation of the higher index problem, and the description of the Pantelides algorithm. In the final part, switch equations are introduced and explained. Finally, the conditional index changes resulting from switch equations are introduced by means of an example.

1.2.1 Model Descriptions

1.2.1.1 Differential Algebraic Equation Systems (DAE)

In general, modeling of physical systems leads naturally to models described by sets of *differential algebraic equations.* A Differential Algebraic Equation system (DAE) is of the following general form:

$$0 = h(x, \frac{\partial x}{\partial t}, y, p, t) \qquad ; \qquad x(t_0) = x_0 \tag{1.1}$$

where x is the vector of unknown variables that truly appear in differentiated form, whereas y is the vector of purely algebraic unknown variables. p is a vector of parameters contained in the model description, and t denotes time. Note that the number of equations equals the sum of the unknown variables, i.e. $dim(h) = dim(x) + dim(y)$.

If Equation (1.1) can be explicitly solved for y and y is independent of any derivative, we can use the more specialized form:

$$0 = f(x, \frac{\partial x}{\partial t}, p, t) \qquad ; \qquad x(t_0) = x_0 \tag{1.2}$$

$$y = g(x, p, t) \tag{1.3}$$

This form is characterized by a set of implicit purely differential equations and a set of explicit purely algebraic equations. Note that this form is contained in the previous general form, and

$$h = \begin{bmatrix} f(x, \frac{\partial x}{\partial t}, p, t) \\ \\ y - g(x, p, t) \end{bmatrix} \tag{1.4}$$

1.2.1.2 Ordinary Differential Equation Systems (ODE)

Another frequently used model description is the state–space model, represented by a set of *ordinary differential equations*, and a set of algebraic output equations. A state–space model is described by an Ordinary Differential Equation System (ODE):

$$\dot{x} = f(x, p, t) \qquad ; \qquad x(t_0) = x_0 \tag{1.5}$$

supplemented by the set of algebraic output equations:

$$y = g(x, p, t) \tag{1.6}$$

where x is the vector of state variables, y is the vector of output variables, p is a vector of parameters contained in the model description, and t denotes time. Note that both parts, the ODE system and the algebraic output system, are contained in the state–space model. This frequently used model description is even more specific than the specialized DAE description. The state–space description assumes that the state variables are known, and calculates the derivatives using the set of assignments given by f. The knowledge of the state variables makes the solution of the algebraic output system trivial, as this results in a mere function evaluation. Thus, the output equation system is frequently neglected.

The number of equations in the ODE system equals the sum of unknown variables, however this time, the derivatives of the state variables and not the state variables themselves are considered to be unknown.

1.2.2 Equation System Concepts

1.2.2.1 Structure Incidence Matrix

The structure incidence matrix is used to describe the properties of an equation system. The following example equation system:

$$f_1(x_1, x_2) = 0 \tag{1.7}$$

$$f_2(x_2) = 0 \tag{1.8}$$

$$f_3(x_1, x_2, x_3) = 0 \tag{1.9}$$

can be characterized by the structure incidence matrix:

$$S = \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{ccc} x_1 & x_2 & x_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \end{array} \tag{1.10}$$

Thereby an element in the i^{th} row and k^{th} column is either one, if the k^{th} variable forms part of the i^{th} equation, or zero, if the k^{th} variable does not show up in the i^{th} equation.

1.2.2.2 Algebraic Loop

The structure of an equation system is preserved if the order of the equations or the order of the variables is changed. Multiplying the structure incidence matrix S with a permutation matrix P from the left corresponds to rearranging the equation sequence, whereas multiplying the structure incidence matrix S with a permutation matrix Q from the right corresponds to rearranging the variable sequence.

$$\hat{S} = P \cdot S \cdot Q \tag{1.11}$$

The equivalent structure incidence matrix $\hat{S}$ has the same properties as the original structure incidence matrix S. The permutation matrices, P and Q, are determined in such a way that they transform the matrix S into a lower block–triangular matrix $\hat{S}$. This lower block–triangular form represents the easiest way to solve the equation system. For the above example

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{and} \qquad Q = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

result in the lower triangular matrix

$$\hat{S} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \tag{1.12}$$

In this matrix $\hat{S}$ the diagonal entries are blocks of size one. However, if the entries in the matrix $\hat{S}$ require diagonal block sizes greater than one, then the equation system contains one or more *algebraic loops*. The number of algebraic loops is equivalent to the number of diagonal blocks with a dimension greater than one.

For example:

$$f_1(x_1, x_2, x_3) = 0 \tag{1.13}$$

$$f_2(x_1, x_2, x_3) = 0 \tag{1.14}$$

$$f_3(x_1, x_2, x_3) = 0 \tag{1.15}$$

is a completely coupled algebraic system that can be characterized by the structure incidence matrix:

$$S = \begin{array}{c c} & \begin{array}{ccc} x_1 & x_2 & x_3 \end{array} \\ \begin{array}{c} f_1 \\ f_2 \\ f_3 \end{array} & \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \end{array} \tag{1.16}$$

This structure incidence matrix is already in a lower block triangular form, and permutation matrices cannot change the form of the structure incidence matrix. In

Table 1.1: Index of a DAE System

DAE index	incidence matrix S	linear	nonlinear
0	lower triangular	can be converted into ODE form	solvable by successive Newton Iterations for single variables
1	block lower triangular	contains algebraic loops	solvable by successive Newton Iterations for several (indicated by diagonal block size) variables together
2 or higher	is singular	contains one or more depending storage elements	contains one or more depending storage elements

this example, we have the extreme case of a single block that can be considered as a diagonal block of dimension three. Consequently, the equation system has one algebraic loop containing all three equations and all three variables.

1.2.3 Differential Algebraic Equation System Concepts

1.2.3.1 DAE Index

The index of a DAE system is a measure of the solvability of a DAE system description by certain DAE solvers, and describes the difficulties involved in solving the DAE system [11]. The index of a DAE system is described concisely in the Table 1.1.

In the linear case, an index 0 DAE system can be converted into ODE form, whereas in the nonlinear case, successive Newton Iterations for one variable result in values for the derivatives that are then the input of an ODE solving integration algorithm. The index 1 DAE system is in the linear case nothing more than a matrix equation that can be solved in many ways, e.g. using Cramer's rule for the inversion of the matrix. A nonlinear index 1 DAE can be solved by successive Newton Iterations over several simultaneous variables. Thus, the main difference to the index 0 DAE system is the need for Newton Iterations over a vector of simultaneous variables, instead of successive Newton Iterations over a single variable. A DAE system with an index of two or higher is called a *higher index problem*, and is described by an example in the next section.

1.2.3.2 Higher Index Problem

Fig. 1.2 represents a simple example of a higher index problem. The two capacitors in parallel are two dependent storage elements. The voltage across the two capacitors is always the same, and thus the amount of energy stored in the electric field is characterized by a single variable. In such a simple example, one would probably replace the two capacitors with one resulting capacitance, however in more complex circuits, detecting dependent storage elements is a difficult and error–prone task. Let us take a look at the DAE description to get a feeling for the peculiarities that occur:

$$U_0 - \hat{U}_0 \cdot cos(\omega t) = f_1 = 0$$

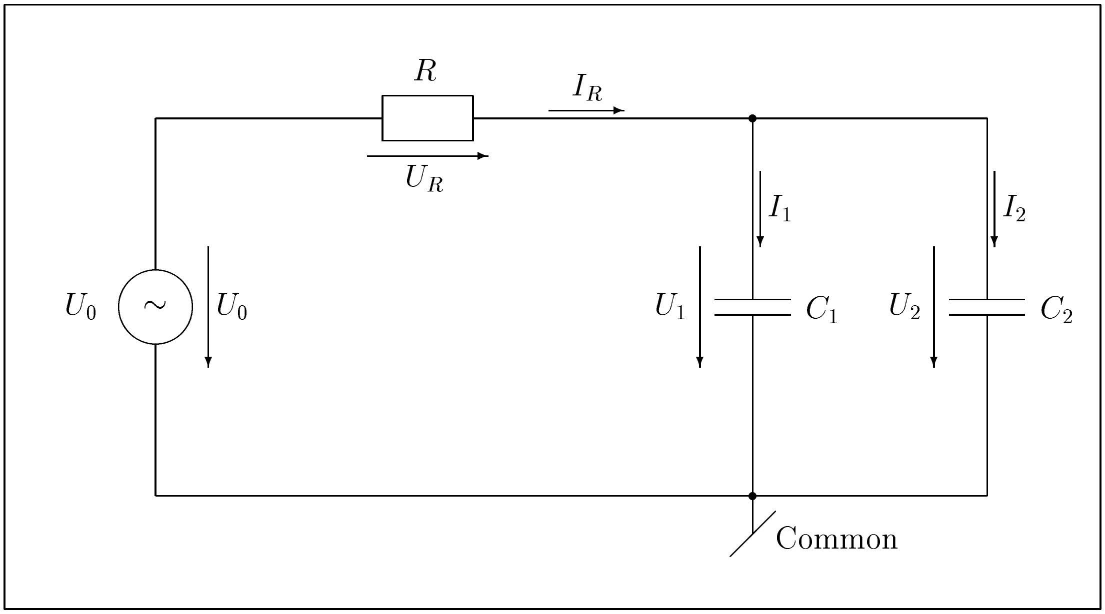


Figure 1.2: Higher Index Problem

$$U_R - R \cdot I_R = f_2 = 0$$

$$I_1 - C_1 \cdot \frac{\partial U_1}{\partial t} = f_3 = 0$$

$$I_2 - C_2 \cdot \frac{\partial U_2}{\partial t} = f_4 = 0$$

$$U_0 - U_R - U_1 = f_5 = 0$$

$$U_1 - U_2 = f_6 = 0$$

$$I_R - I_1 - I_2 = f_7 = 0$$

Let us try to transform the set of equations to a set of explicit equations for an ODE solver. In this case U_1 and U_2 are assumed known, and the structure incidence matrix is thus:

$$S = \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} U_0 & U_R & I_R & I_1 & I_2 & \frac{\partial U_1}{\partial t} & \frac{\partial U_2}{\partial t} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \end{array} \tag{1.17}$$

Equation f_6 does not contain any unknowns, and thus the structure incidence matrix S contains a zero row. This clearly indicates a singular structure incidence matrix and points to the higher index problem. The structure incidence matrix indicates that we have seven equations for seven unknowns, yet equation f_6 is completely useless. Let us take a closer look at equation f_6:

$$U_1 - U_2 = f_6 = 0 \tag{1.18}$$

f_6 is a function of time, and is equal to zero for all times, and therefore, also the derivative $\frac{\partial f_6}{\partial t}$ must be equal to zero for all times.

$$\frac{\partial U_1}{\partial t} - \frac{\partial U_2}{\partial t} = \frac{\partial f_6}{\partial t} = \hat{f}_6 = 0 \tag{1.19}$$

If we substitute this modified equation $\hat{f}_6$ for the previously used equation f_6, we obtain the structure incidence matrix:

$$\hat{S} = \begin{array}{c c} & \begin{array}{ccccccc} U_0 & U_R & I_R & I_1 & I_2 & \frac{\partial U_1}{\partial t} & \frac{\partial U_2}{\partial t} \end{array} \\ \begin{array}{c} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ \hat{f}_6 \\ f_7 \end{array} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \end{array} \tag{1.20}$$

The structure incidence matrix $\hat{S}$ is non–singular, and the problem can now be converted into ODE form.

In ODE simulation, the outputs of integrators are chosen as “state variables”. This notation is already used in the DAE description that should be converted into ODE form. Whenever state variables appear in an algebraic equation, the model contains dependent storage elements that result in a higher index problem. The *Pantelides Algorithm* is used to determine, which equations need to be differentiated in order to reduce the DAE index before the DAE system can be converted to an ODE description.

1.2.3.3 The Pantelides Algorithm

The Pantelides Algorithm determines the equations that need to be differentiated in order to remove algebraic couplings between "state variables." The algorithm does the index reduction in steps. Each step reduces the index by one, and if after a step there are still further algebraic couplings between state variables, another step is necessary. However, the Pantelides algorithm does not work if the DAE description itself can change the index depending on a discrete variable. Such model descriptions are called *Conditional Index Models*, and this thesis is particularly concerned with Conditional Index Changes caused by electrical switch elements.

1.3 Switch Elements

An electric switch element is a two–pin element, just like all of the traditional linear passive circuit elements. Yet whereas resistors, capacitors, inductors, voltage sources, and current sources are all modeled using a single equation, the switch element is usually described by two separate equations, one for each of the two possible switch positions. However it is possible to combine these two equations in a single conditional statement, and consequently, the switch element can be represented by the equation

$$0 = \quad \textbf{if} \quad OpenSwitch \quad \textbf{then} \quad I \quad \textbf{else} \quad U \tag{1.21}$$

OpenSwitch in this statement is a *boolean variable* with the two possible values *true* and *false*. These values correspond to the opened and closed position, respectively. For the purpose of an equation solver, this equation can be rewritten in a more useful form:

$$0 = OpenSwitch \cdot I + (1 - OpenSwitch) \cdot U \tag{1.22}$$

Here, OpenSwitch is a *discrete variable* with two possible values 0 and 1. The value 1 corresponds to the opened switch, and the value 0 represents the closed switch. Thus, an electric switch element can be characterized by a single equation, containing the voltage u and the current i, just like all other linear passive circuit elements. However, and contrary to the equations governing other circuit elements, the equation contains an additional discrete switch variable that describes the position of the switch.

The equation is only of use if we can solve practical problems with it. Let us look at an introductory example, as shown in Fig.1.3. In this example circuit, the switch element is placed in series with an inductor. The current through the inductor is a natural state variable, thus the current I is known, and the switch equation must be solved for U.

$$U = \frac{OpenSwitch}{(OpenSwitch - 1)} \cdot I \tag{1.23}$$

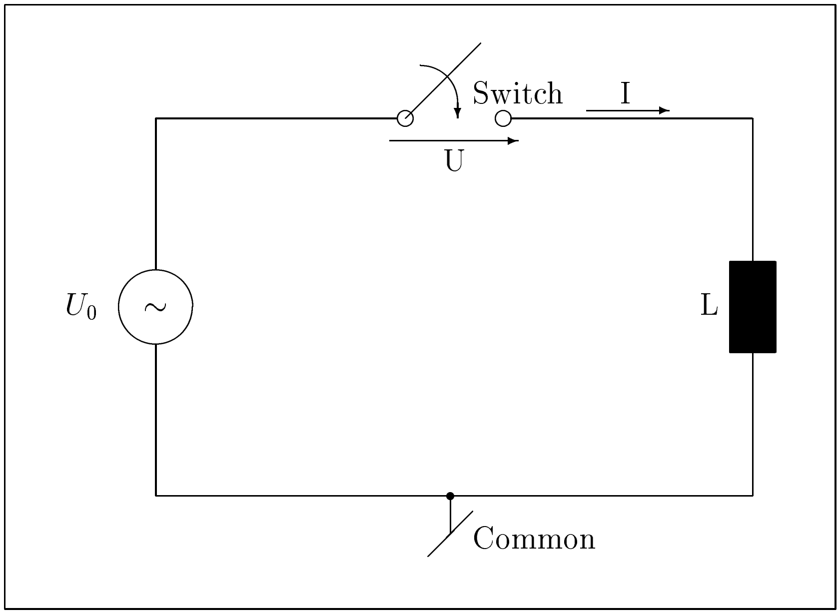


Figure 1.3: Switch–Inductor example circuit

Unfortunately, equation (1.23) is only valid as long as the switch is closed. As soon as the switch opens, the expression in the denominator becomes zero, and the simulation ends with a division by zero. Thus, one may ask oneself how useful the switch equation (1.22) is. In this example, the result is understandable as the circuit behaves in a different manner in reality than in the model description. The current through an inductor cannot jump, and as a result, a light arc will be drawn. This light arc represents a growing resistance until the arc breaks with a resulting infinite resistance value. This example illustrates a property that any switch element used in a simulation model must adhere to. It can be concisely stated as follows:

> *The causality of a switch element must not be dictated by the surrounding circuit, but must be merely a function of the independent discrete variable* OpenSwitch.

The meaning of the causality principle will be further explained in the second chapter.

This characteristic property of an independent switch element leads to the conclusion that switch elements must always be contained in algebraic loops, and thus, the discrete switch variable can assume either value regardless of the switch environment. [2]

At this point it was not known, how this concept could be extended to more than one switch element. However, it was clear that at least one algebraic loop is needed, in which the switch equations are contained, to prevent the switch equations from being solved for either variable. Rather, the whole system of equations constituting the algebraic loop will be solved together.

The next example circuit, shown in Fig.1.4, satisfies this requirement. This example contains a diode, a specific switch element. The relationship between a switch and a diode can be modeled in the object–oriented modeling language Dymola [6] as follows:

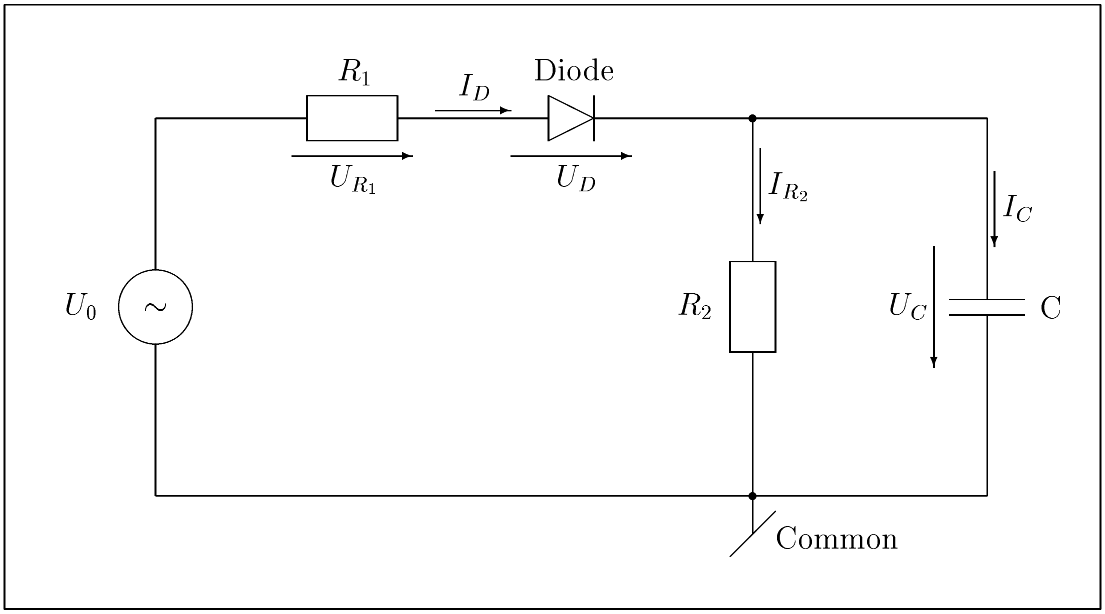


Figure 1.4: Half–wave rectifier circuit

```
model class TwoPin

 cut WireA(Va/I), WireB(Vb/-I)

 main cut Wires [WireA,WireB]

 main path P <WireA-WireB>

 local U

   U = Va - Vb

end
```

```
model class (TwoPin) Switch
 terminal OpenSwitch
   0 = OpenSwitch * I + (1 - OpenSwitch) * U
end

model class (Switch) Diode
 new(OpenSwitch) = if ((not U>0) and (not I>0)) then 1 else 0
end
```

The first part of the code is the basic declaration of a *TwoPin* element, derived from this class is a new class *Switch*, which in turn is the superclass of another derived specific switch class *Diode*. In the same fashion, classes for voltage sources, current sources, capacitors, inductors, and resistors can be derived from the *TwoPin* class by inheritance. Similarly, classes for thyristors, samplers, and other specific switches can inherit the properties of the *Switch* class, which itself inherits properties from the *TwoPin* class.

This definition of an ideal diode is characterized by the diagram shown in Fig.1.5.

The model for the circuit shown in Fig.1.4 contains the following equations:

$$-U_0 + U_C + U_D + U_{R_1} = 0 \tag{1.24}$$

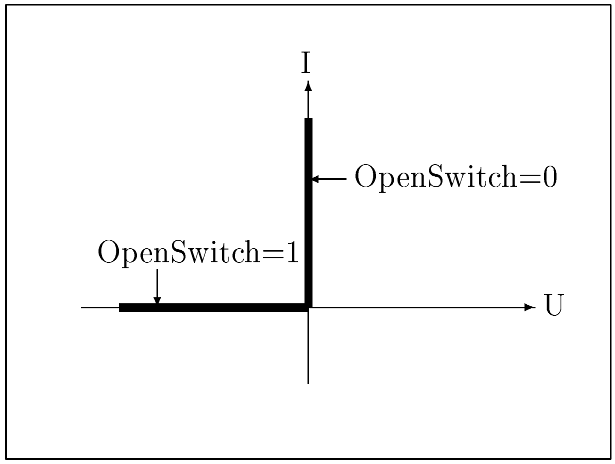


Figure 1.5: Ideal Diode Characteristics

$$I_D - I_C - I_{R_2} = 0 \tag{1.25}$$

$$U_{R_1} - R_1 \cdot I_D = 0 \tag{1.26}$$

$$U_C - R_2 \cdot I_{R_2} = 0 \tag{1.27}$$

$$I_C - C \cdot \frac{\partial U_C}{\partial t} = 0 \tag{1.28}$$

$$OpenSwitch \cdot I_D + (1 - OpenSwitch) \cdot U_D = 0 \tag{1.29}$$

accompanied by the equations:

$$New(OpenSwitch) - \text{If not}(U_D > 0) \text{ and not}(I_D > 0) \text{ then } 1 \text{ else } 0 = 0 \tag{1.30}$$

$$U_0 = \hat{U}_0 \cdot sin(\omega t) \tag{1.31}$$

1.24 - 1.29 describe a system of six equations for the six unknowns U_D, $\frac{\partial U_C}{\partial t}$, U_{R_1}, I_D, I_C, and I_{R_2} with the parameters R_1, R_2, and C. The voltage U_C across the capacitor is a state variable. Its value is known from a simulation step, whereas the derivative $\frac{\partial U_C}{\partial t}$ is one of the unknowns. Equation 1.30 contains the information on how to determine the new value of the discrete switch variable *OpenSwitch* after a simulation step, whereas 1.31 determines how to calculate a new value for the input voltage using additional parameters $\hat{U}_0$, and ω. The Differential Algebraic Equation (DAE) system of six unknowns can be rewritten in matrix form:

$$\underbrace{\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 1 & -R_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & R_2 & 0 \\ 0 & -C & 0 & 0 & 0 & 1 \\ (1-OS) & 0 & 0 & OS & 0 & 0 \end{pmatrix}}_{A_1} \cdot \begin{pmatrix} U_D \\ \frac{\partial U_C}{\partial t} \\ U_{R_1} \\ I_D \\ I_{R_2} \\ I_C \end{pmatrix} = \begin{pmatrix} U_0 - U_C \\ 0 \\ 0 \\ U_C \\ 0 \\ 0 \end{pmatrix} \tag{1.32}$$

where OS has been introduced as an abbreviation for the hitherto used name *OpenSwitch* for the benefit of a more compact notation.

$$\det A_1 = \begin{vmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 1 & -R_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & R_2 & 0 \\ 0 & -C & 0 & 0 & 0 & 1 \\ (1-OS) & 0 & 0 & OS & 0 & 0 \end{vmatrix}$$

$$= (1-OS)\cdot\underbrace{CR_1R_2}_{k_1} - OS\cdot\underbrace{CR_2}_{k_2} \tag{1.33}$$

Equation 1.33 verifies that the DAE system is indeed non–singular for both OS = OpenSwitch = 0 and OS = OpenSwitch = 1, because the determinant assumes the values k_1 and k_2, respectively, in the two cases. Thus, we have an algebraic loop that is always solvable, and the simulation of this sample circuit will work without any problems.

However, in this example we missed one important aspect: the *conditional index change*. A conditional index change is characterized by a change in the index of the DAE system. If one would describe the example circuit 1.4 by two separate models, one for each of the two possible switch positions, the indices of the two DAE systems would be 0 in both cases.

Let us look at a different example in which the aspect of the conditional index change comes to bear. Fig.1.6 contains a diode to assure the proper modeling of the circuit at switch instants. Contrary to Fig.1.3, the switch in this model opens only if the current passes through zero. Thus, the characteristic of the ideal diode prevents the light arc. However, just as in the earlier example, the current through the inductor is a natural state variable, and thus, the causality of the switch is predetermined by the inductor. In reality, if the switch opens, there is no longer any inductor present, because the inductor is no longer contained in any mesh. Yet, the model contains a first order differential equation for the inductor, even if the inductor no longer plays any part in determining the behavior of the circuit. Therefore, in this example, the index jumps from 0 to 2 when the discrete switch variable changes its value from 0 to 1.

For the model of the circuit shown in Fig.1.6, the necessary algebraic loop for the switch element can be achieved quite easily by modifying the switch equation to

$$0 = OpenSwitch \cdot \frac{\partial I}{\partial t} + (1 - OpenSwitch) \cdot U \tag{1.34}$$

This equation uses the knowledge that, if a variable is zero for all times, the higher derivatives of that variable must also be zero for all times. If all lower derivatives

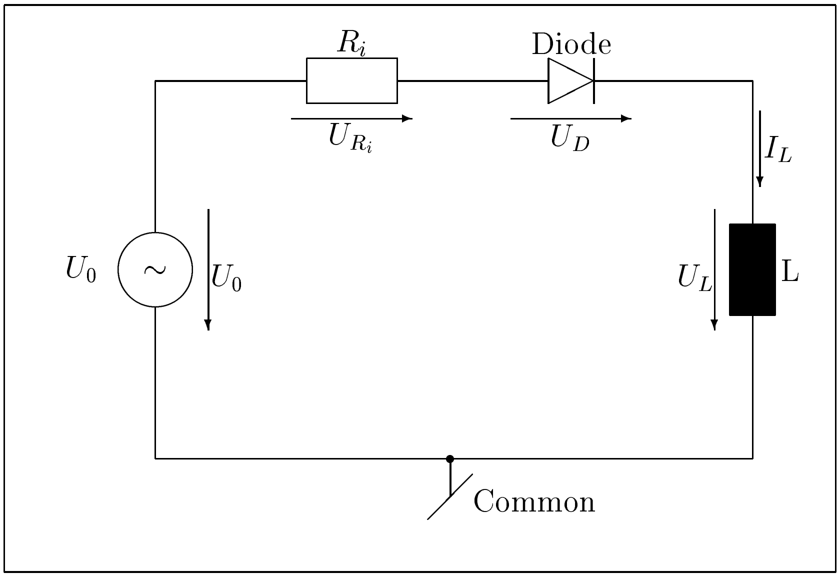


Figure 1.6: Inductive load circuit

of the variable are properly initialized to zero, the modified equation expresses the same condition as the original switch equation. Another view of this modification is that the differentiation introduces auxiliary state variables in the switch element that equalize the DAE Index of the modeled system for all switch positions.

The model for the circuit shown in Fig.1.6 contains the following equations:

$$U_0 - U_{R_i} - U_D - U_L = 0 \tag{1.35}$$

$$U_{R_i} - R_i \cdot I_L = 0 \tag{1.36}$$

$$OpenSwitch \cdot \frac{\partial i_L}{\partial t} + (1 - OpenSwitch) \cdot U_D = 0 \tag{1.37}$$

$$U_L - L \cdot \frac{\partial i_L}{\partial t} = 0 \tag{1.38}$$

accompanied by the equations:

$$New(OpenSwitch) - \text{If not}(U_D > 0) \text{ and not}(I_L > 0) \text{ then } 1 \text{ else } 0 = 0 \tag{1.39}$$

$$U_0 = \hat{U}_0 \cdot sin(\omega t) \tag{1.40}$$

The identity $i_{R_i} = i_D = I_L$ was used to substitute i_{R_i}, and i_D. The Differential Algebraic Equation (DAE) system of four unknowns can be rewritten in matrix form:

$$\underbrace{\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & (1-OS) & 0 & OS \\ 0 & 0 & 1 & -L \end{pmatrix}}_{A_2} \cdot \begin{pmatrix} U_{R_i} \\ U_D \\ U_L \\ \frac{\partial i_L}{\partial t} \end{pmatrix} = \begin{pmatrix} U_0 \\ R_i \cdot i_L \\ 0 \\ 0 \end{pmatrix} \tag{1.41}$$

where OS is again the abbreviation for the previously used name *OpenSwitch* for the benefit of a more compact notation.

$$\det A_2 = \begin{vmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & (1-OS) & 0 & OS \\ 0 & 0 & 1 & -L \end{vmatrix} = (1-OS) \cdot \underbrace{-L}_{k_3} + OS \cdot \underbrace{(+1)}_{k_4} \tag{1.42}$$

Equation 1.42 verifies that the conditional index system with the modified switch equation is indeed non–singular for both $OS = \text{OpenSwitch} = 0$ and $OS = \text{OpenSwitch} =$

1, because the determinant assumes the values k_3 and k_4, respectively, in the two cases. Thus the modification of the switch equation created an algebraic loop that is always solvable, and the simulation of this circuit will proceed correctly without any problems.

In this simple circuit, the necessary modification was easy to find, and in the dual case of a capacitor in parallel with a diode, the modification can be found just as easily. That is, the voltage term **U** is replaced by its first–order derivative and the current term **I** stays the same as in the original switch equation.

This leads to the basic task to be performed:

Replace the general switch equation (1.22) by the modified switch equation (1.43), and determine especially the constants n_1 *and* n_2 *specifying the number of differentiations needed for each of the branches of the* if *statement. Add the necessary equations for the proper initialization.*

$$0 = OpenSwitch \cdot \frac{\partial^{n_1} I}{\partial t^{n_1}} + (1 - OpenSwitch) \cdot \frac{\partial^{n_2} U}{\partial t^{n_2}} \tag{1.43}$$

The modified switch equation is accompanied by $n_1 + n_2$ initial conditions to assure the same behavior as the original switch equation. In Dymola, the initialization process at switch time points is expressed in the following code that is only executed when the expression in the **when** statement *becomes* true.

```
 when OpenSwitch then
  init(I)
  init(Id)
  init(Id2)
   ...
  init(Id(n1-1))
endwhen

when (1 - OpenSwitch) then
  init(U)
  init(Ud)
  init(Ud2)
   ...
  init(Ud(n2-1))
```

endwhen

To find an algorithm to perform such modifications for each switch element in an arbitrarily complex circuit in a deterministic way was the task that should be tackled in this thesis.

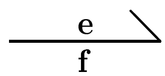
Note that we want to use ideal switch elements. Our conditional index problem could be solved by non–ideal switch elements. In *non–ideal* switch elements, the switch is replaced by a small but non–vanishing resistance in the closed case and by a small yet non–zero conductance in the open case. Using such switch elements allows to simulate conditional index systems. However, the simulation with non–ideal switch elements has a significant disadvantage. Whenever the ideal switch equation would result in a zero denominator, the non–ideal switch equation will have a very small denominator, leading to artificial stiff system behavior. This behavior causes increasing simulation times and thus increased simulation cost.

CHAPTER 2

Graphical Tools and Representations Used Throughout This Thesis

In this chapter, the graphical methods that are being used in later chapters are described and explained by means of simple examples. The meanings of important concepts related to this thesis are briefly described. For further details, the reader is referred to the quoted references.

The chapter consists of two parts, bond graphs and dependence graphs. The first part introduces bond graphs and continues with the bond graph causality concept. The bond graph methodology offers an excellent tool for visualizing the problems associated with conditional index changes. The second part is concerned with dependence graphs and introduces several modifications that are useful in the subsequent work. The modified dependence graphs are used to find requirements for algebraic loops. The algebraic loops should contain the switch equations and thus prevent the problem of singular denominators.



e . . . effort, harpoon side, across variable

f . . . flow, opposite harpoon side, through variable

Figure 2.1: A bond

2.1 Graphical Modeling Tools

2.1.1 Bond Graphs

2.1.1.1 Bond Graph Modeling

This is a short introduction to the technique of Bond Graph Modeling. For a complete understanding, the corresponding literature should be reviewed. A more detailed description of the technique can be found in [1].

A bond, represented by a harpoon, is a graphical way of representing equations. Two variables are associated with each bond, an *across variable*, in bond graph terminology usually referred to as the *effort* **e**, and a *through variable*, called the *flow* **f**. A bond is shown in Fig.2.1.

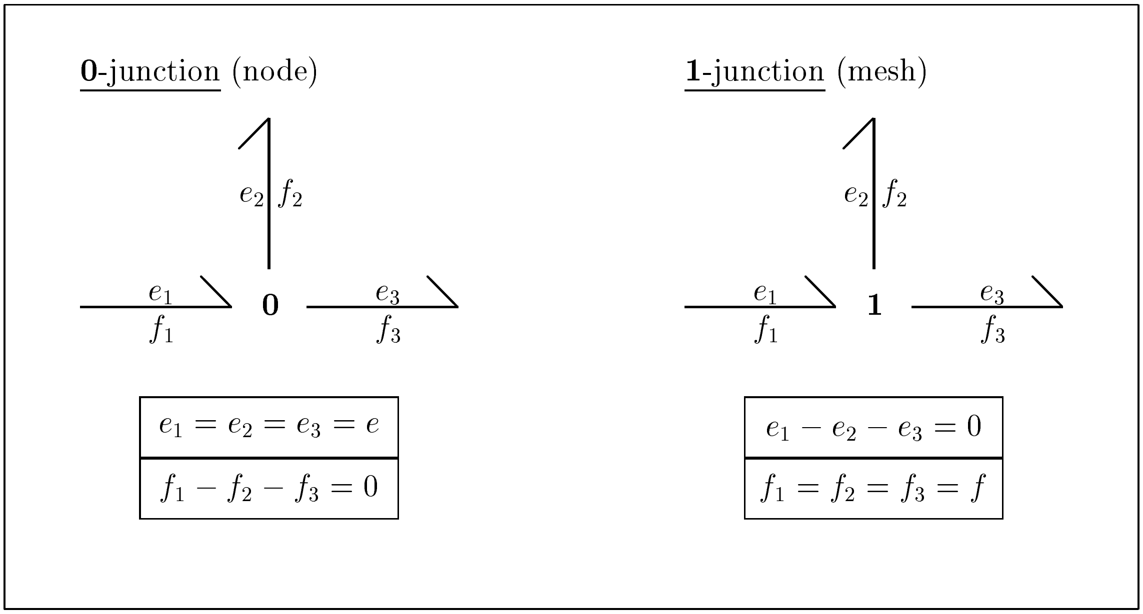


Figure 2.2: Types of Junctions

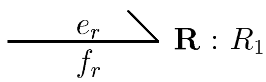
A bond graph, contrary to many other graphical representations, does not separate the two types of variables from each other. Hence a bond graph preserves the *topological structure* of the model [1]. A further advantage of bond graphs is their ability of being used for different application domains, such as electric circuits, translational kinetics, rotational kinetics, hydraulic systems, chemical kinetics, and thermodynamics [1]. Bonds connect either to model elements or to other bonds in a junction. There are two different junction types, shown in Fig.2.2.

For linear circuit theory, the 0–junction represents a node, and the 1–junction represents a mesh. In a 0–junction, all effort variables are set equal whereas all flow variables add up to zero, corresponding to Kirchhoff's current law. In a 1–junction, all effort variables add up to zero whereas all flow variables are set equal, reflecting Kirchhoff's voltage law. At least three bonds are needed to form a true junction, since two–bond junctions can be eliminated by amalgamating the two bonds into one. This follows from the fact, that in the case of only two bonds, the junction equations result in two identities. Neighboring junctions of the same gender can be combined into a single junction. Hence a bond connects either two junctions of different gender or a junction with a model element.

The two–pin elements of the previous chapter are, in bond graph terminology, called oneport elements. The bondgraphic oneport elements are shown in Fig.2.3.

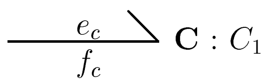
Of course, in the case of electrical circuits, the effort variable corresponds to the voltage across the two–pin element whereas the flow variable maps into the current flowing through the two pins. The switch element is a general switch element that can be modified to become a *special switch*, e.g. an ideal diode, by specializing the functionality that defines the discrete terminal variable $OS1$.

- Resistance **R** with value R_1



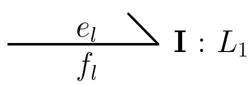
$$\boxed{e_r = R_1 \cdot f_r}$$

- Capacitance **C** with value C_1



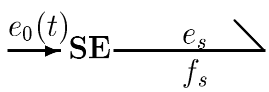
$$\boxed{f_c = C_1 \cdot \frac{de_c}{dt}}$$

- Inductance **I** with value L_1



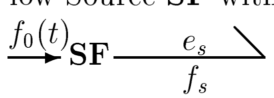
$$\boxed{e_l = L_1 \cdot \frac{df_l}{dt}}$$

- Effort Source **SE** with value $e_0(t)$



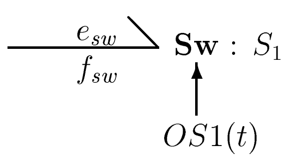
$$\boxed{e_s = e_0(t)}$$

- Flow Source **SF** with value $f_0(t)$



$$\boxed{f_s = f_0(t)}$$

- Switch **Sw** with discrete terminal variable $OS1(t)$



$$\boxed{e_{sw} \cdot (1 - OS1(t)) + f_{sw} \cdot OS1(t) = 0}$$

Figure 2.3: Bond Elements for Circuit Theory

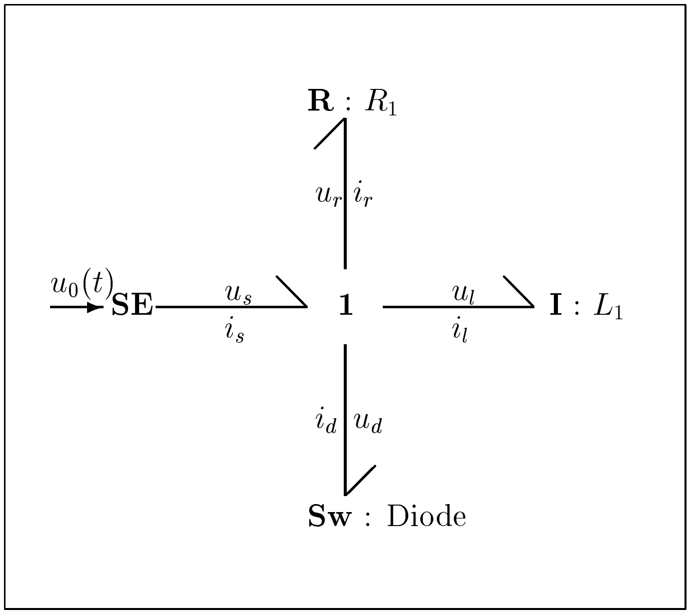


Figure 2.4: Bond Graph Inductive Load

The bond graph representation for the last example of the introduction, 1.6, is shown in Fig.2.4.

2.1.1.2 Bond Graph Causality

The computational structure behind a bond graph can be easily represented using *causality strokes*. Each bond is involved in two equations, one to determine its effort variable **e**, the other to determine its flow variable **f**. The causality can be indicated by a short stroke perpendicular to the bond. The stroke is placed at one side of the

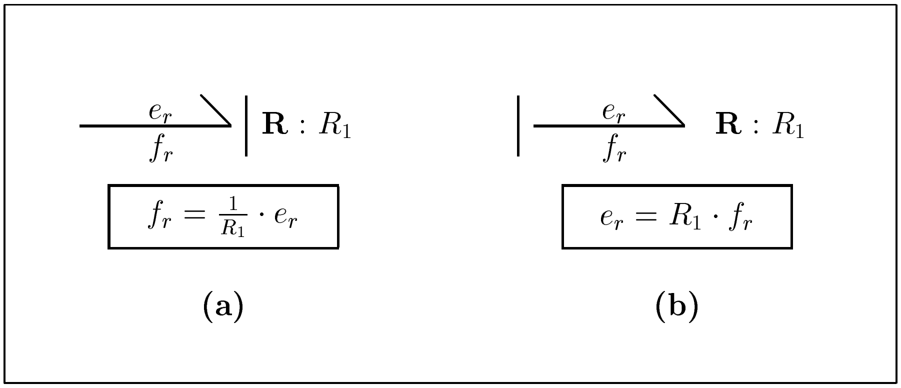


Figure 2.5: Resistor Causalities

bond. There it marks the side of the bond, at which the flow variable is determined [1].

For a resistor **R**, both causalities are meaningful since the element equation $e_r = R \cdot f_r$ can be solved for either the effort variable e_r or the flow variable f_r. In Fig.2.5(a), the flow variable is determined at the resistor element, and the resistor equation is solved for f_r. In contrast, in Fig.2.5(b), the effort variable is determined at the resistor, and the equation is solved for e_r. The second variable, e_r in Fig.2.5(a) and f_r in Fig.2.5(b), is determined at the node to which the element is connected.

However for both types of source elements, the capacitor element, and the inductor element the causality is fixed. In the case of sources, the causality is physically determined through the source type. For the capacitor and inductor, the causality

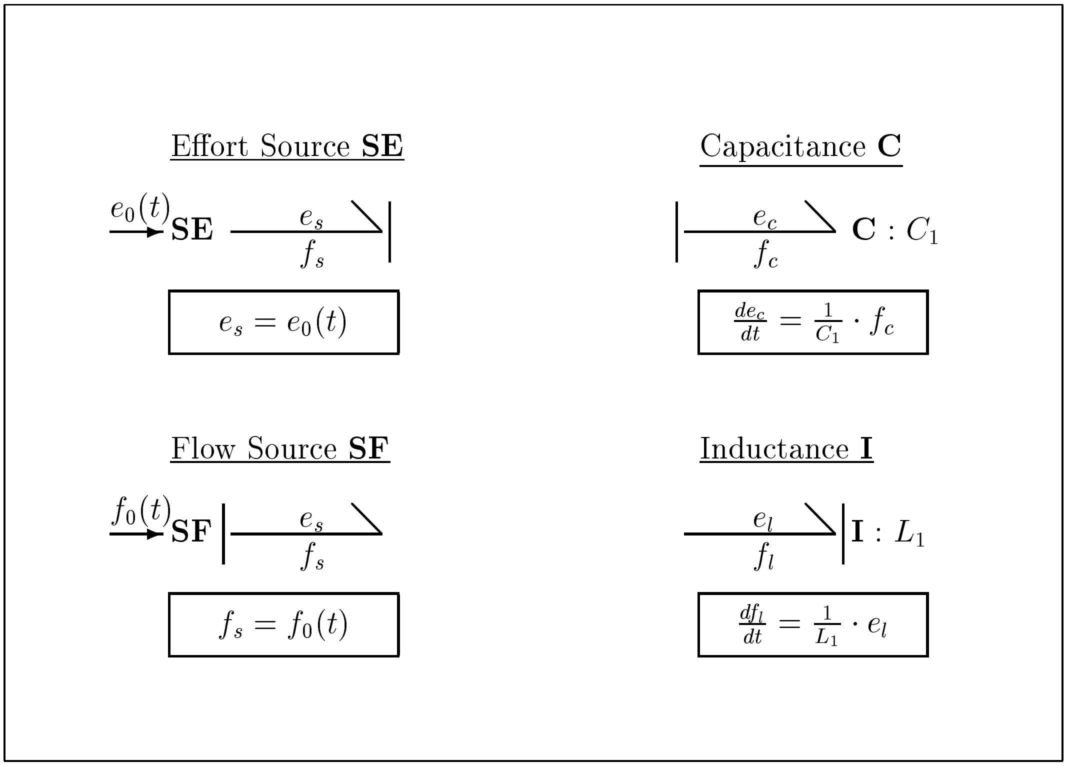


Figure 2.6: Required Element Causalities

is given by a computational requirement. In these two elements, one of the variables is a state variable, and to determine the value of a state variable in a simulation, the derivative of that state variable needs to be calculated. The mandated causality strokes for these element types are shown in Fig.2.6.

Also junctions have requirements since only one flow variable can be determined at any 0–junction whereas only one effort variable can be determined at any 1–junction. Thus at a 0–junction, only one causality stroke can be present, whereas at

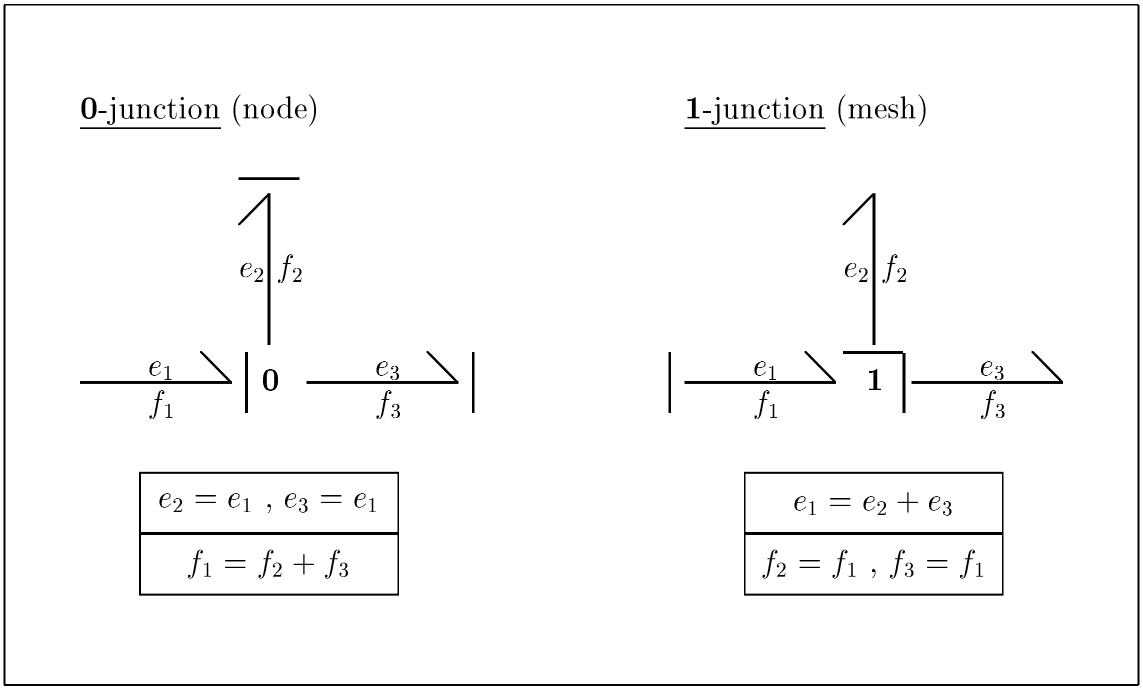


Figure 2.7: Required Junction Causalities

a 1-junction, only one missing stroke is allowed. These requirements are shown in Fig.2.7.

The process of assigning causality strokes results in the conclusions shown in Table 2.1.

As already mentioned in the introduction, an algebraic loop is necessary for an independent switch element. It must be possible to open and close the switch independently from the circuit in which it is embedded. Therefore, a switch element

Table 2.1: Conclusions Causality Strokes Assignment

causality requirements	system is called	causes and implications
can be satisfied	causal	computational structure uniquely determined
cannot be satisfied	non-causal	if not satisfied at a source (e.g. two parallel voltage sources with different voltages)
	degenerate	if not satisfied at an I or C element, structural singularity, higher index DAE
are insufficient	having an algebraic loop	there is a free choice in the computational structure

cannot have a fixed requirement for its causality stroke. Its causality must be determined by the process of opening or closing the switch, as reflected in the value of the discrete variable *OpenSwitch*. Whenever a switch is forced to assume a fixed causality, this will invariably result in a crash of the simulation as soon as the *OpenSwitch* variable changes its value. This led to the conclusion that a switch element can only operate properly if contained in an algebraic loop.

Now we have another interpretation, from the point of view of the bond graph causality, of what goes wrong with the inductive load circuit shown in Fig.1.6. In this example circuit, the diode is used as a specialized switch element. To satisfy the causality requirement of a 1–junction, the causality of the switch element is predetermined, and therefore, the switch can only be simulated in the externally

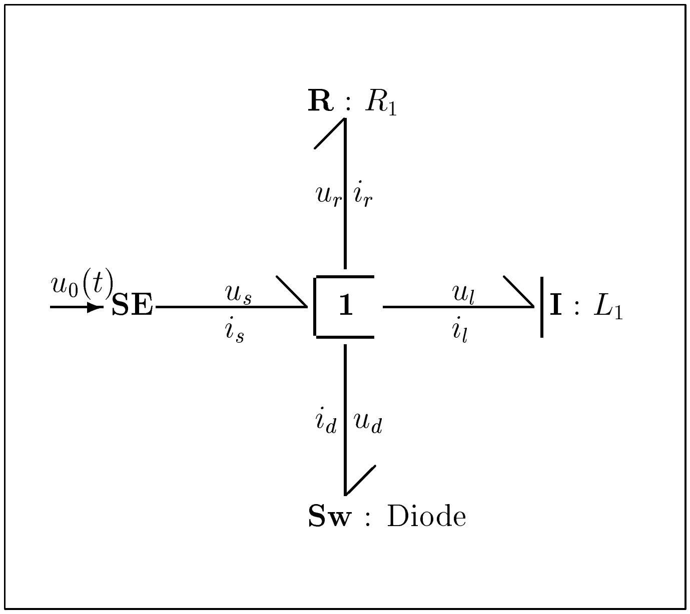


Figure 2.8: Bond Graph with Causality

enforced position. The bond graph containing the causality strokes for the inductive load circuit is shown in Fig.2.8.

2.2 Graphical Methods to Represent Algebraic Structures

So far we have seen that we need an algebraic structure to include a properly working switch element. In this subsection, several possibilities for visualizing algebraic structures are described. The *modified dependence graphs* are extensively used throughout the later chapters.

2.2.1 Bipartite Graphs

Bipartite Graphs [9] are a good way to visualize the dependences between a set of variables and a set of equations. First, it must be noted that only the dependence and not the functional relationship is shown in such a graph. It is thus a good tool for general equations, but does not include any specifications about the functions. The dependence among variables in a generic equation such as $f(x_1, x_2, x_3) = 0$ can be visualized in a *bipartite graph*, and the equation can be regenerated from that graph. The dependence of variables in a specific equation such as $sin(x_1) + log(x_2) = x_3$ can also be represented in a bipartite graph, but the equation can no longer be regenerated from that graph. Hence the graph is only useful for showing dependences among variables in equations, and not quantitatively specified functionalities, as e.g. in a signal flow graph.

Let us look at an example:

$$f_1(x_1, x_2, x_3) = 0 \tag{2.1}$$

$$f_2(x_1, x_2) = 0 \tag{2.2}$$

$$f_3(x_2, x_3) = 0 \tag{2.3}$$

The system (2.1-2.3) can be visualized through a bipartite graph. On the left side, the set of equations is being listed as leaves, whereas on the right side, the union set

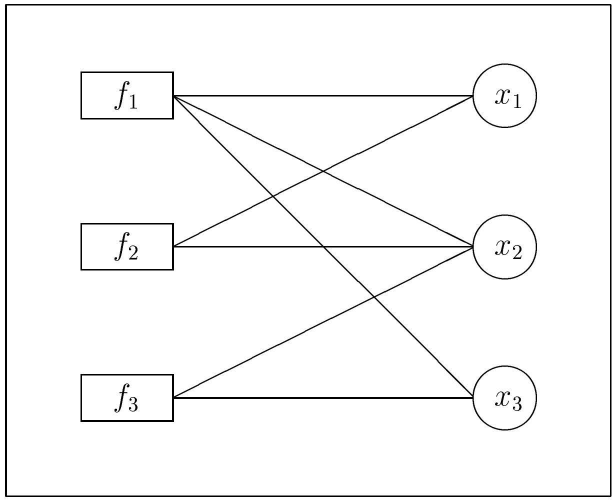


Figure 2.9: Bipartite Graph

of all variables is listed element by element as leaves. Branches connect the two sets of vertices to visualize the dependences among the variables in the equations. The bipartite graph is shown in Fig.2.9.

While this visualization is reversible, it is difficult to see the algebraic loop behind this bipartite graph. Indeed, the variable x_1 can be determined from equation (2.1) or (2.2), but only with knowledge of x_2 and x_3, or x_2 respectively. The variable x_2 itself can be determined from either of the three equations, but only with knowledge of x_1 and x_3, x_1, or x_3 depending on the equation used. Finally, x_3 can be calculated from either equation (2.1) or (2.3) with knowledge of x_1 and x_2, or x_2. Thus, none of the variables can be calculated independently without knowing already at least one of the others, which clearly indicates an algebraic loop. The awkwardness of this

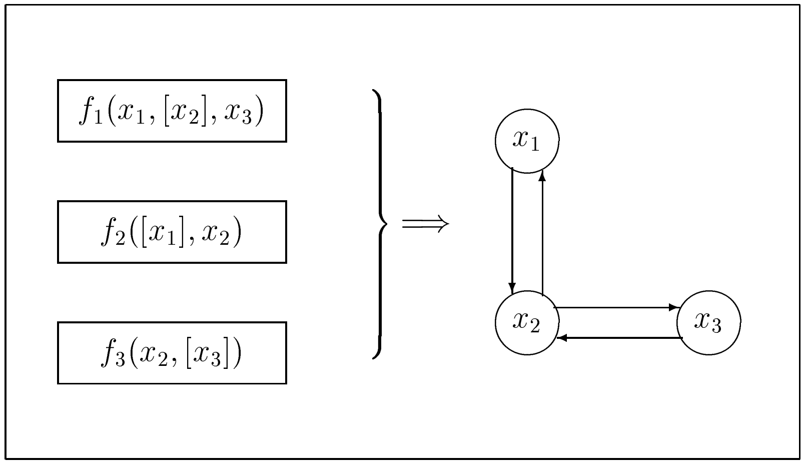


Figure 2.10: Dependence Graph

graph for more complex systems leads to the need for an alternative representation, as provided in the *dependence graphs*. These are described in the next section.

2.2.2 Dependence Graphs

A dependence graph has only one set of elements as leaves, and is therefore less complex than a bipartite graph. However there is no unique way of determining a dependence graph from any set of equations. A computational order has to be determined that is shown in the dependence graph in the form of arrows. In Fig. 2.10, square brackets are used to denote the computational structure. Each equation is solved for the variable marked by square brackets. Hence each equation must contain exactly one set of square brackets, and each loop variable must be marked in exactly one equation. The computational structure indicated in the functions of Fig. 2.10

shows that x_2 is evaluated from equation f_1, x_1 is determined using equation f_2, and x_3 is calculated using equation f_3. The variables x_1 and x_3 are needed to evaluate x_2 from equation f_1, and these *dependences* are indicated by two arrows pointing from the leaves representing the variables x_1 and x_3 to the leaf showing variable x_2. An arrow from x_2 to x_1 indicates that knowledge of x_2 is necessary to determine x_1 using equation f_2. In the same way, an arrow from x_2 to x_3 indicates that knowledge of x_2 is needed to compute x_3 through use of equation f_3. Algebraic structures are recognizable as loops formed by the arrows.

As there exists freedom in the assignment of the computational structure in an algebraically coupled equation system, the dependence graph is not unique, and even the resulting algebraic structures are not invariant to the selection of the computational order. However, once the computational order has been chosen, the resulting algebraic loops can be seen easily from the dependence graph.

2.2.3 Modified Dependence Graphs

For the purpose of this thesis, the need to predetermine the computational structure of an algebraically coupled equation system is not optimal. A slight modification makes it possible to abstract the dependence graph a little further. It is

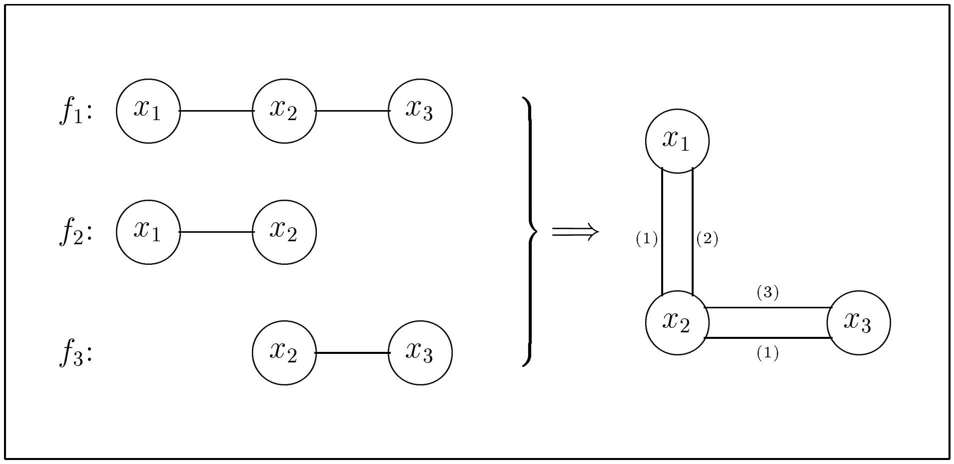


Figure 2.11: Modified Dependence Graph

always possible to eliminate the arrows, making the connections between the variables bi–directional, if the equation number is added to each connection. The so modified dependence graph contains less information than the original one, because the chosen computational structure can no longer be reconstructed from it. However, the modified dependence graph is also more general, because it is possible to draw modified dependence graphs that do not correspond to any possible computational structure, i.e., there exists even more freedom in drawing the modified dependence graph. Yet, the modified dependence graph is better suited for the task at hand.

Fig.2.11 shows one possible version of a modified dependence graph for the same example. On the left side, the equations are depicted together with the variables that they contain. The right side provides the same information in a more compact form. There is no need to draw a connection between variables x_1 and x_3, because these two variables are already connected indirectly through variable x_2 by means of two connections carrying the same equation number.

2.2.4 Modified Dependence Graphs for Time Derivatives

However, we still need one more abstraction level. Since we are dealing with DAE systems and the Pantelides algorithm, we encounter many equations in differentiated form. It may even happen that the same equation needs to be differentiated several times. As an example, we may consider that it was necessary to differentiate equation f_1 three times. It would be possible to represent the differentiated equation f_1 as shown in Fig.2.12(a) using the previously introduced notation. However, this would lead to overloaded figures that are hard to decipher. Therefore, an alternative representation was chosen as shown in Fig.2.12(b). This for derivatives once more modified dependence graph concentrates the information contained in the graph, and thus simplifies it.

This final notation may seem quite cryptic and abstract at first, but it increases the readability of the graphs used later. It is therefore the preferred representation chosen in subsequent chapters of this thesis.

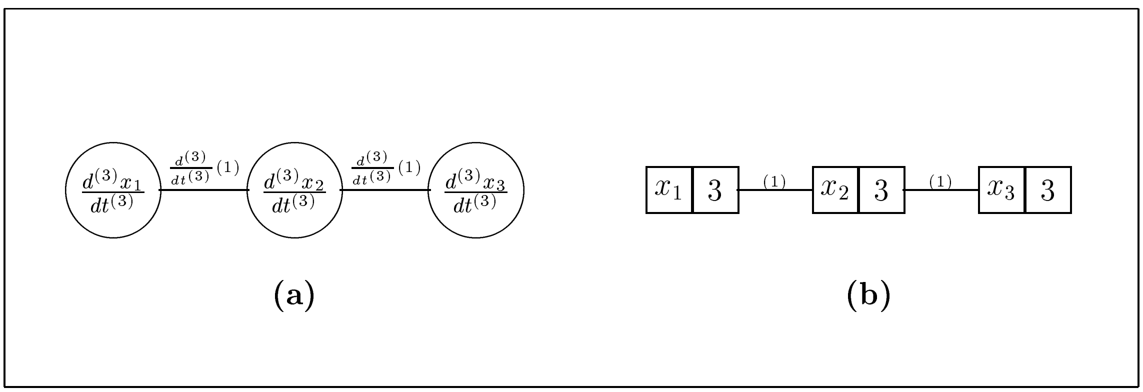


Figure 2.12: For Derivatives Modified Dependence Graph

CHAPTER 3

First Unsuccessful Attempts at Solving the Problem

3.1 The Example Circuit

The example circuit was introduced in the introduction and is shown once more in Fig. 3.1 with the variable names used in the sequel. The circuit, which consists of two diodes, one inductor, one capacitor, and one resistor, contains a structure that makes the modifications of the switch equations difficult. While the modifications necessary to deal with an inductor in series with a diode or a capacitor in parallel with a diode were found quite easily, it is not at all trivial to find the modifications necessary to deal with this sample circuit. Somehow the diode D_2 causes problems that defy attempts at finding a successful modification for the switch equation. If the diode was to the left of the node, as shown in Fig. 3.2, the known modifications would work. In the configuration of Fig. 3.2, the switch equation for diode D_1 is modified to Equ. 3.1 whereas the switch equation for diode D_2 is modified to Equ. 3.2. These two modifications create the necessary algebraic loops as desired. In both switches, the diode characteristic is needed to assure proper modeling. The second switch should only be opened if the current is 0, whereas the first switch should only be closed if the voltage is 0.

$$0 = OS1 \cdot I + (1 - OS1) \cdot \frac{\partial U}{\partial t} \tag{3.1}$$

$$0 = OS2 \cdot \frac{\partial I}{\partial t} + (1 - OS2) \cdot U \tag{3.2}$$

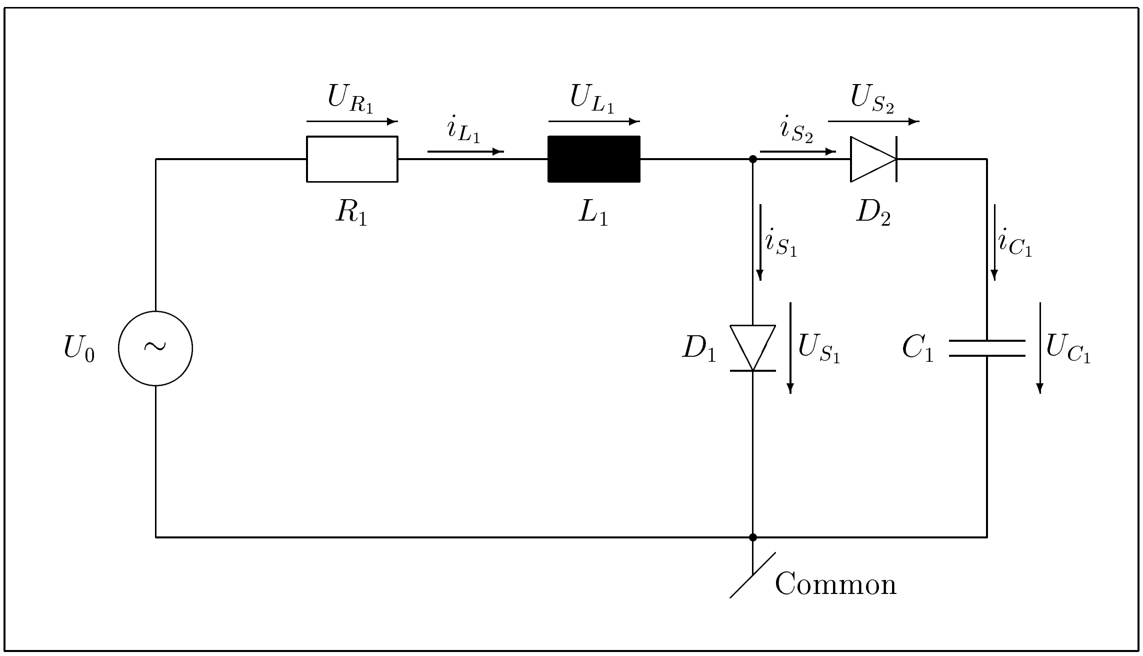


Figure 3.1: Detailed example circuit

However, in the case of the example circuit shown in Fig. 3.1, the required modifications don't follow such a simple pattern. Let us take different views of the model structure to gain a better understanding of the peculiarities of this example.

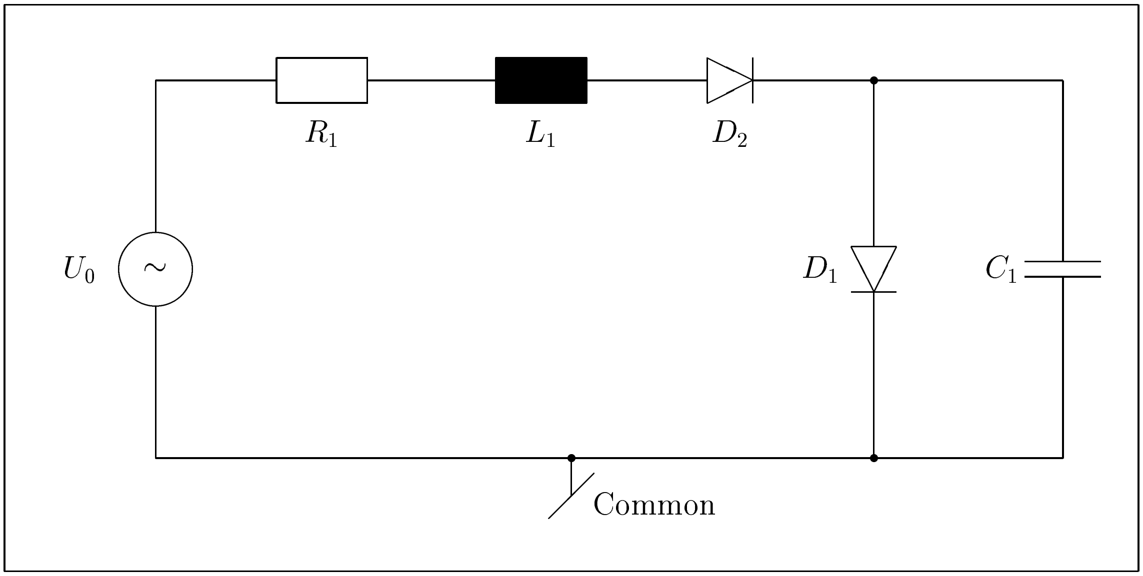


Figure 3.2: Example 2

3.1.1 Bond Graph Causalities for the Example Circuit

There are two possibilities for assigning causality strokes to the bond graph for this example circuit. The two possibilities are shown in Fig. 3.3 and Fig. 3.4. The presence of more than one possibility for assigning the causality strokes leads to the conclusion that there must exist an algebraic loop. Yet, if we feed the model to a simulator, the simulation won't work. The explanation is simple. The algebraic loop contains *both* switches. Once the position of one switch is specified, the second switch position is dictated by the first one. Another aspect is that we have only two possibilities of assigning causality strokes, yet we have four possible switch positions.

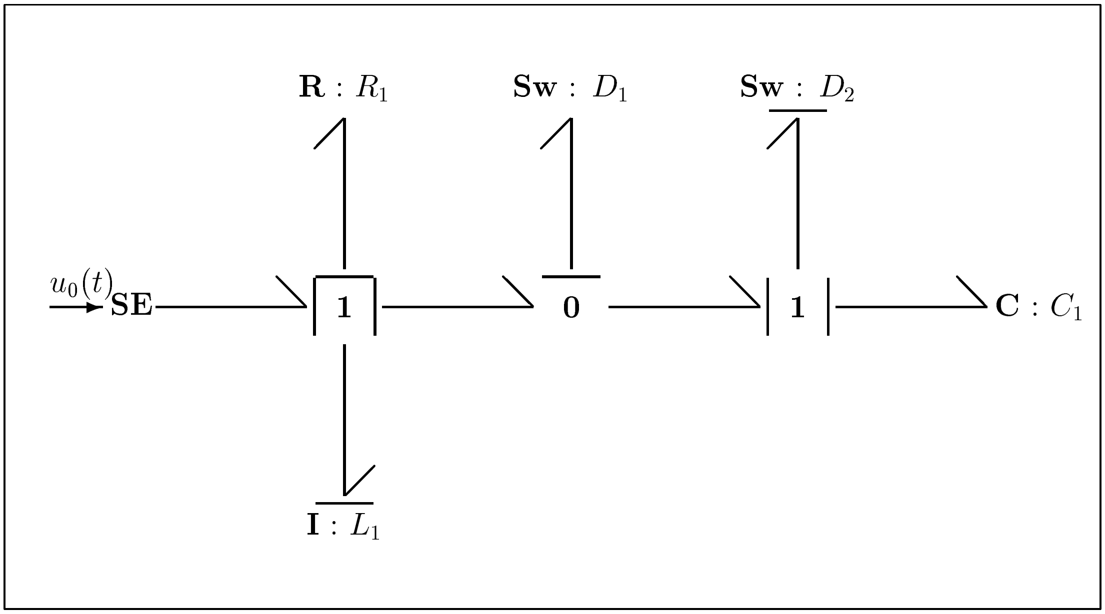


Figure 3.3: Bond Graph Causality (a) for Example 1

In the physical circuit, this can be described as follows.

Let us consider two separate models that represent either switch D_1 closed, represented in the model by a short circuit, or switch D_1 opened, represented in the model by removing the switch element. In the first case, the second switch D_2 must be opened, as otherwise, the voltage across the parallel capacitor would be forced to zero at once irrespective of its former value. In the second case, the second switch D_2 must be closed, as otherwise, the current through the inductor would be forced to zero at once irrespective of its former value.

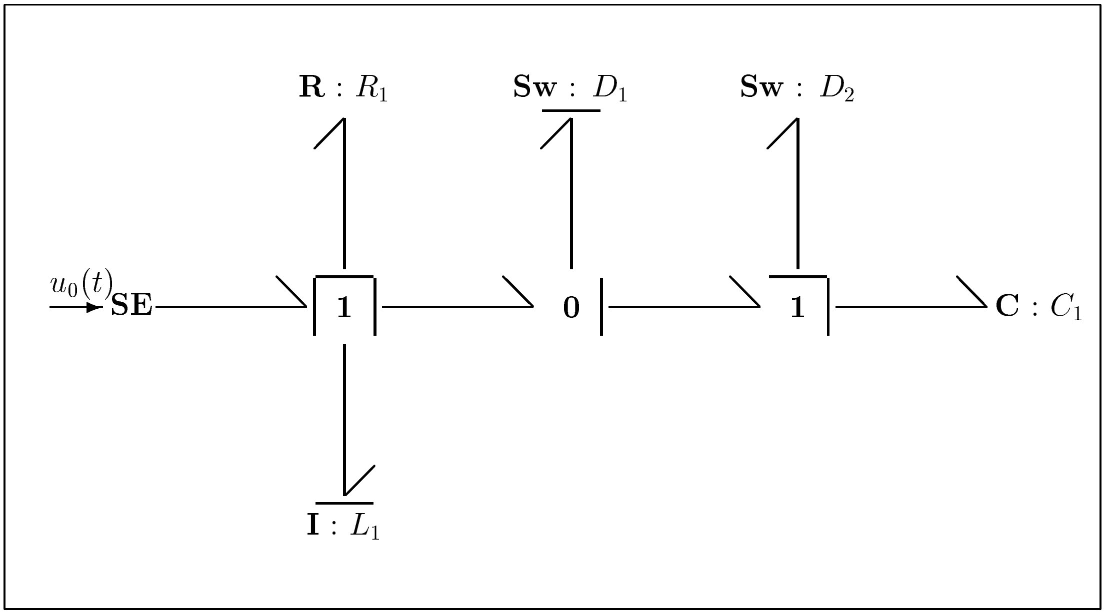


Figure 3.4: Bond Graph Causality (b) for Example 1

3.1.2 DAE Indices for the Example Circuit

Table 3.1 describes the four different combinations of possible switch positions and their associated DAE indices. The *equation order* in all four cases is two, because the system of equations contains two first order derivatives representing the two storage

Table 3.1: DAE Indices for Example Circuit

case	OS1	OS2	Order(Equ)	Order(Phy)	DAE index
1	0	0	2	1	2
2	0	1	2	1	2
3	1	0	2	2	0
4	1	1	2	0	2

elements, the inductor and the capacitor. The *physical order* describes how many storage elements are currently in use, depending on the switch positions. Only in case 3, when switch 1 is open and switch 2 is closed, is the physical order also two. In case 1 the two closed switches short out the capacitor, and thus the capacitor is taken out of the circuit. In case 2, where switch 1 is closed and switch 2 is opened, the capacitor is not properly connected. In case 4, both switches are open, no current flows at all, and the physical order of the system is thus zero. These differences between the equation order and the physical order results in a higher index problem. In the third case, no algebraic loop is present, and the index is zero.

3.2 The Task to be Accomplished

The example circuit contains two diodes, and thus the model has two switch equations. The task that needs to be addressed is the following. Determine the four integer parameters n_1, n_2, n_3, and n_4 in the switch equations, (3.3) and (3.4).

$$0 = OS1 \cdot \frac{\partial^{n_2} I}{\partial t^{n_2}} + (1 - OS1) \cdot \frac{\partial^{n_1} U}{\partial t^{n_1}} \tag{3.3}$$

$$0 = OS2 \cdot \frac{\partial^{n_4} I}{\partial t^{n_4}} + (1 - OS2) \cdot \frac{\partial^{n_3} U}{\partial t^{n_3}} \tag{3.4}$$

such that the two modified switch equations show up in two separate, independent algebraic loops. Unfortunately for the example circuit at hand, these parameters cannot be determined in an easy way. So far, no rules have been derived that would allow us to determine the smallest possible values for the four unknown parameters.

3.3 An Inductive Approach

Since the example model description contains only two switch equations and the DAE index changes in the range from zero to two depending on the four possible switch positions, the first approach was an inductive trial and error method. In each step, values were chosen for the four parameters n_1 to n_4, and the equations were modified accordingly. Then the corresponding determinant of the resulting equation system was determined using Dymola. It was subsequently inspected for singularities. From the knowledge of the singular cases for the given parameter values, a new set of hopefully better suited parameter values was chosen, and the process was repeated. Unfortunately, no progress was made in this manner. Each chosen parameter set resulted in a singularity in at least one of the four cases.

3.4 The Permutation Approach

As the solution could not be found through trial and error, a more structured approach was called for. The process of selecting the parameter vector was made more systematic. First, all permutations up to order one, which result in n_1, n_2, n_3, $n_4 \in \{0, 1\}$, i.e., $2^4 = 16$ possibilities, were examined. Thereafter all permutations up to order two, resulting in n_1, n_2, n_3, $n_4 \in \{0, 1, 2\}$, i.e., $3^4 = 81$ possibilities, were investigated. Of course, these 81 possibilities include only 65 new possibilities as well as the 16 previously investigated possibilities. Unfortunately, this approach did not improve the result at all. The simulation would still only work in a maximum of three out of the four cases. However, through examining the dependence graphs associated with these possibilities, an important first result was achieved. If the two switches showed up in a single algebraic loop, the description always contained a singularity in two cases. If only one switch equation was contained in an algebraic loop, there was only a singularity in one case. This led to the following extension of the requirements for the switch equation:

> *Choosing the position of a switch must not determine variables that are part of the algebraic system in which another switch equation is contained.*

This concludes that a system of equations containing n switches needs at least n independent algebraic loops, each containing a single switch equation. These loops can only be interconnected in such a manner that the connecting leaves cannot be determined prior to the calculation of the loop variables using the matrix solver.

This concept is physically intuitive, as replacing a switch in a model with either an open or a short circuit, should not influence the remaining circuit at all. Replacing a switch corresponds to fixing the value of one discrete switch variable, which should not determine any other switch variables. The switch variables are expected to be independent of each other.

3.5 The Direct Approach

After all the trials of the permutation approach, including some really promising ones, had failed, a search was initiated to find a method to examine what was going wrong in all the previous attempts at modifying the switch equations. In this search for the right tool to examine the modification problem, the dependence graphs appeared to be the most useful tool. Several dependence graphs were examined in full detail. This was quite cumbersome, since it involved several sets of the equations. These sets included the equations from zeroth up to the highest differential order included in a modification of the switch equation. For example in the case of $n_1 = 1$, $n_2 = 2$, $n_3 = 4$, and $n_4 = 3$, which was thought to be close to the solution. Five sets of equations were involved including the zeroth, first, second, third, and fourth order derivatives, and thus, already $5 \cdot 9 + 2 = 47$ equations. This case was thought to be promising, because it should create one loop containing the diode D_1 and the inductor L_1, and a second loop containing the diode D_2 together with the capacitor C_1.

However, neither this nor any other parameter sets created the necessary algebraic loops to guarantee the independence of the discrete switch variables. Yet, examining this case in full detail resulted in a starting point for formulating necessary conditions for the four unknowns n_1, n_2, n_3, and n_4. The two expected loops were not created because of two facts:

- Variables of one proposed loop, or lower order derivatives of variables contained in the proposed loop, had connections to variables forming the second proposed loop. Thus, solving the first algebraic loop resulted in the knowledge of elements of the second loop through the connecting equations, thereby destroying the second algebraic loop.

- Loops were not even created, because a used equation contained a surplus undetermined variable that was not part of the algebraic loop. Hence, the algebraic system was not completely determined.

The conditions were developed to prevent exactly these two ways of destroying the algebraic loops for each switch element using the modified dependence graph. The necessary dependence graphs were constructed using the bond graph notation. As a reminder, the detailed bond graph for the example circuit is shown in Fig. 3.5.

From this bond graph, we can easily determine the following set of equations, where the same variables as in Fig. 3.1 show up. The only additional variable is the

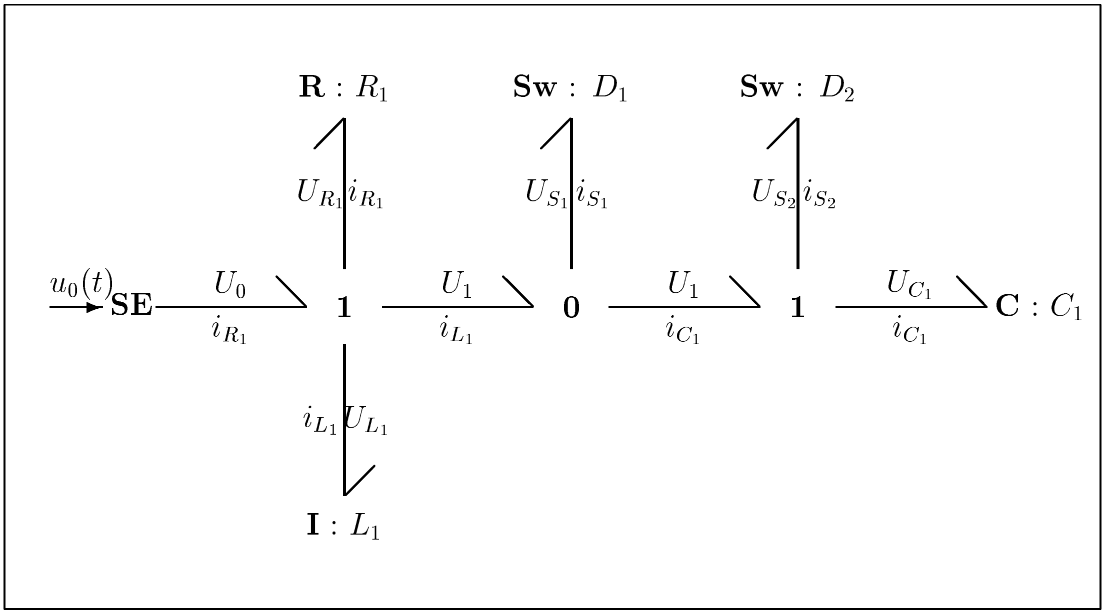


Figure 3.5: Detailed Bond Graph for Example 1

potential U_1 of the node connecting the two diodes and the inductor. The identity $i_{R_1} = i_{L_1}$ is already used to replace i_{R_1} in the set of equations. The identities in equations f_5 and f_{12} were kept in that form, because they contain variables also contained in the switch equations. $OS1$ and $OS2$ in equations f_6 and f_9 are the abbreviated discrete switch variables that determine the positions of the switches. These positions are determined through equations f_7 and f_{10} using the result of an earlier integration step or an initial condition. The operator, **New(.)**, expresses the difference in time instants, and the complete equations f_7 and f_{10} are representing the diode characteristic. This equation system is shown in the modified dependence graph notation in Fig. 3.6

$$U_0 - u_0(t) = f_1 = 0$$

$$U_{R_1} - R_1 \cdot i_{L_1} = f_2 = 0$$

$$-U_{L_1} + L_1 \cdot \frac{\partial i_{L_1}}{\partial t} = f_3 = 0$$

$$U_0 - U_{R_1} - U_1 - U_{L_1} = f_4 = 0$$

$$U_1 - U_{S_1} = f_5 = 0$$

$$OS1 \cdot \frac{\partial^{n_2} i_{S_1}}{\partial t^{n_2}} + (1 - OS1) \cdot \frac{\partial^{n_1} U_{S_1}}{\partial t^{n_1}} = f_6 = 0$$

$$\text{if } [\text{not}(U_{S_1} > 0) \text{ and not } (i_{S_1} > 0)] \text{ then 1 else } 0 - New(OS1) = f_7 = 0$$

$$i_{L_1} - i_{S_1} - i_{C_1} = f_8 = 0$$

$$OS2 \cdot \frac{\partial^{n_4} i_{S_2}}{\partial t^{n_4}} + (1 - OS2) \cdot \frac{\partial^{n_3} U_{S_2}}{\partial t^{n_3}} = f_9 = 0$$

$$\text{if } [\text{not}(U_{S_2} > 0) \text{ and not } (i_{S_2} > 0)] \text{ then 1 else } 0 - New(OS2) = f_{10} = 0$$

$$-i_{C_1} + C_1 \cdot \frac{\partial U_{C_1}}{\partial t} = f_{11} = 0$$

$$i_{C_1} - i_{S_2} = f_{12} = 0$$

$$U_1 - U_{S_2} - U_{C_1} = f_{13} = 0$$

Fig. 3.6 is composed of three parts, a switch part at the bottom, a switch part at the top, and a general equations part in the middle section. Both the bottom and top parts consist of a switch equation and an equation to express the diode characteristic. The dashed line crossing the arrow represents the extraordinary character

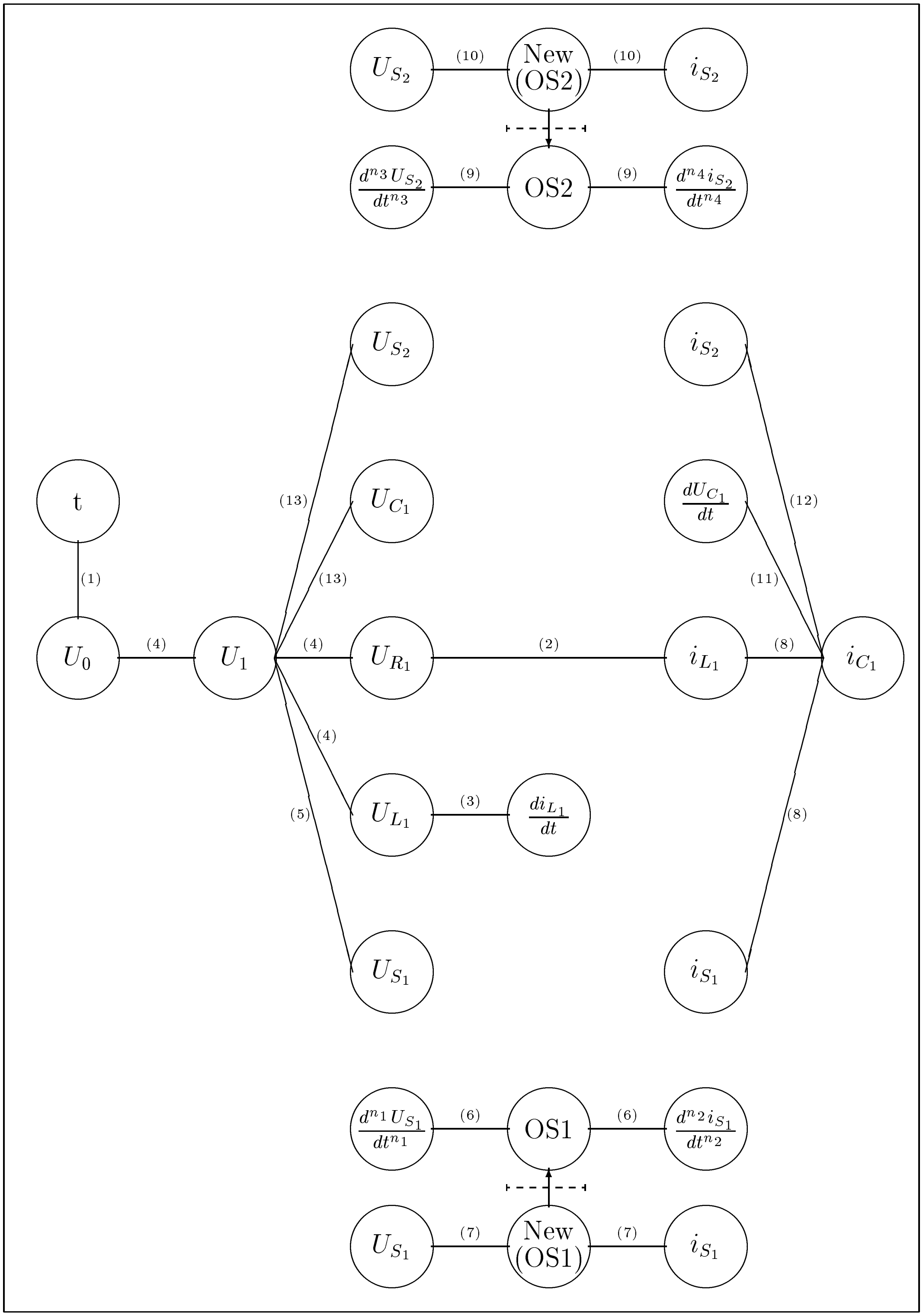


Figure 3.6: Dependence Graph for Example 1

of the equations f_7 and f_{10}. These equations determine values for the discrete switch variable. But these resulting values for *OS1* and *OS2* are time delayed by the **New(.)** operator. In a simulation run, the values of the switch variables for the current simulation step are calculated from previous simulation results, or an initial value. The two switch equations, f_6 and f_9, contain the unknown parameters n_1, n_2, n_3, and n_4, and have thus the needed degrees of freedom to solve the equation modification problem.

The middle part contains all equations except for the two switch equations and the two diode equations. This part forms the equation set of differentiable equations. The equations in this set may be differentiated as needed in order to create the desired independent algebraic loops. Only the highest derivatives of the differentiated variables are algebraic variables. All lower derivatives of these variables, as well as the original undifferentiated variables are *added* state variables that are created through the equation modification process, and that therefore need to be appropriately initialized.

The Pantelides algorithm deals with the equations in a special way. Whenever a variable is differentiated in an equation and lower derivatives of this same variable are contained in other equations, all of these equations are differentiated as well, until the occurrences of this variable in all equations are of the same derivative order. This process of differentiating equations is iterative, as any new differentiation can generate yet higher derivatives that themselves cause other equations to be once more differentiated.

In the modification task, the only requirement for the four parameters, n_1, n_2, n_3, and n_4, is to form two separate loops, each containing a single switch element. The only variables of the middle part of the dependence graph in Fig. 3.6 that are present in more than one differential order, and that are therefore potentially capable of creating new connections in the dependence graph as a consequence of the differentiation process, are U_C and i_L. Both of these variables are contained literally, i.e., in zeroth derivative order, and as first order derivatives. These variables are the only possibilities whereby additional branches can be generated in the dependence graph. These branches of the *highest* orders of the switch equation are needed to form different algebraic loops.

Fig. 3.7 displays the behavior that had previously been examined in the bond graph causality notation. With zero order derivatives, equivalent to $n_1 = n_2 = n_3 = n_4 = 0$, the switch equations in the bottom and top part of the dependence graph have been connected to the middle part. In this graph, we see indeed an algebraic loop, as we concluded earlier, but instead of forming two independent loops, one for

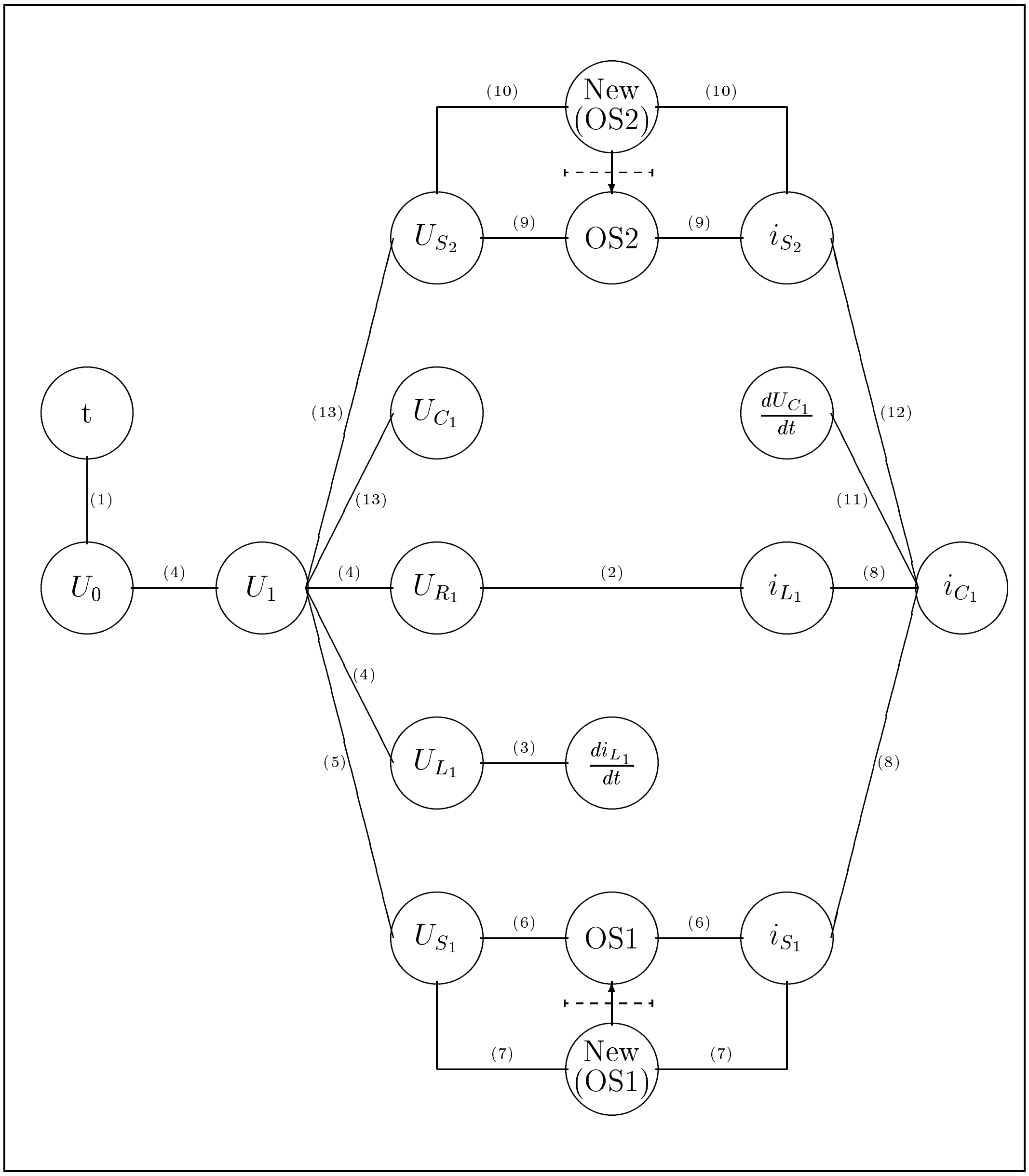


Figure 3.7: Dependence Graph for $n_1 = n_2 = n_3 = n_4 = 0$

each switch element, both switch equations are contained in a single loop in the shape of the number eight. This leads us to the same result, namely that, in the case of the unmodified switch equations, only one discrete switch variable can assume a value independently.

3.5.1 The Two Basic Possibilities

As stated in the previous section, new branches and loops can only be formed through the variables U_C and i_L. Each so–formed loop should contain one of the important equations, f_{11} and f_3, together with either the capacitor or the inductor equation.

Possibility A:

- The diode D_1 is contained in one algebraic loop together with the capacitor C_1. Equations f_6 and f_{11} are contained in the same algebraic structure.
- The diode D_2 is contained in the other algebraic loop together with the inductor L_1. Equations f_9 and f_3 are contained in that algebraic structure.

Possibility B:

- The diode D_1 is contained in one algebraic loop together with the inductor L_1. Equations f_6 and f_3 are contained in the same algebraic structure.

- The diode D_2 is contained in the other algebraic loop together with the capacitor C_1. Equations f_9 and f_{11} are contained in that algebraic structure.

3.5.2 General Considerations About Loops in the Modified Dependence Graph Notation

So far, we have examined the possibilities of forming the necessary loops, but we have not found yet a method to systematically determine values for the four parameters, n_1, n_2, n_3, and n_4. In order to derive such a technique, we need to take another general look at the modified dependence graphs.

Consider the simple example shown in Fig. 3.8. This example describes an interconnected algebraic structure with five equations and five variables. The three equations f_1, f_2, and f_3 form an algebraic loop containing the three variables x_1, x_2, and x_3 if the following requirements are satisfied:

- Variable e_1 is **known** in equation f_5, i.e., equation f_5 is used to compute x_4, otherwise the algebraic loop is underdetermined, because the loop then contains the four unknowns x_1, x_2, x_3, and x_4 within only three equations f_1, f_2, and f_3.

- Variable e_2 is **unknown** in equation f_4, i.e., equation f_4 is used to compute e_2, otherwise x_2 would be determined from equation f_4, x_1 and x_3 could then be determined using equations f_2 and f_1 respectively, and finally x_4 could be determined from equation f_3.

Whenever we have $n > 2$ leaves connected through branches of the same equation in a ring structure, we need exactly $n - 2$ exterior branches to determine exactly $n - 2$ of the variables. A sufficient number of variables in the exterior leaves must be **known** in order to determine these $n - 2$ variables.

Wherever an equation is represented by a single branch, the connecting equations must have at least one unknown variable, so that the variables inside the ring cannot be determined from them.

However, these requirements are only easy to find for ring structures as the one shown in Fig. 3.8. This ring structure indicates a sparsely populated matrix of the associated structure incidence matrix of the DAE system.

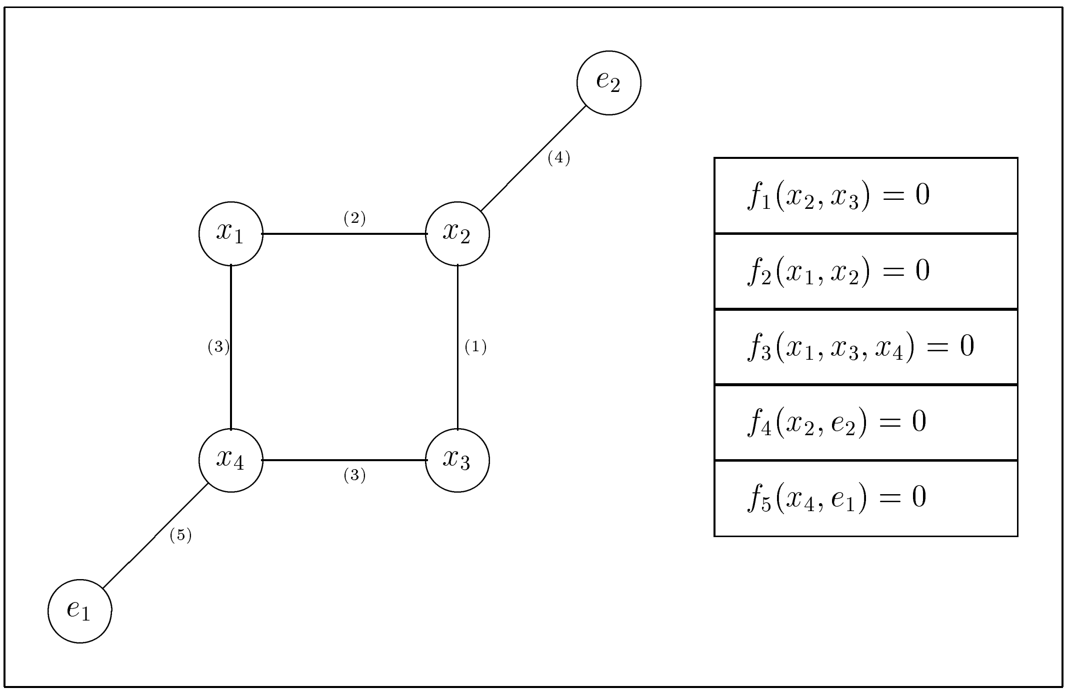


Figure 3.8: Dependence Graph Example (a)

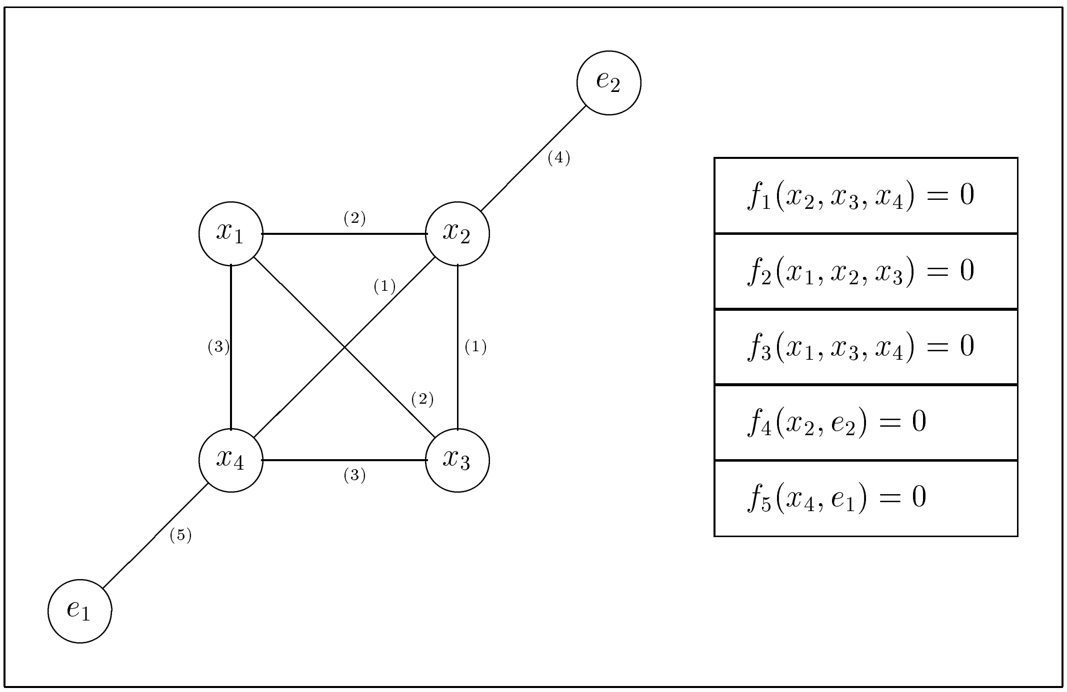


Figure 3.9: Dependence Graph Example (b)

In the case of a densely populated matrix or an interconnected structure, such as the one shown in Fig.3.9, the requirements must be derived from the set of equations. In this modified example, additional interconnections are present inside the ring structure. In this case, the equations f_1, f_2, and f_3 form an algebraic structure with the variables x_1, x_2, x_3, and x_4. This structure is underdetermined as the three equations contain four unknowns. If either e_1 or e_2 are **known**, f_4 or f_5 can be used to compute x_4 or x_2, thereby reducing the algebraic structure to a system of three equations in three unknowns. However, if e_1 and e_2 are both **known**, the algebraic structure is destroyed.

Luckily in the search of the requirements for the example circuit problem, we are dealing with a simple ring structure, and thus, the requirements for the necessary loops can be determined from the dependence graph directly.

3.5.3 Examples for Requirements in Ring Structures

Let us examine some simple graphs to see how, using the previously introduced concepts, a complete set of requirements can be derived.

Fig. 3.10 contains a ring structure containing equations f_1 to f_7 and eight unknowns x_1 to x_8. The external variable e_3 must be **known** in order to determine variable x_8 from equation f_8. Both variables e_1 **and** e_2 must be **known** also, otherwise the equation system is underdetermined, as f_1 then contains at least one surplus unknown

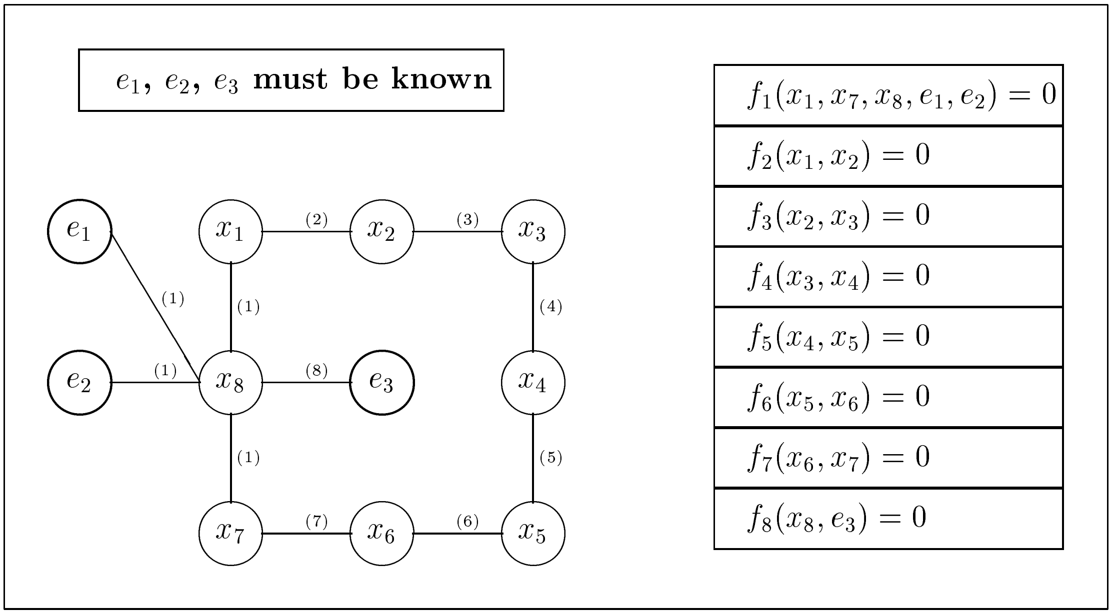


Figure 3.10: Dependence Graph Example (c)

external variable. If this requirement, i.e., e_1 **and** e_2 **and** e_3 are **known**, is satisfied, then there results an algebraic structure with the seven unknowns x_1 to x_7 described by the seven equations f_1 to f_7.

Fig. 3.11 contains a similar ring structure. However this time around, the ring structure contains eight unknowns and eight variables. In this example, e_1 must be **known** in order to have a fully determined algebraic structure, whereas e_2 **and** e_3 must be **unknown**. In this example, knowledge of either e_2, **or** e_3 would allow to calculate all variables of the ring structure. Knowledge of both e_2 **and** e_3 would result in a conflict. The difference to the former example is that here, the external variables are connected to a ring leaf that has a *single branch connection*, whereas in the previous example, two branches for equation f_1 are contained in the ring.

In Fig. 3.12, either e_1 **or** e_2 must be **unknown**, **and** e_3 must be **unknown**. If either e_3 is **known** or both e_1 **and** e_2 are **known**, the leaf x_8 can be calculated using either equation f_9 or f_{10}, and subsequently, all variables of the ring structure could be determined. If all three variables, e_1, e_2, **and** e_3 are **unknown**, the equation system is underdetermined. Finally, if all three variables, e_1, e_2, **and** e_3 are **known**, there results a conflict.

Note that the requirements are **and** connected if several external variables are contained in an equation that forms part of the ring structure. If several external variables are connected through the same equation to a leaf of the ring structure and the equation is not used in the ring structure, the requirements are **or** connected. Finally, if a single external variable is connected to a ring leaf through an equation that does not belong to the ring structure, it must be added in an **and** connection to the other requirements.

Remember that algebraic loops are preventing the problem of a singular denominator. A switch equation contained in an algebraic loop is solved together with all other equations of the algebraic loop. The determinant of the corresponding matrix system can be non–singular in all switch cases. Thus it is our goal to find algebraic

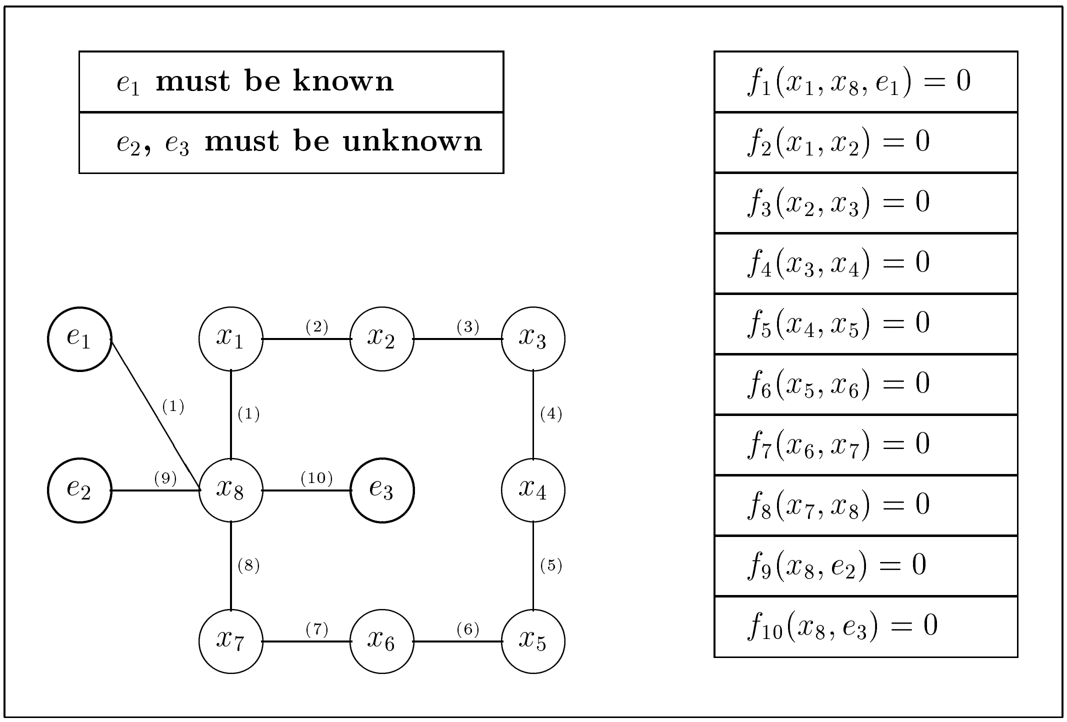


Figure 3.11: Dependence Graph Example (d)

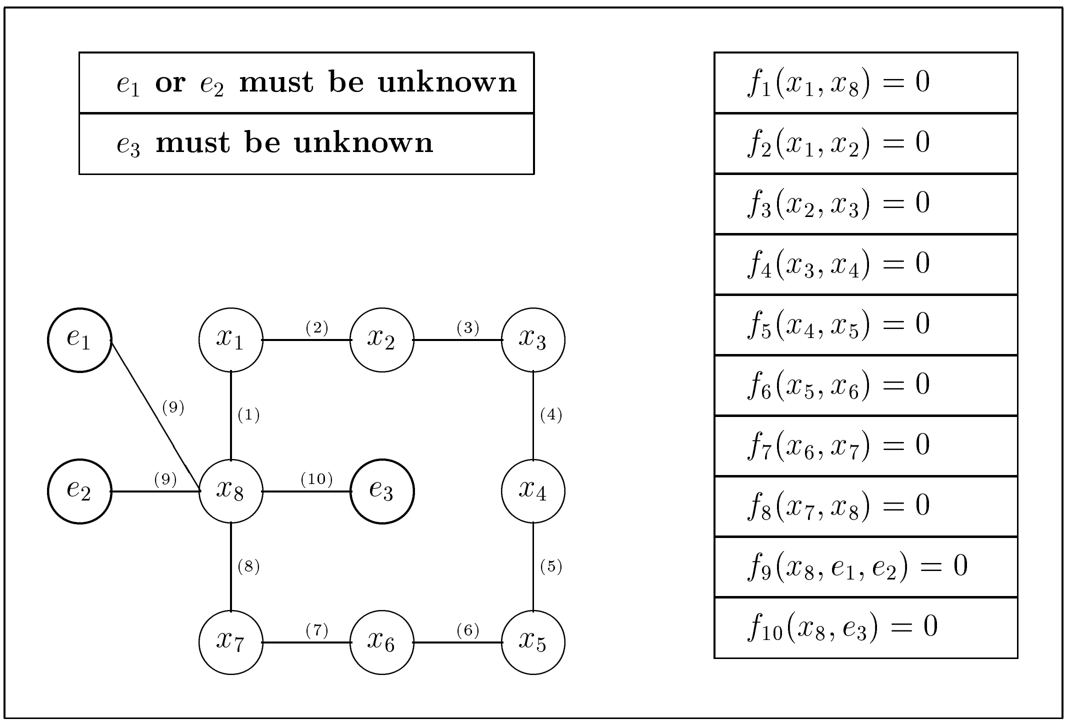


Figure 3.12: Dependence Graph Example (e)

loops containing switch equations. Afterwards we have to assure that the determinant of the matrix system description is indeed unequal zero. If algebraic loops are destroyed, switch equations are solved after either of the two contained variables. This results in solving a single switch equation that causes a singularity in one switch position. Thus the previous requirements provide us with a new means of determining conditions for the existence of algebraic loops.

CHAPTER 4

Conditions for the Example Circuit and Conclusions

This chapter will determine the requirements for the example circuit to form separate algebraic loops each containing one of the switch elements. As stated in the previous chapter, there exist two possibilities for forming separate algebraic loops. These two possibilities are examined separately in the following sections in order to find the requirements and the resulting conditions. The graphs are presented in the time derivatives modified form of the dependence graphs, as redrawn in Fig. 4.1. The two forms of requirements, that is, a variable must be of **known** type or it must be of **unknown** type, are indicated in the notation shown in Fig. 4.2.

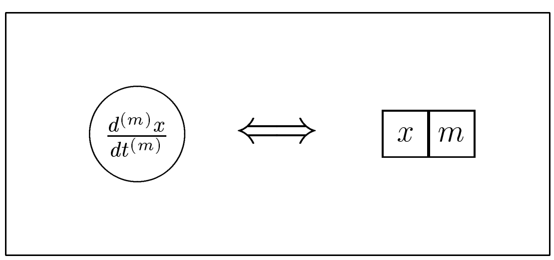


Figure 4.1: Modified Dependence Graph Notation

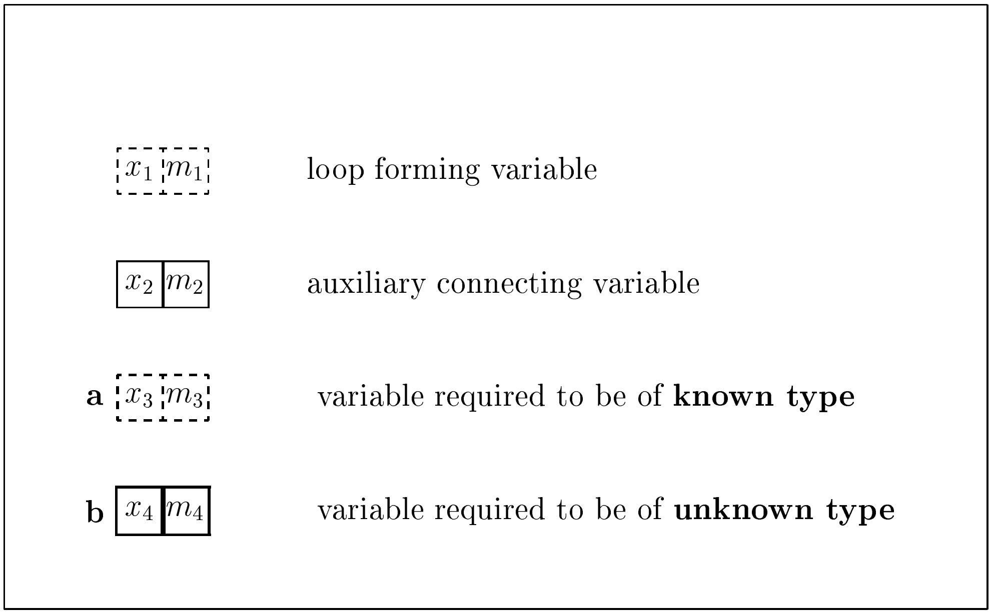


Figure 4.2: Variable Convention in Modified Dependence Graph

As a reminder, the example circuit is described by the following equation set:

$$U_0 - u_0(t) = f_1 = 0$$

$$U_{R_1} - R_1 \cdot i_{L_1} = f_2 = 0$$

$$-U_{L_1} + L_1 \cdot \frac{\partial i_{L_1}}{\partial t} = f_3 = 0$$

$$U_0 - U_{R_1} - U_1 - U_{L_1} = f_4 = 0$$

$$U_1 - U_{S_1} = f_5 = 0$$

$$OS1 \cdot \frac{\partial^{n_2} i_{S_1}}{\partial t^{n_2}} + (1 - OS1) \cdot \frac{\partial^{n_1} U_{S_1}}{\partial t^{n_1}} = f_6 = 0$$

$$\text{if } [\text{not}(U_{S_1} > 0) \text{ and not } (i_{S_1} > 0)] \text{ then 1 else } 0 - New(OS1) = f_7 = 0$$

$$i_{L_1} - i_{S_1} - i_{C_1} = f_8 = 0$$

$$OS2 \cdot \frac{\partial^{n_4} i_{S_2}}{\partial t^{n_4}} + (1 - OS2) \cdot \frac{\partial^{n_3} U_{S_2}}{\partial t^{n_3}} = f_9 = 0$$

$$\text{if } [\text{not}(U_{S_2} > 0) \text{ and not } (i_{S_2} > 0)] \text{ then } 1 \text{ else } 0 - New(OS2) = f_{10} = 0$$

$$-i_{C_1} + C_1 \cdot \frac{\partial U_{C_1}}{\partial t} = f_{11} = 0$$

$$i_{C_1} - i_{S_2} = f_{12} = 0$$

$$U_1 - U_{S_2} - U_{C_1} = f_{13} = 0$$

which was shown in Fig. 3.6

The modified dependence graphs consist only of the highest derivatives of each variable, as well as, all the connections that form either kind of the requirements. As the modifications of the switch equations result in an introduction of $n_1 + n_2 + n_3 + n_4$ new state variables, a detailed full graph should depict all these state variables as well. However, since state variables are always of **known** type, they do not contribute in any way to the loop structure. Therefore, for the examination of the requirements of how loops are formed and preserved, we only need to consider the loop–forming variables, as well as, all connections that form a requirement for the existence of the loops.

For each possibility, two graphs are used, one containing the first switch element diode D_1, the other containing the second switch element diode D_2. The loop–forming variables are of differential order n_1 or n_2 for the first loop, and of n_3 or n_4 for the second loop. The basis of a graph for the second loop is the second switch equation f_9, whereas the basis of a graph for the first loop is the first switch equation f_6. The equations f_7, and f_{10} that describe the diode characteristics are not contained in the graphs, because they only represent a switch characteristic and do not influence the structure of the algebraic system in any way.

The basic equation for the first loop contains the leaves U_{S_1} and i_{S_1}, which are of orders n_1 and n_2, respectively, as well as the leaf $OS1$. These leaves and the two branches representing equation f_6 form the bottom part of Fig. 3.6. The loop for the first switch element is built connecting a set of equations of differential order n_1 to the U_{S_1} leaf, and a set of equations of differential order n_2 to the i_{S_1} leaf. These two equation sets are the n_1 and n_2 times differentiated middle part of Fig. 3.6. The left and right parts of each graph are connected at the U_{C_1} leaf in possibility A, and at the I_{L_1} leaf in possibility B.

The basic equation for the second loop contains the leaves U_{S_2} and i_{S_2} that are of orders n_3 and n_4, respectively, as well as the leaf $OS2$. These leaves and the two branches representing equation f_9 form the top part of Fig. 3.6. The loop for the second switch element is built connecting a set of equations of differential order n_3 to the U_{S_2} leaf, and a set of equations of differential order n_4 to the i_{S_2} leaf. These two equation sets are the n_3 and n_4 times differentiated middle part of Fig. 3.6. The left and right parts of each graph are connected at the I_{L_1} leaf in possibility A, and at the U_{C_1} leaf in possibility B.

Equations f_3 and f_{11} form the links between variables of different differential orders. They impose constraints on the values that the parameters n_1, n_2, n_3, and n_4 can assume.

In order to keep the graphs reasonably simple, the graphs show only the loop–forming variables, i.e., the variables that are coupled together in an algebraic system as well as the variables that influence the existence of the loops. These *non–loop–forming* variables, contained as side connections to the ring structure, are either additional variables contained in one of the loop–forming equations or variables contained in a non–loop–forming equation that depends on a loop–forming variable. The union of all loop–forming variables of the first and second loop contains all end leaves of side connections. Of course, these end leaves are of different differential order than that of the loop–forming variables. The comparison of these different differential orders results in the set of mathematical conditions to be derived.

The graphs show the highest differential order of switch voltages and switch currents only. The graphs show that U_{S_1} is of order n_1, i_{S_1} is of order n_2, U_{S_2} is of order n_3 , and i_{S_2} is of order n_4. All lower derivatives of these four variables are introduced state variables. As with the original state variables, these are considered to be known from a previous simulation step or from initial conditions. The introduced state variables can be used to calculate variables using the differentiated equation sets of lower orders than n_1, n_2, n_3, and n_4. Starting from $(\frac{\partial^{n_1-1}U_{S_1}}{\partial t^{n_1-1}}, \ldots, \frac{\partial^2 U_{S_1}}{\partial t^2}, \frac{\partial^1 U_{S_1}}{\partial t^1}, U_{S_1})$, $(\frac{\partial^{n_2-1}i_{S_1}}{\partial t^{n_2-1}}, \ldots, \frac{\partial^2 i_{S_1}}{\partial t^2}, \frac{\partial^1 i_{S_1}}{\partial t^1}, i_{S_1})$, $(\frac{\partial^{n_3-1}U_{S_2}}{\partial t^{n_3-1}}, \ldots, \frac{\partial^2 U_{S_2}}{\partial t^2}, \frac{\partial^1 U_{S_2}}{\partial t^1}, U_{S_2})$, and $(\frac{\partial^{n_4-1}i_{S_2}}{\partial t^{n_4-1}}, \ldots, \frac{\partial^2 i_{S_2}}{\partial t^2}, \frac{\partial^1 i_{S_2}}{\partial t^1}, i_{S_2})$ all variables of lower differential orders than n_1, n_2, n_3, and n_4 can be calculated. This leads to a set of **known** variables that contains all lower derivatives of loop–forming variables. The set of all lower derivatives of loop–forming variables forms the *base set* for the conditions. This base set and the graphs together result in a set of requirements for each of the two possibilities. A requirement is that a connected variable must be either **known** or **unknown** to form the loop in the described fashion. The conversion from the graphical requirement to a mathematical condition results in inequalities that are formed in two different ways, depending on the type of the requirement.

For the **unknown** type, the order of the requirement variable must be **greater** than the corresponding order of the same variable contained in one of the loops.

Hence this variable is **unknown**, because it is not a state variable and cannot be determined by solving one of the loops.

For the **known** type, the order of the requirement variable must be **smaller** than the corresponding order of the same variable contained in one of the loops. Hence this variable is **known** as a state variable. An exemption represents the presence of derivatives of the variable U_0, because these derivatives can always be computed using the first equation f_1. In the multicondition cases, this variable shows up in **and** connections in the **known** type, and in **or** connections in the **unknown** type, thus it causes no condition at all. However, the leaves for this variable are included in the graphs for completeness.

In Table 4.1, Table 4.2, Table 4.3, and Table 4.4, the inequalities $>$ and $<$ must be used, if the two loops are to be truly decoupled ring structures. The equal signs allow for couplings between the two loops and/or multiply connected algebraic structures within each of the loops.

4.1 Conditions for Possibility A

The graph for the second loop containing the switch element D_2 and the inductor L_1 is shown in Fig. 4.3, while the graph for the first loop containing the switch

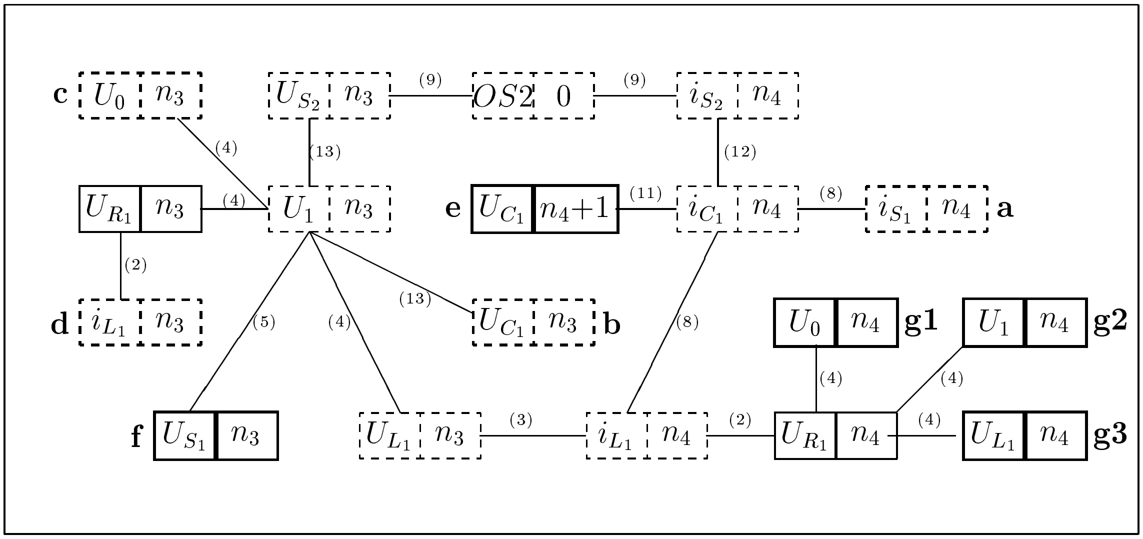


Figure 4.3: Modified Dependence Graph Possibility A Loop 2

element D_1 and the capacitor C_1 is shown in Fig. 4.4.

4.1.1 Requirements for Loop 2 in Possibility A

Fig. 4.3 shows the following requirements:

- **a and b and c and d** must be **known**
- **e and f and (g1 or g2 or g3)** must be **unknown**
- $n_3 + 1 = n_4$ because of the inductor equation

Table 4.1: Conditions for Loop 2 in Possibility A

restriction	type	condition
a	known	$n_4 \leq n_2$
b	known	$n_3 \leq n_1$
c	known	no condition as $U_0 = f(t)$
d	known	$n_3 \leq n_4$
e	unknown	$n_4 + 1 \geq n_1$
f	unknown	$n_3 \geq n_1$
g1 or **g2** or **g3**	unknown	**g1** : no condition as $U_0 = f(t)$, $[(n_4 \geq n_3) \wedge (n_4 \geq n_1)] \vee (n_4 \geq n_3)$
f_3	inductor	$n_3 + 1 = n_4$

4.1.2 Conditions for Loop 2 in Possibility A

4.1.3 Requirements for Loop 1 in Possibility A

Fig. 4.4 shows the following requirements:

- **h and i** must be **known**
- **(j1 or j2 or j3) and k** must be **unknown**
- $n_1 = n_2 + 1$ because of the capacitor equation

4.1.4 Conditions for Loop 1 in Possibility A

4.1.5 Result of Combined Conditions for Possibility A

It is impossible to find a solution for n_{1S}, n_{2S}, n_{3S}, and n_{4S} that satisfies all the conditions **a** to **k**. From conditions **h** and **k**, it can be concluded that $n_2 = n_4$. From

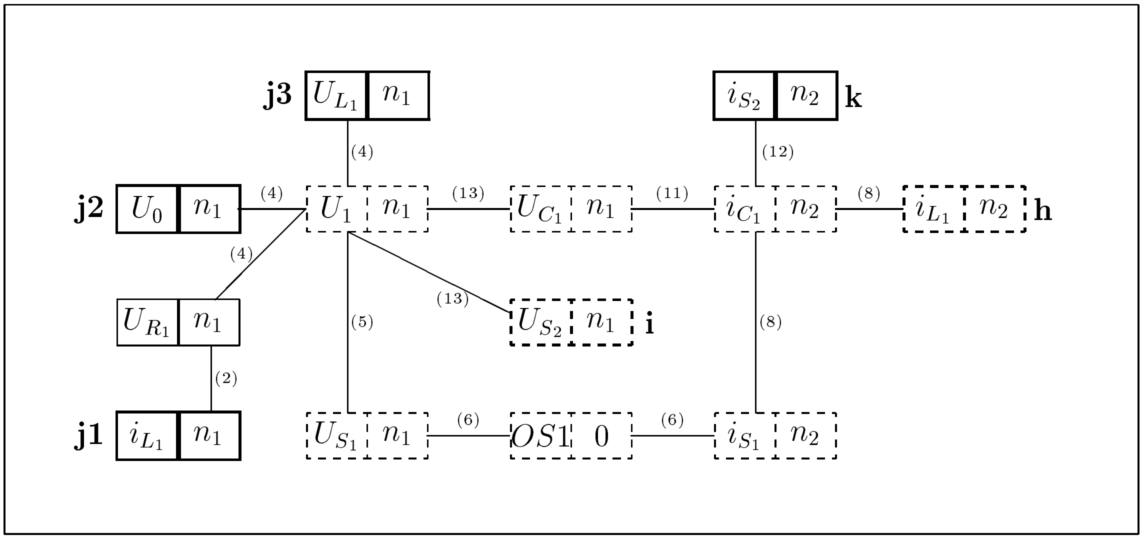


Figure 4.4: Modified Dependence Graph Possibility A Loop 1

conditions **b** and **f**, it must be concluded that $n_1 = n_3$. However, these two conditions are in conflict with the inductor and capacitor constraints. Let us assume that $n_3 = 1$. From the inductor constraint, we conclude that $n_4 = n_3 + 1 = 2$. However, $n_2 = n_4 = 2$. Hence, from the capacitor constraint, we find that $n_1 = n_2 + 1 = 3$. Yet, $n_3 = n_1 = 3$, which is in contradiction with the original assumption.

4.2 Conditions for Possibility B

The graph for the second loop containing the switch element D_2 and the

Table 4.2: Conditions for Loop 1 in Possibility A

restriction	type	condition
h	known	$n_2 \leq n_4$
i	known	$n_1 \leq n_3$
j1 or**j2** or **j3**	unknown	**j2** : no condition as $U_0 = f(t)$, $(n_1 \geq n_4) \vee (n_1 \geq n_3)$
k	unknown	$n_2 \geq n_4$
f_{11}	capacitor	$n_1 = n_2 + 1$

capacitor C_1 is shown in Fig. 4.5, while the graph for the first loop containing the switch element D_1 and the inductor L_1 is shown in Fig. 4.6.

4.2.1 Requirements for Loop 2 in Possibility B

Fig. 4.5 shows the following requirements

- **a or (b1 and b2 and b3)** must be **known**
- **(c1 or c2)** must be **unknown**
- $n_3 = n_4 + 1$ because of the capacitor equation

4.2.2 Conditions for Loop 2 in Possibility B

4.2.3 Requirements for Loop 1 in Possibility B

Fig. 4.6 shows the following requirements:

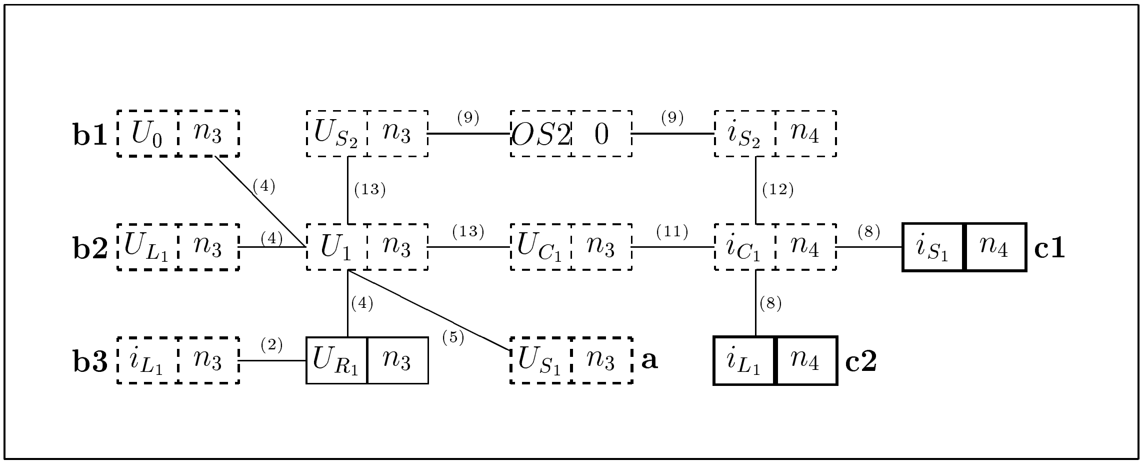


Figure 4.5: Modified Dependence Graph Possibility B Loop 2

Table 4.3: Conditions for Loop 2 in Possibility B

restriction	type	condition
a or (**b1** and **b2** and **b3**)	known	**b1** : no condition as $U_0 = f(t)$, $[(n_3 \leq n_1) \wedge (n_3 \leq n_2)] \vee [n_3 \leq n_1]$
(**c1** or **c2**)	unknown	$(n_4 \geq n_2) \vee (n_4 \geq n_2)$
f_{11}	capacitor	$n_3 = n_4 + 1$

- **d and e and (f1 or f2)** must be **known**
- **(g1 or g2) and (i1 or i2 or i3)** must be **unknown**
- $n_1 + 1 = n_2$ because of the inductor equation

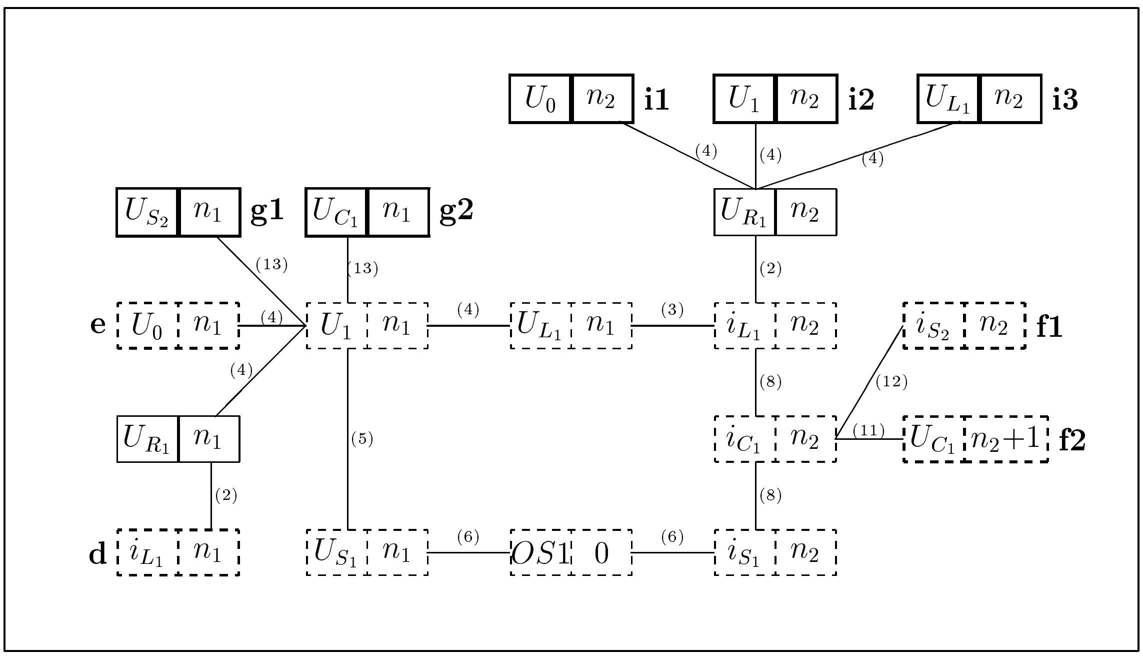


Figure 4.6: Modified Dependence Graph Possibility B Loop 1

Table 4.4: Conditions for Loop 1 in Possibility B

restriction	type	condition
d	known	$n_1 \leq n_2$
e	known	no condition as $U_0 = f(t)$
f1 or **f2**	known	$(n_2 \leq n_4) \vee (n_2 + 1 \leq n_3)$
g1 or **g2**	unknown	$(n_1 \geq n_3) \vee (n_1 \geq n_3)$
i1 or **i2** or **i3**	unknown	**i1** : no condition as $U_0 = f(t)$, $[(n_2 \geq n_1) \wedge (n_2 \geq n_3)] \vee (n_2 \geq n_1)$
f_3	inductor	$n_1 + 1 = n_2$

4.2.4 Conditions for Loop 1 in Possibility B

4.2.5 Result of Combined Conditions for Possibility B

It is impossible to find a solution for n_{1S}, n_{2S}, n_{3S}, and n_{4S} that satisfies all the conditions **a** to **i**. From conditions **g**, **d**, and **c**, we can conclude that $n_3 \leq n_1 \leq n_2 \leq n_4$. The inductor constraint $n_2 = n_1 + 1$, allows us to make the above magnitude relationship even more stringent: $n_3 \leq n_1 < n_2 \leq n_4$. Thus, $n_3 < n_4$. However, the capacitor constraint requires that $n_3 = n_4 + 1 > n_4$, which is in contradiction with the above.

4.3 Verification of the Method

The second, slightly modified, example has almost the same equation structure as the first example. The detailed circuit is shown in Fig. 4.7. The detailed bond graph is shown in Fig. 4.8. This second example is used to verify that the previously derived

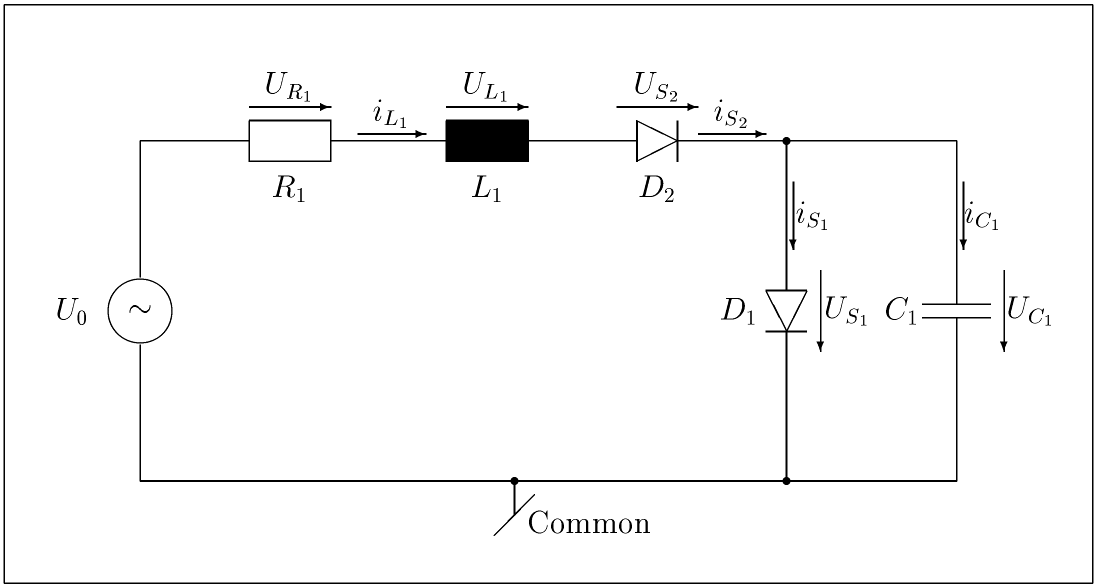


Figure 4.7: Detailed Example 2

method for finding conditions results in the right solution. As stated earlier, the necessary modifications for this example circuit are already known. The modifications result in the parameters $n_{1S} = 1$, $n_{2S} = 0$, $n_{3S} = 0$, and $n_{4S} = 1$. Let us verify that the previously introduced method results in the same set of parameter values.

First, we need the set of equations that can easily be formulated using the bond graph technique.

$$U_0 - u_0(t) = f_1 = 0$$

$$U_{R_1} - R_1 \cdot i_{L_1} = f_2 = 0$$

$$-U_{L_1} + L_1 \cdot \frac{\partial i_{L_1}}{\partial t} = f_3 = 0$$

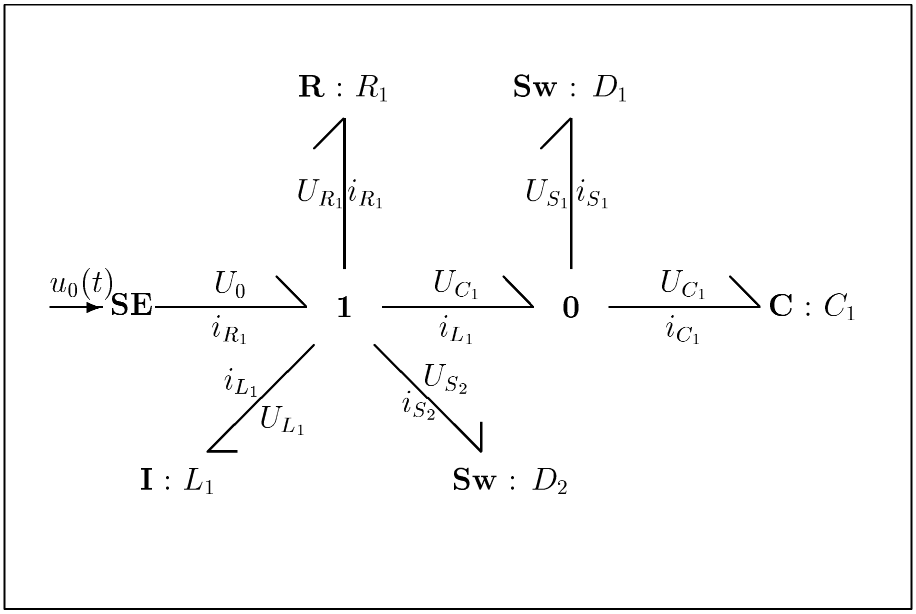


Figure 4.8: Detailed Bond Graph for Example 2

$$U_0 - U_{R_1} - U_{S_1} - U_{S_2} - U_{L_1} = f_4 = 0$$

$$U_{C_1} - U_{S_1} = f_5 = 0$$

$$OS1 \cdot \frac{\partial^{n_2} i_{S_1}}{\partial t^{n_2}} + (1 - OS1) \cdot \frac{\partial^{n_1} U_{S_1}}{\partial t^{n_1}} = f_6 = 0$$

$$\text{if}\,[\text{not}(U_{S_1} > 0)\ \text{and not}\,(i_{S_1} > 0)]\ \text{then 1 else 0} - New(OS1) = f_7 = 0$$

$$i_{L_1} - i_{S_1} - i_{C_1} = f_8 = 0$$

$$OS2 \cdot \frac{\partial^{n_4} i_{S_2}}{\partial t^{n_4}} + (1 - OS2) \cdot \frac{\partial^{n_3} U_{S_2}}{\partial t^{n_3}} = f_9 = 0$$

$$\text{if}\,[\text{not}(U_{S_2} > 0)\ \text{and not}\,(i_{S_2} > 0)]\ \text{then 1 else 0} - New(OS2) = f_{10} = 0$$

$$-i_{C_1} + C_1 \cdot \frac{\partial U_{C_1}}{\partial t} = f_{11} = 0$$

$$i_{L_1} - i_{S_2} = f_{12} = 0$$

A dependence graph representing this equation system is shown in Fig. 4.9.

The dependence graph is very similar to the dependence graph for the first example. To build the necessary algebraic loops, we are once again faced with the same two basic possibilities.

Possibility A:

- The diode D_1 is contained in one algebraic loop together with the capacitor C_1. Equations f_6 and f_{11} are contained in the same algebraic structure.
- The diode D_2 is contained in the other algebraic loop together with the inductor L_1. Equations f_3 and f_9 are contained in that algebraic structure.

Possibility B:

- The diode D_1 is contained in one algebraic loop together with the inductor L_1. Equations f_3 and f_6 are contained in the same algebraic structure.
- The diode D_2 is contained in the other algebraic loop together with the capacitor C_1. Equations f_9 and f_{11} are contained in that algebraic structure.

4.3.1 Requirements for Both Loops in Possibility A

Fig. 4.10 shows the following requirements:

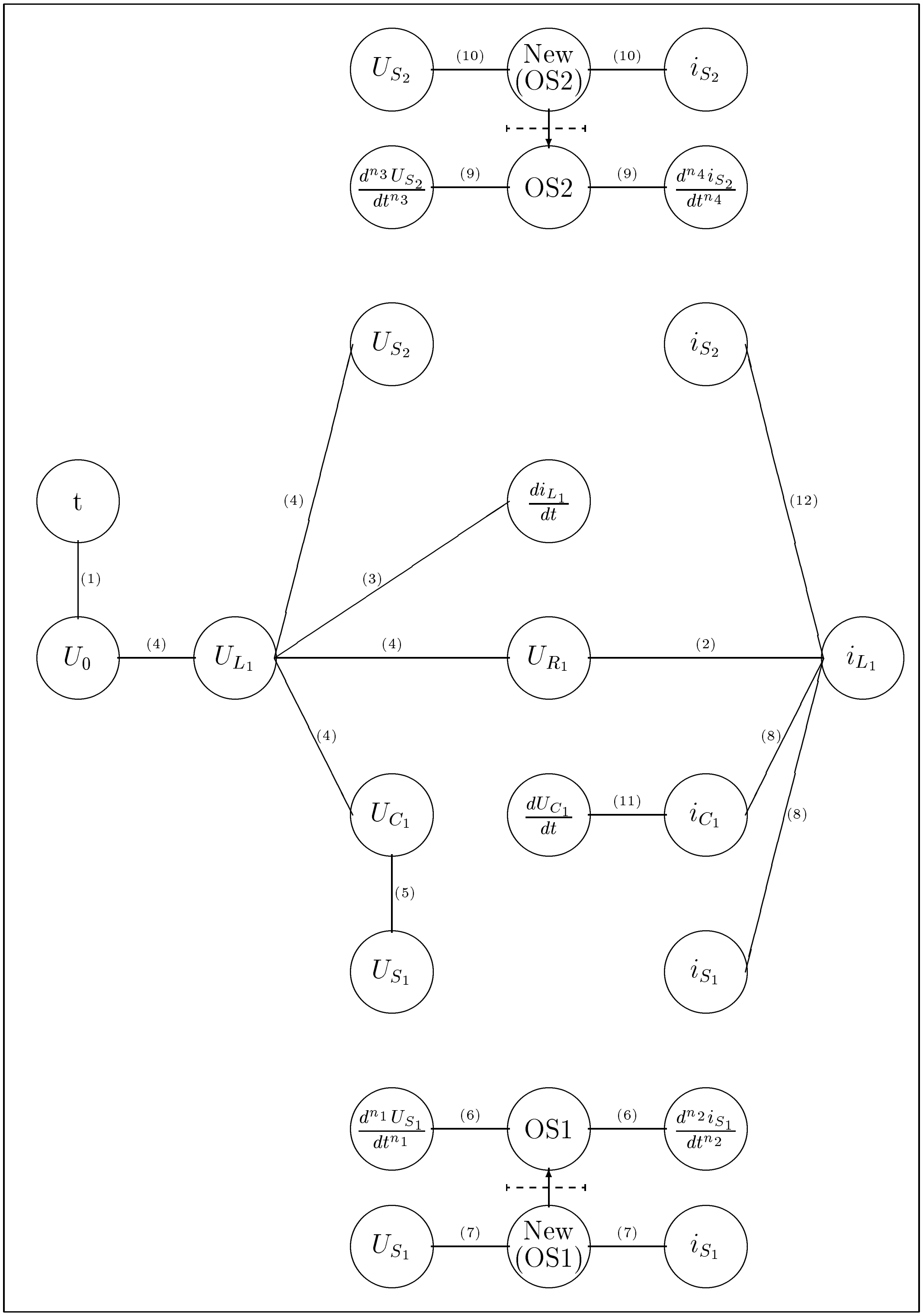


Figure 4.9: Dependence Graph for Example 2

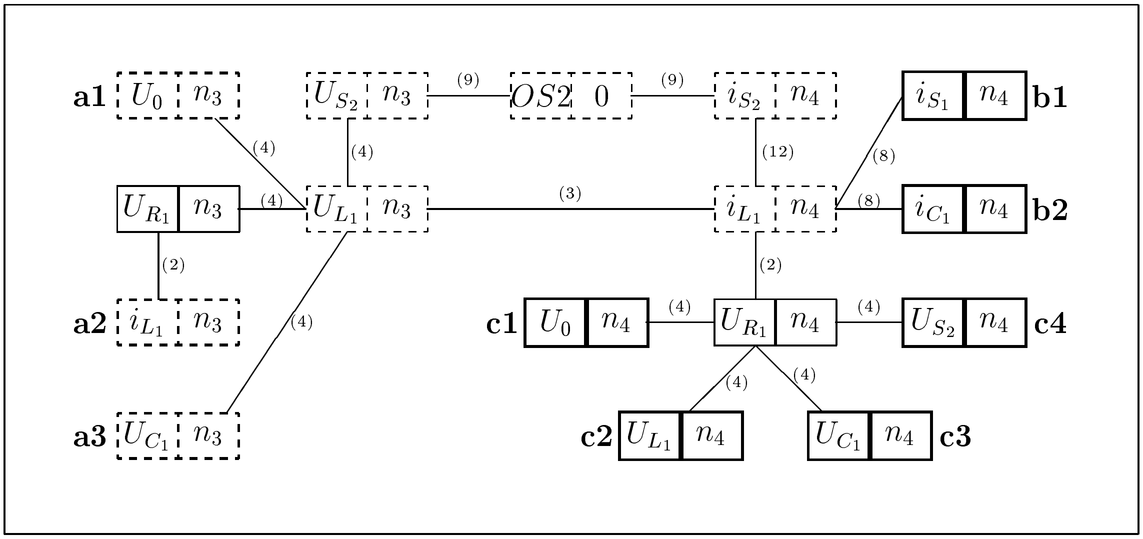


Figure 4.10: Example 2 MDG Possibility A Loop 2

- **(a1 and a2 and a3)** must be **known**

- **(b1 or b2) and (c1 or c2 or c3 or c4)** must be **unknown**

- $n_3 + 1 = n_4$ because of the inductor equation

Fig. 4.11 shows the following requirements:

- **d** must be **known**

- **(e1 or e2 or e3 or e4)** must be **unknown**

- $n_1 = n_2 + 1$ because of the capacitor equation

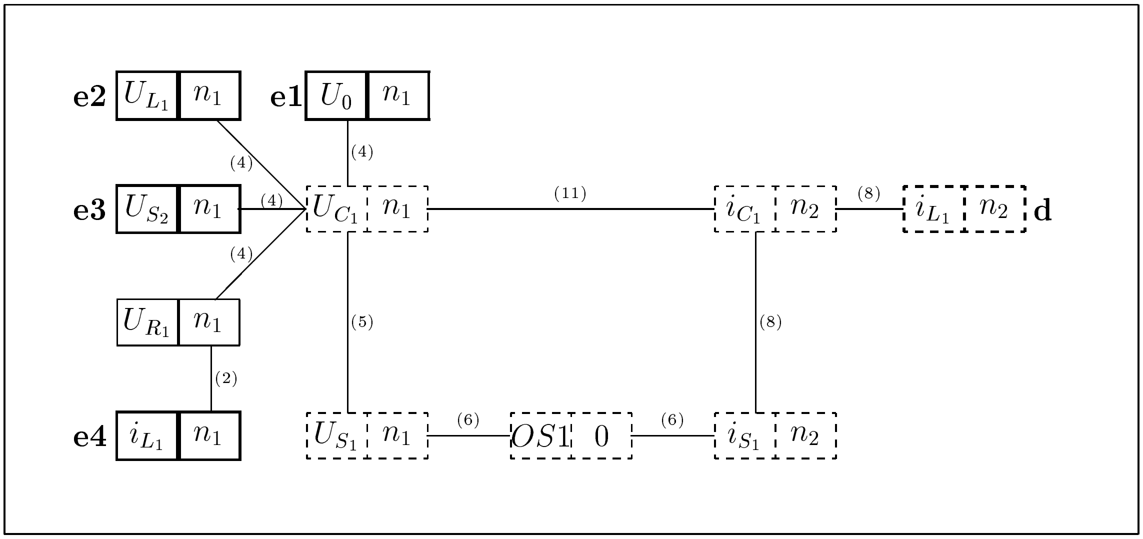


Figure 4.11: Example 2 MDG Possibility A Loop 1

4.3.2 Conditions for Both Loops in Possibility A

The restrictions can be simplified with the two equations $n_3 + 1 = n_4$ and $n_1 = n_2 + 1$:

- **a** $\Rightarrow (n_3 \leq n_3 + 1) \wedge (n_3 \leq n_1) \Rightarrow n_3 \leq n_1$
- **b** $\Rightarrow n_4 \geq n_2$
- **c** $\Rightarrow (n_3 + 1 \geq n_3) \vee (n_3 + 1 \geq n_1) \vee (n_3 + 1 \geq n_3) \Rightarrow$ always fulfilled
- **d** $\Rightarrow n_2 \leq n_4$

Table 4.5: Conditions for Both Loops in Possibility A

restriction	type	condition
a1 and **a2** and **a3**	known	**a1** : no condition as $U_0 = f(t)$, $(n_3 \leq n_4) \wedge (n_3 \leq n_1)$
b1 or **b2**	unknown	$(n_4 \geq n_2) \vee (n_4 \geq n_2)$
c1 or **c2** or **c3** or **c4**	unknown	**c1** : no condition as $U_0 = f(t)$, $(n_4 \geq n_3) \vee (n_4 \geq n_1) \vee (n_4 \geq n_3)$
f_3	inductor	$n_3 + 1 = n_4$
d	known	$n_2 \leq n_4$
e1 or **e2** or **e3** or **e4**	unknown	**e1** : no condition as $U_0 = f(t)$, $(n_1 \geq n_3) \vee (n_1 \geq n_3) \vee (n_1 \geq n_4)$
f_{11}	capacitor	$n_1 = n_2 + 1$

- **e** $\Rightarrow (n_1 \geq n_3) \vee (n_1 \geq n_3) \vee (n_1 \geq n_3 + 1) \Rightarrow n_1 \geq n_3$

All restrictions can be summarized into $n_3 \leq n_1$ and $n_2 \leq n_4$ together with $n_1 = n_2 + 1$ and $n_4 = n_3 + 1$. Let us try the simplest case, where $n_2 = n_3 = 0$. Consequently, $n_1 = n_4 = 1$, which satisfies the two inequalities as well. Hence we have found a valid solution satisfying all equations and restrictions: $n_{1S} = 1$, $n_{2S} = 0$, $n_{3S} = 0$, and $n_{4S} = 1$. This solution is exactly the solution that had been found earlier using a heuristic approach. Of course, each of the parameters could be increased by any positive integer, which would generate other, yet more complicated solutions. Yet, we are only interested in the simplest solution, as all other solutions can be derived from it.

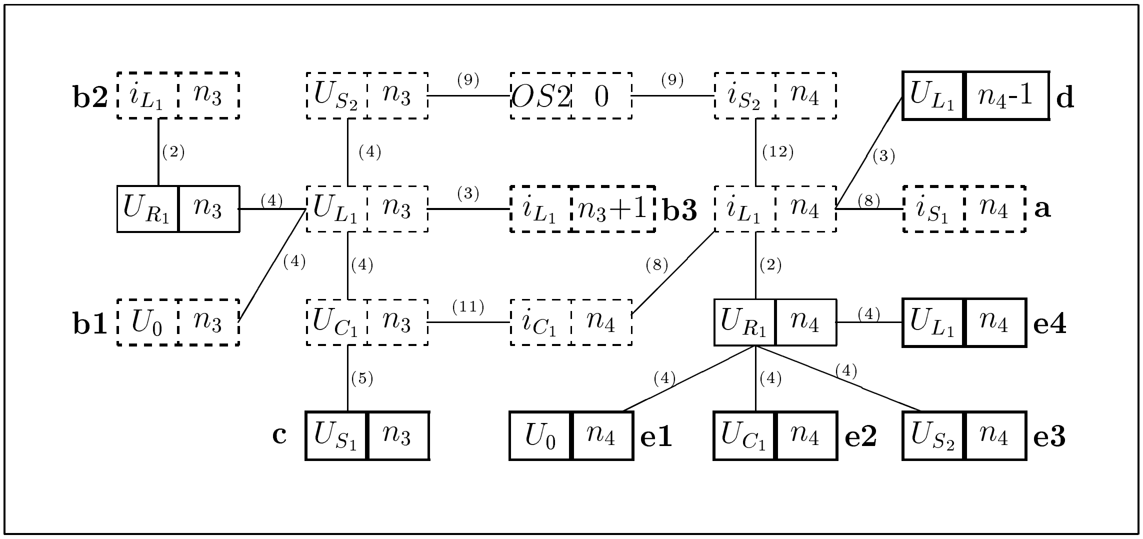


Figure 4.12: Example 2 MDG Possibility B Loop 2

Until now, we have only examined possibility A, let us examine possibility B as well to ensure that this possibility does not create further solutions.

4.3.3 Requirements for Both Loops in Possibility B

Fig. 4.12 shows the following requirements:

- **a and (b1 and b2 and b3)** must be **known**
- **c and d and (e1 or e2 or e3 or e4)** must be **unknown**
- $n_3 = n_4 + 1$ because of the capacitor equation

Fig. 4.13 shows the following requirements:

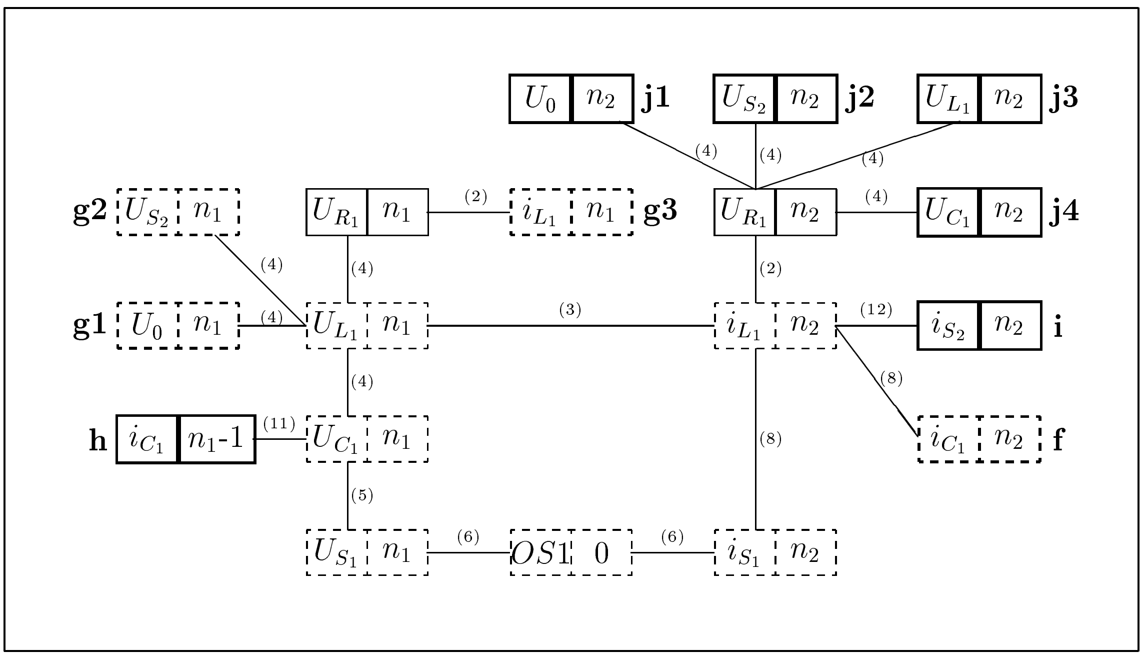


Figure 4.13: Example 2 MDG Possibility B Loop 1

- **f and (g1 and g2 and g3)** must be **known**
- **h and i and (j1 or j2 or j3 or j4)** must be **unknown**
- $n_1 + 1 = n_2$ because of the inductor equation

4.3.4 Conditions for Both Loops in Possibility B

Note that these graphs add a further stage of complexity to the restrictions, as the variables U_{L_1} and U_{C_1} are part of both loops and are contained in both sets of restricting leaves.

These restrictions cannot be fulfilled. For example, condition **d** is in contradiction with the capacitor constraint. $n_4 - 1 \geq n_3$ can be rewritten as: $n_4 \geq n_3 + 1$, yet we know that $n_3 = n_4 + 1$, which contradicts the above.

4.4 Conclusions

In the first part of this chapter, a systematic method for determining necessary conditions for the parameters n_1, n_2, n_3, and n_4 from the dependence graphs was introduced. The conditions resulted in no solutions for the example circuit. The verification part then showed that the method indeed results in the correct solution for a slightly modified circuit. The example circuit proves that there exist even fairly

Table 4.6: Conditions for Both Loops in Possibility B

restriction	type	condition
a	known	$n_4 \leq n_2$
b1 and **b2** and **b3**	known	**b1** : no condition as $U_0 = f(t)$, $[(n_3 \leq n_2) \vee (n_3 \leq n_4)]$ $\wedge[(n_3 + 1 \leq n_2) \vee (n_3 + 1 \leq n_4)]$
c	unknown	$n_3 \geq n_1$
d	unknown	$(n_4 - 1 \geq n_3) \wedge (n_4 - 1 \geq n_1)$
e1 or **e2** or **e3** or **e4**	unknown	**e1** : no condition as $U_0 = f(t)$, $[(n_4 \geq n_1) \wedge (n_4 \geq n_3)]$ $\vee(n_4 \geq n_3) \vee [(n_4 \geq n_1) \wedge (n_4 \geq n_3)]$
f_{11}	capacitor	$n_3 = n_4 + 1$
f	known	$n_2 \leq n_4$
g1 and **g2** and **g3**	known	**g1** : no condition as $U_0 = f(t)$, $(n_1 \leq n_3) \wedge [(n_1 \leq n_4) \vee (n_1 \leq n_2)]$
h	unknown	$n_1 - 1 \geq n_4$
i	unknown	$n_2 \geq n_4$
j1 or **j2** or **j3** or **j4**	unknown	**j1** : no condition as $U_0 = f(t)$, $\vee(n_2 \geq n_3) \vee [(n_2 \geq n_3) \wedge (n_2 \geq n_1)]$ $\vee[(n_2 \geq n_3) \wedge (n_2 \geq n_1)]$
f_3	inductor	$n_1 + 1 = n_2$

simple switching circuits that do not lend themselves to an automated index reduction approach using a modified Pantelides algorithm. Hence, it can be concluded that merely modifying the switch equations does not bring us any closer to the desired goal, the formulation of a *single model* of an ideal switching circuit involving conditional index changes that can be simulated in all switch positions.

Yet, the previous effort was not useless, because the work resulted in the idea for a new concept. In this chapter, we determined the conditions for the example circuit, but could not find a solution that would satisfy all restrictions. If we could relax some of the restrictions, maybe it would be possible to find a solution. We could provide the derivative variables of the capacitor and inductor equations using *implicit difference formulae*, which are in fact used in Differential Algebraic Equation Solvers. This would relax the capacitor and inductor constraints that caused many of the problems faced in this chapter.

The next chapter explores the possibility of using difference formulae to relax the set of necessary conditions that would give the switch equations the necessary freedom to assume both causalities independent of the environment in which they are used. It shall be shown that the use of *implicit difference formulae*, such as the *Backward Difference Formulae* that are widely used in commercial DAE solvers, makes the modifications of the switch equations unnecessary. The difference formulae used to substitute the original derivatives in the inductor and capacitor equations by themselves create the necessary loops that free up the former restrictions on the causality assignments for the switch equations.

CHAPTER 5

The Concept of Using Difference Formulae

5.1 Difference Formulae

The previous chapter showed that no general solution exists for the equation modification problem. Sets of necessary conditions were derived by comparing needed orders of differentiated variables that should be **unknown** with loop–forming variables. During this work, the idea was created that it might be possible to relax some of the restrictions through the use of *implicit difference formulae.* [4]

$$x = h \cdot \dot{x} + \mathrm{old}(x) \tag{5.1}$$

$$\dot{x} = \frac{x - \mathrm{old}(x)}{h} \tag{5.2}$$

Equ. 5.1 describes the structure of the discretization for a large class of implicit integration algorithms. The known scalar h depends on the step size and on method–specific constants, whereas *old(x)* is a function of known values of x at previous time instants. The *Backward Difference Formulae (BDF)* of different orders described by

Equ. 5.1 are widely used to solve stiff systems. The first order BDF algorithm is also known under the name *backward Euler* method. It is used throughout this chapter to keep the concept as simple as possible. To improve the accuracy, higher order BDF algorithms could be used instead. The concept remains the same.

Equ. 5.3 and Equ. 5.4 describe the simplified formulae used by the backward Euler algorithm. Here, h represents the step size directly.

Of course by using implicit difference formulae, we sacrifice the separation between the model equations and the numerical solver equations in order to achieve the modeling of a conditional index system using a single model.

$$x = h \cdot \dot{x} + xold \tag{5.3}$$

$$\dot{x} = \frac{x - xold}{h} \tag{5.4}$$

5.2 Using Difference Formulae in Inductor and Capacitor Equations

Let us apply the difference formulae to replace the first order derivatives that show up in the capacitor and inductor equations. By using the difference formulae, we relax the constraint equations that had previously been imposed by the capacitor and inductor equations. In the dependence graphs, the first order derivatives of the state variables get replaced by the state variables themselves, and there is no longer a need for a difference between the orders of n_1 and n_3 on the one hand, and between n_2 and n_4 on the other.

The set of equations for the modified example circuit making use of the backward Euler formulae directly, Equ. 5.4, contains the step size h explicitly.

5.2.1 The Equation System

$$U_0 - u_0(t) = f_1 = 0$$

$$U_{R_1} - R_1 \cdot i_{L_1} = f_2 = 0$$

$$-U_{L_1} + L_1 \cdot \frac{i_{L_1} - i_{L_1 old}}{h} = f_3 = 0$$

$$U_0 - U_{R_1} - U_1 - U_{L_1} = f_4 = 0$$

$$U_1 - U_{S_1} = f_5 = 0$$

$$OS1 \cdot i_{S_1} + (1 - OS1) \cdot U_{S_1} = f_6 = 0$$

$$\text{if } [\text{not}(U_{S_1} > 0) \text{ and not } (i_{S_1} > 0)] \text{ then 1 else } 0 - New(OS1) = f_7 = 0$$

$$i_{L_1} - i_{S_1} - i_{C_1} = f_8 = 0$$

$$OS2 \cdot i_{S_2} + (1 - OS2) \cdot U_{S_2} = f_9 = 0$$

$$\text{if } [\text{not}(U_{S_2} > 0) \text{ and not } (i_{S_2} > 0)] \text{ then 1 else } 0 - New(OS2) = f_{10} = 0$$

$$-i_{C_1} + C_1 \cdot \frac{U_{C_1} - U_{C_1 old}}{h} = f_{11} = 0$$

$$i_{C_1} - i_{S_2} = f_{12} = 0$$

$$U_1 - U_{S_2} - U_{C_1} = f_{13} = 0$$

5.2.2 Dependence Graph Considerations

The dependence graph for this set of equations is shown in Fig. 5.1. The former state derivatives have been merged with the state variables themselves, which now show a dependence on old values of themselves as well as the simulation time step h. The old values of U_{C_1} and i_{L_1} are known from one or more previous simulation steps. The use of the difference formulae creates two new branches in the dependence graph that can form parts of algebraic loops containing the switch equations. The dependence graph can be interpreted in two different ways.

First, it can be viewed as consisting of two algebraic loops that are connected at U_1 and i_{C_1}. The two separate loop structures are shown in Fig.5.2 and in Fig. 5.3. The second loop consists of the four equations f_9, f_{11}, f_{12}, and f_{13} in the five unknowns U_{S_2}, i_{S_2}, i_{C_1}, U_{C_1}, and U_1. Similarly, the first loop consists of the six equations f_2, f_3, f_4, f_5, f_6, and f_8 in the seven unknowns U_{S_1}, i_{S_1}, i_{C_1}, i_{L_1}, U_{L_1}, U_{R_1}, and U_1. Note that U_0 is considered **known** from the time dependence, and U_{C_1old} and i_{L_1old} are **known** from earlier simulation steps.

The two dependence graphs are interconnected at U_1 and at I_{C_1} in such a way that

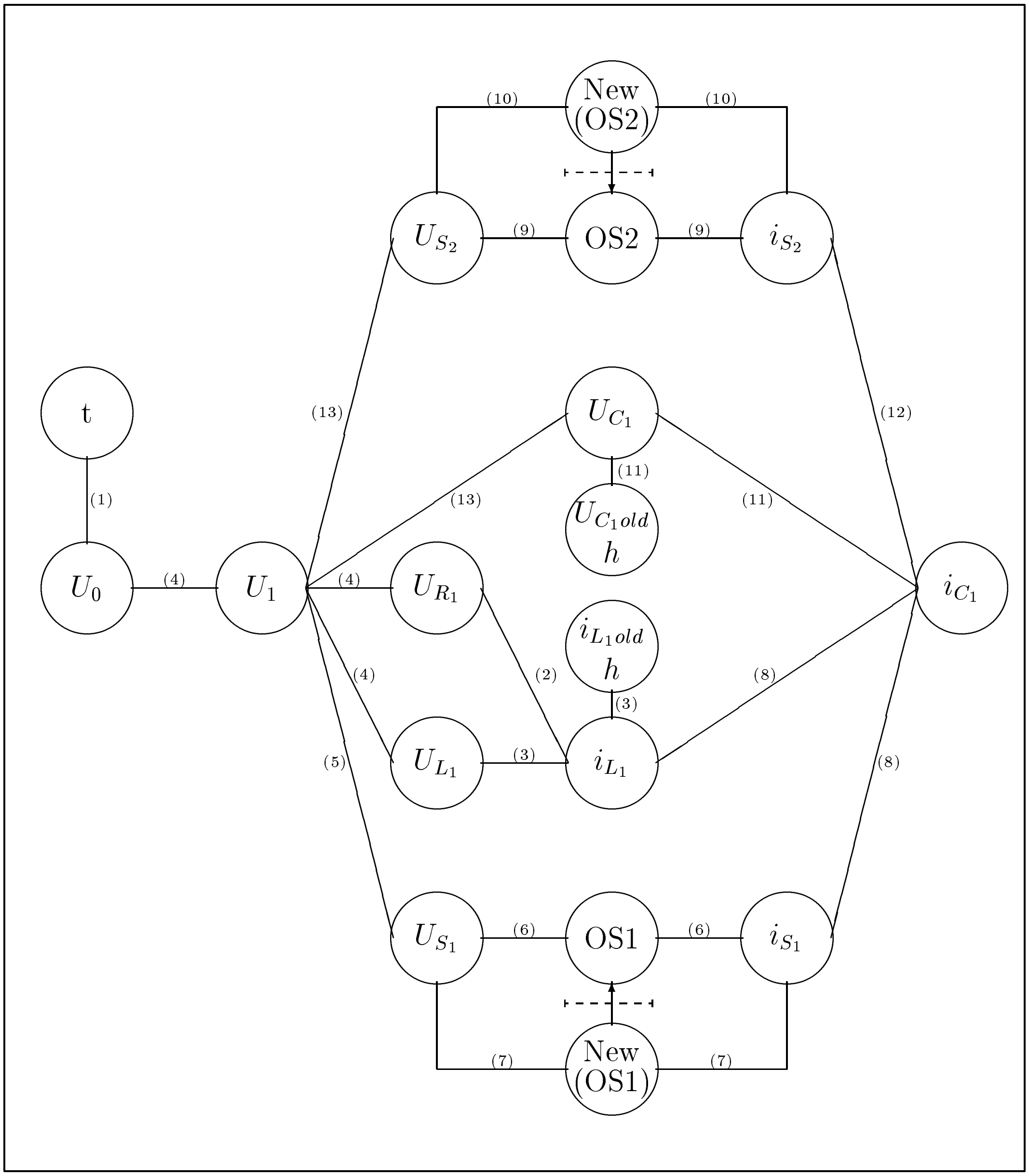


Figure 5.1: Dependence Graph for Example 1 with BDF

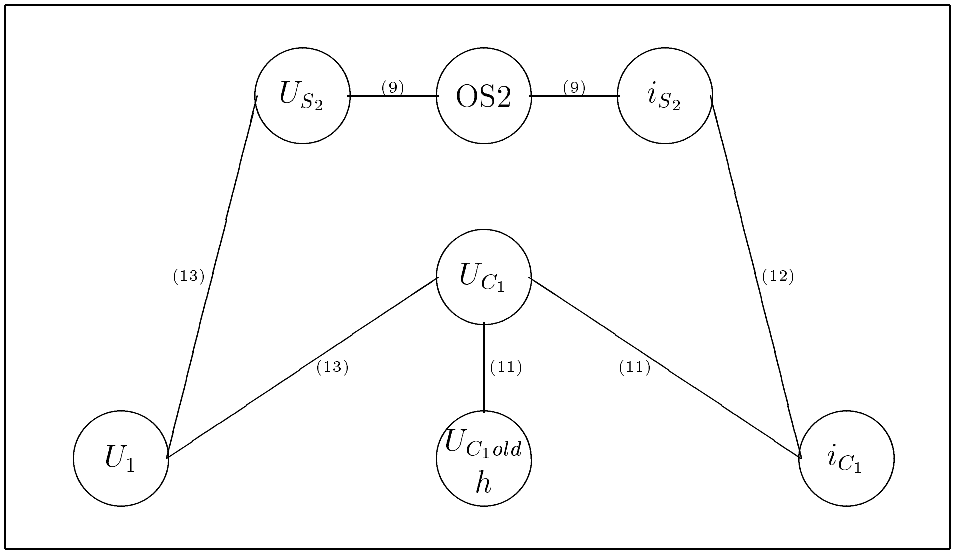


Figure 5.2: Loop 2 for Example 1 with BDF

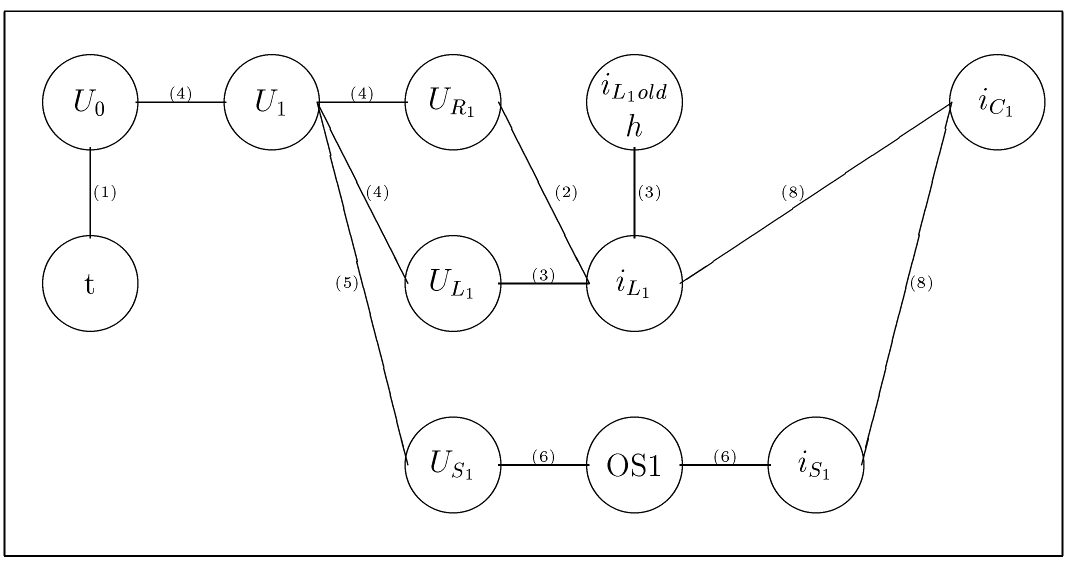


Figure 5.3: Loop 1 for Example 1 with BDF

neither of the loops can be solved separately. The two algebraic structures are represented by the two structure incidence matrices:

$$S_1 = \begin{array}{c} \\ f_9 \\ f_{12} \\ f_{11} \\ f_{13} \end{array} \begin{array}{c} \begin{array}{ccccc} U_{S_2} & i_{S_2} & U_{C_1} & i_{C_1} & U_1 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array} \tag{5.5}$$

$$S_2 = \begin{array}{c} \\ f_6 \\ f_8 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{array} \begin{array}{c} \begin{array}{ccccccc} i_{C_1} & U_1 & U_{S_1} & i_{S_1} & U_{L_1} & i_{L_1} & U_{R_1} \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \tag{5.6}$$

Second, the dependence graph can be viewed as a single algebraic structure consisting of the ten equations f_2, f_3, f_4, f_5, f_6, f_8, f_9, f_{11}, f_{12}, and f_{13} in the ten unknowns U_{S_1}, i_{S_1}, U_{S_2}, i_{S_2}, U_{C_1}, i_{C_1}, U_{L_1}, i_{L_1}, U_{R_1}, and U_1. The complete algebraic structure can be represented by the complete structure incidence matrix:

$$S_C = \begin{array}{c} \\ f_9 \\ f_{12} \\ f_{11} \\ f_{13} \\ f_6 \\ f_8 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{array} \begin{array}{c} \begin{array}{cccccccccc} U_{S_2} & i_{S_2} & U_{C_1} & i_{C_1} & U_1 & U_{S_1} & i_{S_1} & U_{L_1} & i_{L_1} & U_{R_1} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \tag{5.7}$$

that shows that two interconnected algebraic structures are present. The interconnection structure is such that knowledge of one of the switch variables does not break the algebraic structure for the other, i.e., either of the switch variables $OS1$ or $OS2$ can be given, and yet, there still remains an algebraic loop embedding the other switch equation.

5.2.3 Determinant and Singular Step Sizes

The equation set written in matrix form contains now the parameter h explicitly. The parameter h represents the step size directly in the case of the backward Euler algorithm. In the case of higher order BDF techniques, it is a constant times the step size. As a result, the determinant of the matrix A depends on the parameter h, and thus on the step size. Let us examine the value of the determinant to ensure that it is unequal to zero for the step size used.

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -R_1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & \frac{L_1}{h} \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ (1-OS1) & 0 & 0 & 0 & 0 & 0 & OS1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 \\ 0 & (1-OS2) & 0 & 0 & 0 & 0 & 0 & OS2 & 0 & 0 \\ 0 & 0 & \frac{C_1}{h} & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}}_{A} \cdot \begin{pmatrix} U_{S_1} \\ U_{S_2} \\ U_{C_1} \\ U_{L_1} \\ U_{R_1} \\ U_1 \\ i_{S_1} \\ i_{S_2} \\ i_{C_1} \\ i_{L_1} \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{L_1 \cdot i_{L_1 old}}{h} \\ U_0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{C_1 \cdot U_{C_1 old}}{h} \\ 0 \\ 0 \end{pmatrix}$$

where

$$\det A = \begin{vmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -R_1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & \frac{L_1}{h} \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ (1-OS1) & 0 & 0 & 0 & 0 & 0 & OS1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 \\ 0 & (1-OS2) & 0 & 0 & 0 & 0 & 0 & OS2 & 0 & 0 \\ 0 & 0 & \frac{C_1}{h} & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{vmatrix}$$

$$\begin{aligned} &= (1-OS1)\cdot(1-OS2)\cdot\underbrace{(-\frac{L_1}{h}-R_1)}_{k_1} \\ &\quad + (1-OS1)\cdot OS2\cdot\underbrace{(\frac{R_1C_1}{h}+\frac{L_1C_1}{h^2})}_{k_2} \\ &\quad + OS1\cdot(1-OS2)\cdot\underbrace{(1+\frac{R_1C_1}{h}+\frac{L_1C_1}{h^2})}_{k_3} \end{aligned}$$

$$+ OS1 \cdot OS2 \cdot \underbrace{\frac{-C_1}{h}}_{k_4}$$

The determinant equals one of the four values k_1, k_2, k_3, or k_4 depending on the values of the two discrete switch variables. Setting one of these four expressions equal to zero leads to one of four equations. Each of these equations, when set equal to zero, can be used to identify a value of the parameter h that causes a singularity. In our special case, the parameter h is equivalent to the step size.

$$(-\frac{L_1}{h} - R_1) = 0 \tag{5.8}$$

$$(\frac{R_1C_1}{h} + \frac{L_1C_1}{h^2}) = 0 \tag{5.9}$$

$$(1 + \frac{R_1C_1}{h} + \frac{L_1C_1}{h^2}) = 0 \tag{5.10}$$

$$\frac{-C_1}{h} = 0 \tag{5.11}$$

The first two equations result in the singular value $h_1 = -\frac{L_1}{R_1}$, the third equation is quadratic in h and leads to the two singular values $h_{2/3} = -\frac{C_1L_1}{2} \cdot (1 \pm \sqrt{1 - (2\frac{L_1}{R_1})^2})$, and the fourth equation has no solution. All singular values are negative, and thus we need not worry about the step size for the case of the example circuit.

However for a general model and a higher order difference formulae, the corresponding values of the determinant should always be examined for singular values. The dependence between the parameter h and the step size should be used to calculate the singular step sizes accordingly.

5.2.4 Using the Equation System in a Simulation

In a simulation, the equation system must be solved at every simulation step. There are two possible ways to invert the matrix of the equation system for every simulation step.

The first way is to invert the system matrix with a symbolic formulae manipulation program, such as Dymola [6] or Mathematica [8]. The result will be an analytic scheme of equations resulting in values of all variables at each simulation time point.

The second way is to invert the system matrix numerically, after all values for the current simulation time have been plugged in. This method requires only a numerical matrix solver instead of a symbolic one. This method requires more execution time during a simulation run by saving compilation time prior to executing the simulation run.

A numerical method should use the slightly modified equation system:

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -R_1 \\ 0 & 0 & 0 & -h & 0 & 0 & 0 & 0 & 0 & L_1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ (1-OS1) & 0 & 0 & 0 & 0 & 0 & OS1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 \\ 0 & (1-OS2) & 0 & 0 & 0 & 0 & 0 & OS2 & 0 & 0 \\ 0 & 0 & C_1 & 0 & 0 & 0 & 0 & 0 & -h & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}}_{A_{mod}} \cdot \begin{pmatrix} U_{S_1} \\ U_{S_2} \\ U_{C_1} \\ U_{L_1} \\ U_{R_1} \\ U_1 \\ i_{S_1} \\ i_{S_2} \\ i_{C_1} \\ i_{L_1} \end{pmatrix} = \begin{pmatrix} 0 \\ L_1 \cdot i_{L_1 old} \\ U_0 \\ 0 \\ 0 \\ 0 \\ 0 \\ C_1 \cdot U_{C_1 old} \\ 0 \\ 0 \end{pmatrix}$$

This equation system differs from the previous one only in that the two rows that contain the difference formulae were modified by multiplying the equations with the step size h. The step size h is usually a small number that is converging to zero during event iterations. The modified system matrix contains, in addition to the small step size h, the parameters L_1, C_1, and R_1, the discrete switch variables $OS1$ and $OS2$, as well as 0 and 1 elements. This is numerically far better than having to deal with very big numbers that are caused by dividing through h on the left and right side of the equations. Numerical problems, such as currents that toggle between small positive and negative values instead of assuming the analytically correct zero value, vanish with this alternative formulation of the problem.

The reader may wonder why the system matrix has to be inverted in a simulation run at every simulation step, as the matrix only depends on fixed parameters, the switch variables, and the step size. Two reasons can be mentioned that may call for a changing step size. First, the detection and exact calculation of an event may temporarily reduce the step size, and second, accuracy requirements can force a decrease of the step size. The switch variables can change with each detected and calculated event, each time leading to a different numerical matrix.

The necessary steps in a simulation will be discussed in more detail in a later section.

5.3 Elimination of Restrictions on Bond Graph Causalities for Inductor and Capacitor Elements

The use of backward difference formulae eliminates the causality requirements for the inductor and the capacitor. $e_L = L_1 \cdot \frac{f_L - f_{Lold}}{h}$, the discretized inductor equation, can be solved for either the flow variable f_L or the effort variable e_L. Thus both causalities are meaningful for the new discretized inductor element. The two different causalities for the modified inductor element are shown in Fig. 5.4 (a) and (b).

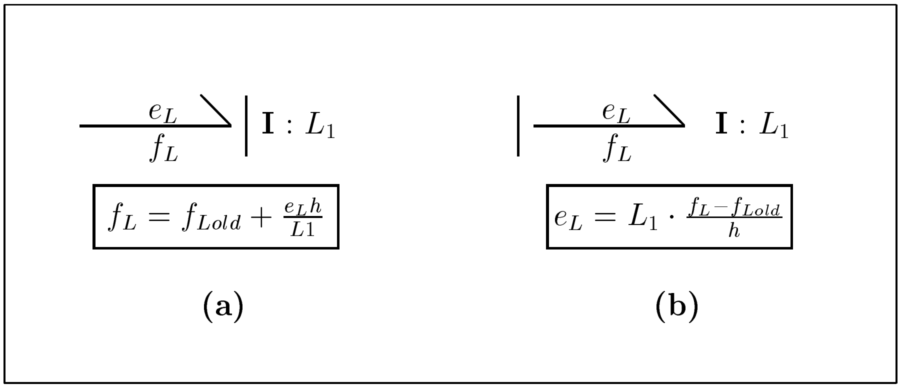


Figure 5.4: Inductor Causalities with BDF

The same considerations apply to the capacitor element. $f_C = C_1 \frac{e_C - e_{Cold}}{h}$, the essential discretized capacitor equation, can be solved for either the flow variable f_C or the effort variable e_C. Thus both causalities are meaningful for the new discretized capacitor element. The two different causalities for the modified capacitor element are shown in Fig. 5.5 (a) and (b).

The use of the difference formulae eliminates the causality requirement for inductor and capacitor elements by eliminating the need to compute the derivatives. The inductor and capacitor elements have now properties similar to the resistor element. Note that there is no need anymore for detecting higher index problems, as they will be solved automatically using this concept. In this description, no difference exists any longer between determining the relationship between current and voltage in a

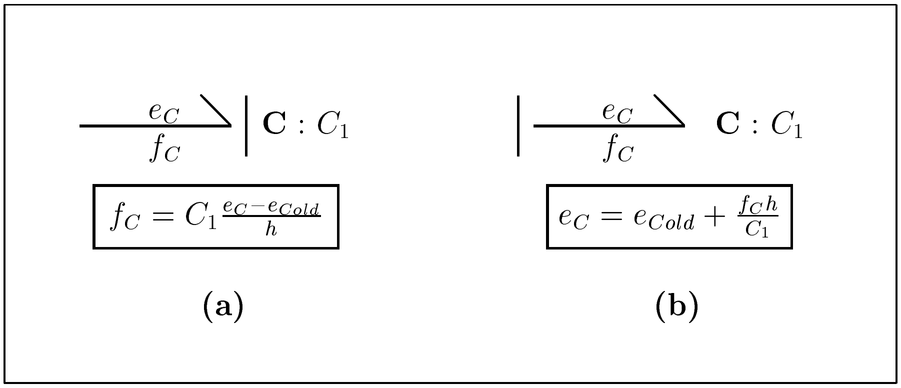


Figure 5.5: Capacitor Causalities with BDF

resistor, an inductor, or a capacitor. All these relationships are algebraic in current and voltage.

5.3.1 New Possibilities for Assigning Bond Graph Causalities for the Example Circuit

The relaxation of the causality requirements for capacitor and inductor elements makes all four possible causality assignments for the two switch elements feasible in a conflict–free manner. Fig. 5.6, Fig. 5.7, Fig. 3.3, and Fig. 3.4 show the four basic possibilities. These four possibilities correspond to the four cases: $(OS1 = 0, OS2 = 0)$, $(OS1 = 1, OS2 = 1)$, $(OS1 = 0, OS2 = 1)$, and $(OS1 = 1, OS2 = 0)$. Therefore, we have at least one conflict–free causality assignment for each of the four switch positions. In reality, there are even more possibilities in assigning causality strokes. In

Fig. 5.6, Fig. 3.3, and Fig. 3.4, the causality assignment for the corresponding case is not even unique, i.e., even if both switch positions are fixed, there still remains an algebraic loop in the set of equations. This observation can be explained by the new behavior of inductor and capacitor elements. Using the difference formulae method, inductor and capacitor elements exhibit essentially the same behavior as a resistors. Thus the example circuit, in which an inductor is placed in series with a resistor, displays the same behavior as a circuit with two resistors placed in series that could be replaced by a single resistor. In such circuits, algebraic couplings between resistors are present, and these algebraic structures result in free choices for assigning causality strokes.

5.3.2 Remaining Causality Requirements

With the new concept, inductor and capacitor elements have no fixed causality requirements, but we still are left with the causality requirements for both types of junctions and sources. These requirements can still be the cause of singular determinants of the resulting equation system. However, such cases are not physically meaningful. For example, two parallel switch elements cannot assume independent causalities as a result of the requirement at a 0–junction. In this example, it would be impossible to calculate the current distribution between the two switches if both

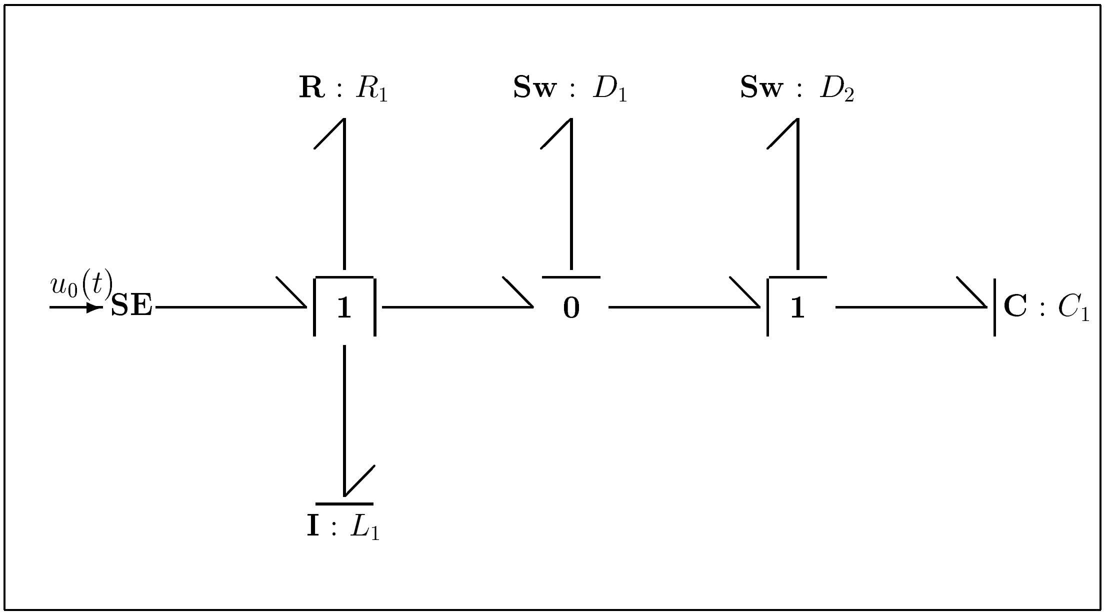


Figure 5.6: Bond Graph Causality (c) for Example 1

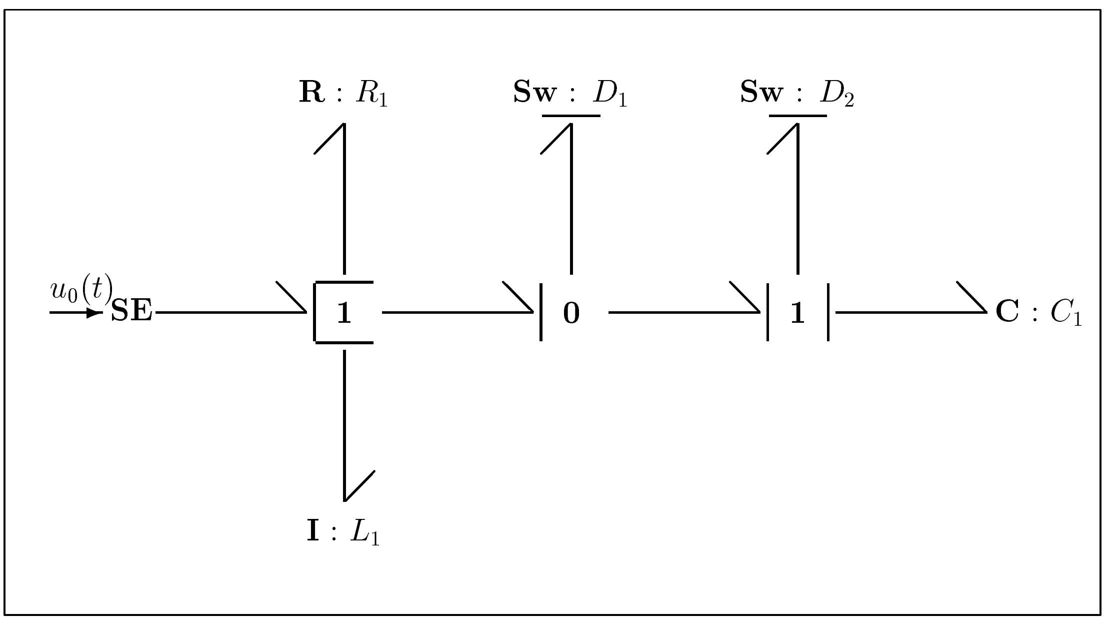


Figure 5.7: Bond Graph Causality (d) for Example 1

switches were closed simultaneously. Similarly, two switches in series cannot assume independent causalities because of the requirement at a 1–junction. This time, if both switches were simultaneously open, we could no longer compute the potential of the node between them, because it would be floating relative to the rest of the circuit. For similar reasons, switches cannot be placed in series with current sources, or in parallel with voltage sources.

The next chapter, showing the example of an SCR circuit for train speed control, will explain these remaining problems.

5.4 Necessary Simulation Steps

A DAE system simulation using the concept of difference formulae consists of a loop in which the new variables are calculated from old variable values, previous time values, and the current time. The DAE system is converted to a purely algebraic equation system by the use of the difference formulae. This equation system, if it is linear, can be described by:

$$A(p, \frac{1}{h}, t) \cdot x = b(x_{old}, p, \frac{1}{h}, t) \tag{5.12}$$

which is the algebraic structure that we have encountered several times already. p is a vector containing model parameters, x is the vector of simulation variables that are of concern, x_{old} are previous values of the variable vector, t denotes time, and h

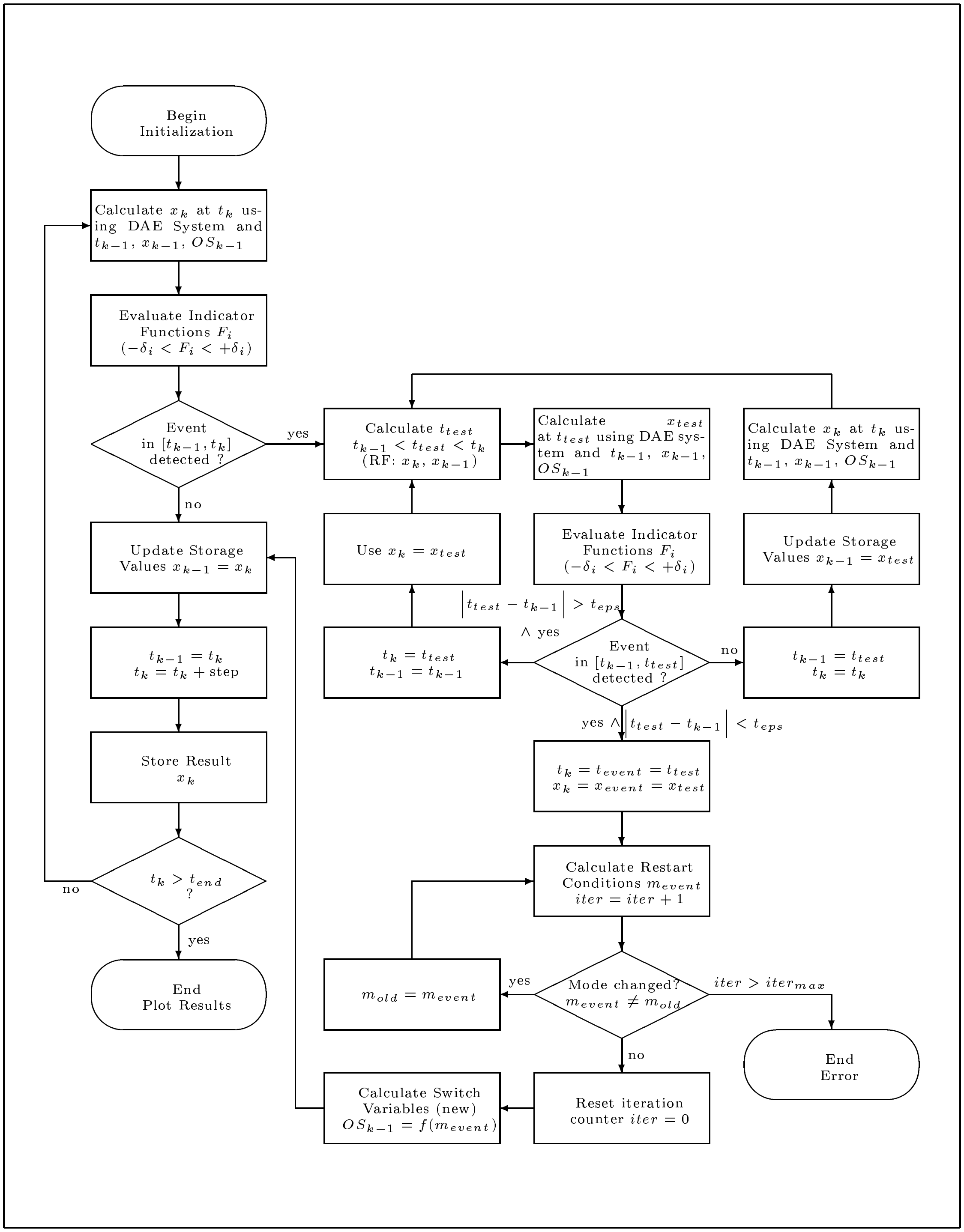


Figure 5.8: Simulation Flow Diagram

depends on the step size. The values for x_{old} are either provided by previous steps or initial conditions.

$$A(OS, p, \frac{1}{h}, t) \cdot x = b(x_{old}, p, \frac{1}{h}, t) \tag{5.13}$$

In the case of variable structure models that are described by switch equations, the matrix depends additionally on the vector OS that contains the discrete switch variables. The central element of a linear DAE simulation of a variable structure model is the inversion of the matrix A. This matrix can be inverted either symbolically or numerically. As stated in a previous section, the numerical inversion is better conditioned in the alternative form:

$$A(OS, p, h, t) \cdot x = b(x_{old}, p, h, t) \tag{5.14}$$

Additional elements needed by the simulation of a variable structure model are:

- Event Detection
- Event Calculation
- Evaluating Results of Events

These steps are shown in more detail in Fig. 5.8 and are further explained in the following subsections.

5.4.1 Event Detection

Event handling is based on indicator functions that indicate the event time. Indicator functions depend on both the variable vector and the simulation time, and describe the event time indirectly by means of a zero crossing of the indicator function. The event conditions can be further specialized by only detecting positive to negative or negative to positive crossings. For example, the simulation language ACSL [7] allows the code to specify these types of zero crossings.

In our example circuit the indicator functions are described by equations f_7 and f_{10}. These equations that specify the diode characteristics are:

$$New(OS2) = \text{if } [\text{not}(U_{S_1} > 0) \text{ and not } (i_{S_1} > 0)] \text{ then } 1 \text{ else } 0$$

$$New(OS2) = \text{if } [\text{not}(U_{S_2} > 0) \text{ and not } (i_{S_2} > 0)] \text{ then } 1 \text{ else } 0$$

Thus for the example circuit, we have the following four indicator functions:

$$\begin{aligned} F_1 &= U_{S_1} \\ F_2 &= i_{S_1} \\ F_3 &= U_{S_2} \\ F_4 &= i_{S_2} \end{aligned}$$

Usually an event is detected whenever one or more of these indicator functions crosses through zero. However a *delta*– vicinity around zero is used to avoid problems that would otherwise occur if the function were to assume a value of zero at a particular point in time, then stay at that value for some time, and only then assume a value different from zero again, either with the same or with the opposite sign from before the zero crossing. δ is usually a very small constant in the range of the machine resolution ϵ.

For these reasons, it is common to use eight indicator functions instead of the previously proposed four to avoid problems with such special functions:

$$\begin{aligned}
F_1 &= U_{S_1} - \delta \\
F_2 &= U_{S_1} + \delta \\
F_3 &= i_{S_1} - \delta \\
F_4 &= i_{S_1} + \delta \\
F_5 &= U_{S_2} - \delta \\
F_6 &= U_{S_2} + \delta \\
F_7 &= i_{S_2} - \delta \\
F_8 &= i_{S_2} + \delta
\end{aligned}$$

Whenever one of these eight functions crosses through zero, the event calculation is triggered. It consists of an iterative loop that adjusts the step size in order to hit the event accurately, i.e., force the value of the triggering indicator function to zero. After the event calculation, the *mode* of the indicator function is determined :

$$\text{mode}_i = \begin{cases} 1 & \forall \quad F_i > +\delta \\ 0 & \forall \quad -\delta < F_i < +\delta \\ -1 & \forall \quad F_i < -\delta \end{cases} \tag{5.15}$$

The problems associated with constant zero values vanish with the discretization into a positive area, a zero area, and a negative area. For more details see [3].

A mode evaluation takes place *after* the event calculation. It may happen that a changing mode triggers another mode change, and thus, this process is iterative, until either a consistent mode configuration is found or an iteration counter stops the simulation.

5.4.2 Event Calculation

The event calculation is started after an event has been detected. Commonly, the *Regula Falsi* algorithm is used to calculate event times. However, our diode characteristics are described by functions that are not truly zero–crossing functions. The diode current is always either greater or equal to zero, whereas the diode voltage is always either less or equal to zero. The aforementioned algorithm exhibits poor convergence behavior in the case of such indicator functions. Therefore, a *bi–section* algorithm was used instead. Thereby, the arithmetic mean of the beginning of the search time interval t_{k-1} and the end t_k is used as the next evaluation time t_{test}. If an event detection occurs between t_{k-1} and t_{test}, the right border of the search interval is updated to t_{test}, otherwise the left border of the search interval is updated to t_{test}. The iteration ends when $|t_{test} - t_{k-1}| < t_{eps}$, where t_{eps} is a constant that determines the accuracy of the event calculation. The event calculation algorithm is shown in Fig. 5.8.

5.4.3 Evaluating Results of Events

The new *mode* values are evaluated after the event time has been determined. After the previously described mode iteration has converged, the switch variables are updated using the modes of the indicator functions. Note that the diode characteristics can only be simulated using the δ–vicinity concept.

5.5 Results for the Example Circuit

The example circuit was simulated as a Fortran executable. Fortran was chosen, because the circuit was modeled using Dymola, and Dymola was instructed to generate code for the simulation language ACSL. However, ACSL was developed to simulate ODE problems, and an ACSL simulation without state variables did not work. As ACSL is Fortran–based, the code was then manually modified to be used by a DAE solver developed as part of the project.

The example circuit has no real use. It had been chosen because of its structural properties, not because of the physical system that it represents. The source voltage is sinusoidal with a frequency of 50 Hz and an amplitude of 10 V. If the capacitor is initially positively charged, and the two diodes are assumed to be non–conducting at the beginning of the simulation, the capacitor keeps the charge and the diode D_2 remains non–conducting during the entire simulation run. However, if the capacitor is initially negatively charged, the diode D_2 switches at once to conducting mode, and the capacitor discharges itself immediately by means of the two diodes D_1 and D_2. Thereafter, it never gets an opportunity to recharge itself. The circuit basically behaves in the same way as the inductive load circuit shown in the introduction, because the diode D_2 remains in its non–conducting mode almost throughout the entire simulation.

The example circuit was simulated with initial conditions $U_{C_{10}} = -8\ V$, $i_{L_{10}} = 0\ A$, $OS1_0 = 1$, and $OS2_0 = 1$ (both diodes are initially in their non–conducting mode). Fig. 5.9 shows the results for U_{L_1}, U_{R_1}, and i_{L_1}. These variables show the same behavior as an inductive load circuit mentioned in the introduction, but this time with a real inductor with a resistance in series.

Fig. 5.10 shows the results for U_{C_1} and i_{C_1}. The initial discharging of the capacitor can only be seen in an extreme zoom. Ideally the initial current should be infinite. However in the simulation, the current is restricted by the step size h, as the discharging time cannot be shorter than one step.

Fig. 5.11 shows the results for $OS1$ and $OS2$. The second diode D_2 becomes conducting for just a short instant at the beginning of the simulation, i.e., while the capacitor is being discharged. The first diode D_1 toggles between its two modes with the frequency of the source voltage.

Fig. 5.12 shows the results for U_{S_1} and U_{S_2}. The voltage across the first diode equals that of the second diode, as the second diode is non–conducting for all times after the first time instant, and the capacitor remains discharged after the same time instant. Note the small peaks of U_{S_1} in the plot at the switch times. These are caused by the choice of the accuracy of the event calculation. The event calculation always results in a time t_{test} that is essentially too large. This behavior is intended to prevent an endless loop detecting the same event forever, always resulting in a time earlier than the real event time. The inaccuracy in calculating the exact event time causes a small positive voltage across the diode that does not correspond to true diode behavior. The small positive voltage vanishes already in the next step when the changed switch position results in a zero voltage.

Finally Fig. 5.13 shows the results for i_{S_1} and i_{S_2}. These curves are only given for completeness, as the inductor current is identical to the diode current through the first switch after the first time instant, and the current through the second switch is always identical to the capacitor current.

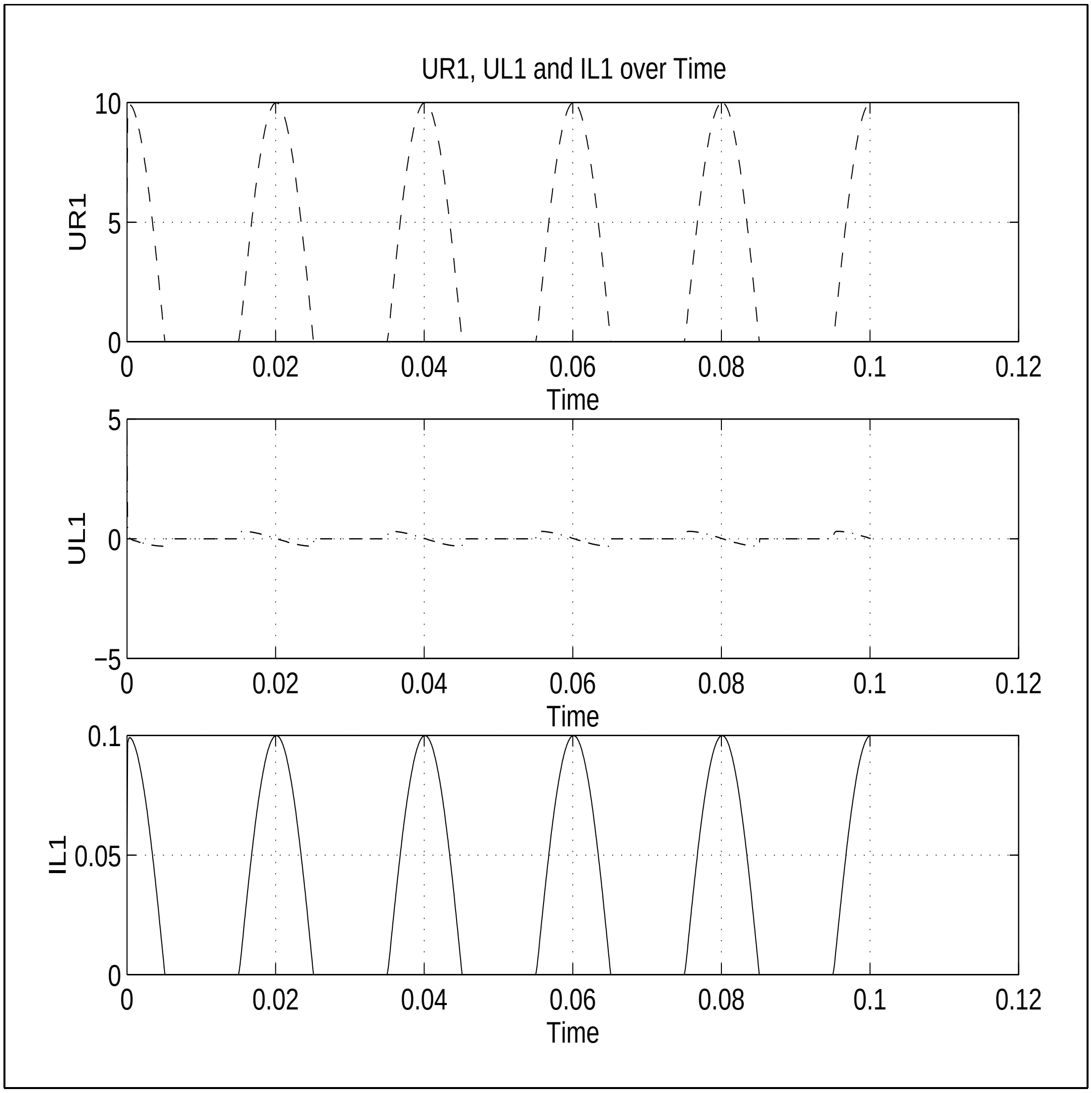


Figure 5.9: Simulation Results Example (1)

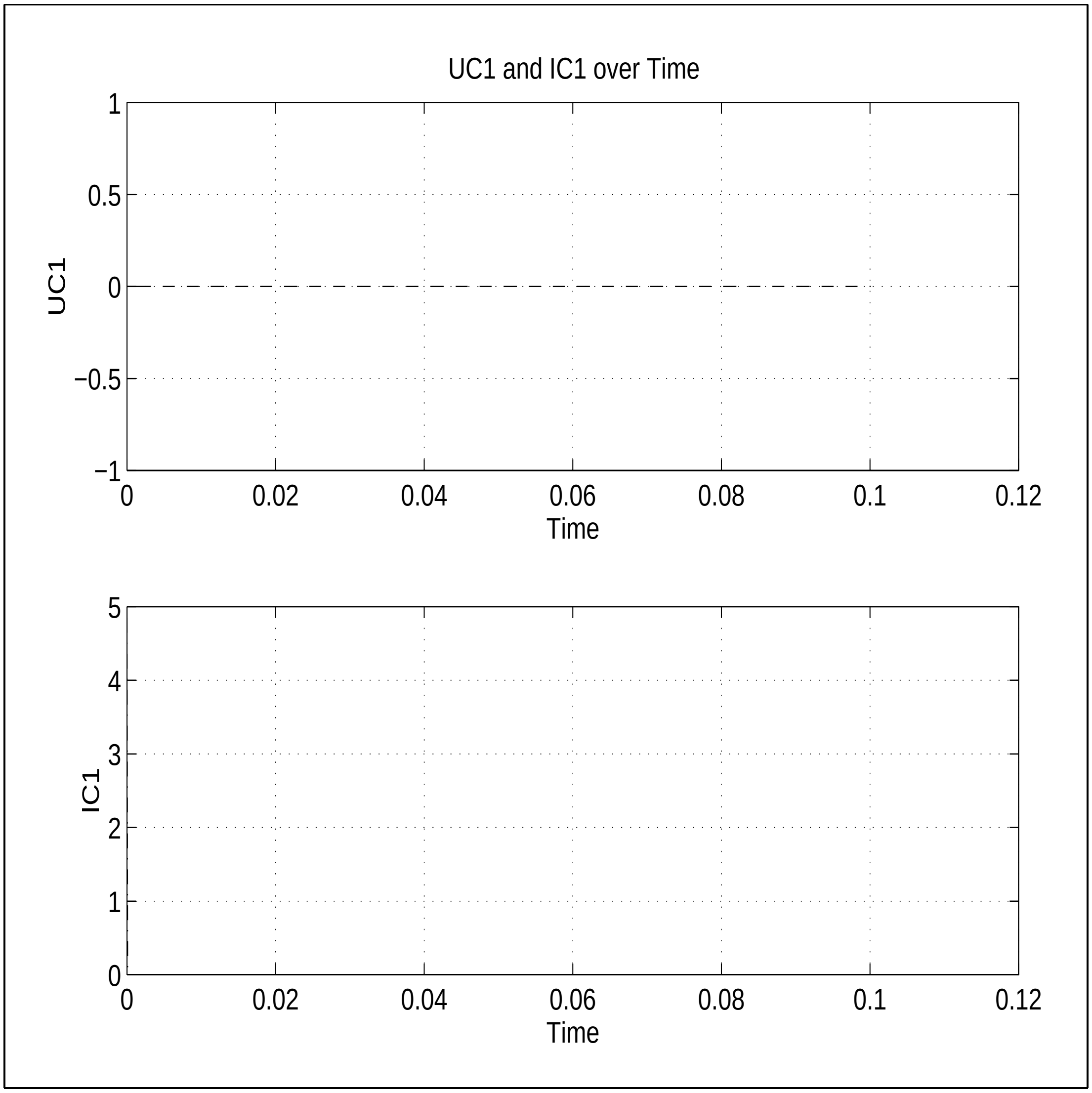


Figure 5.10: Simulation Results Example (2)

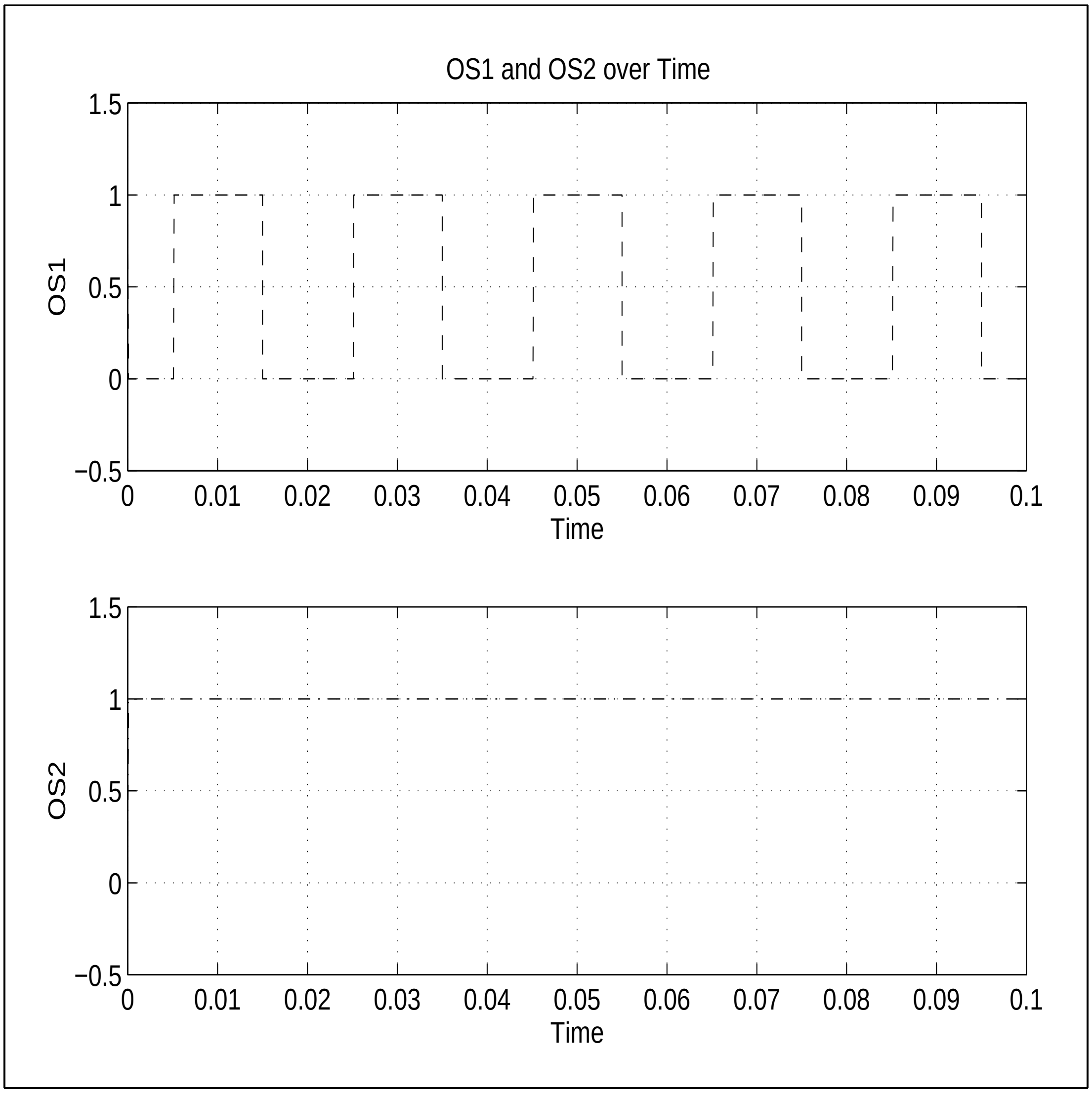


Figure 5.11: Simulation Results Example (3)

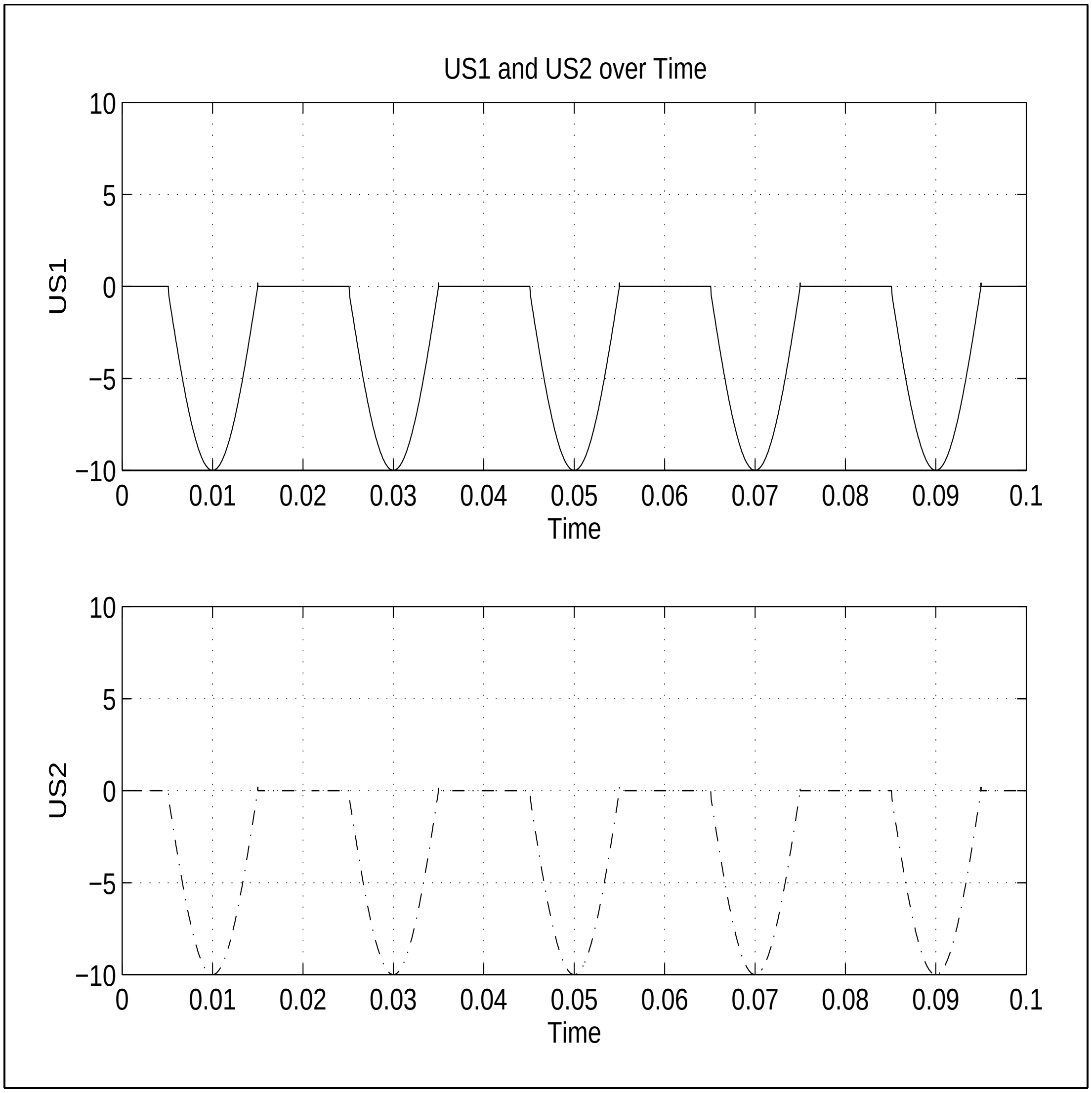


Figure 5.12: Simulation Results Example (4)

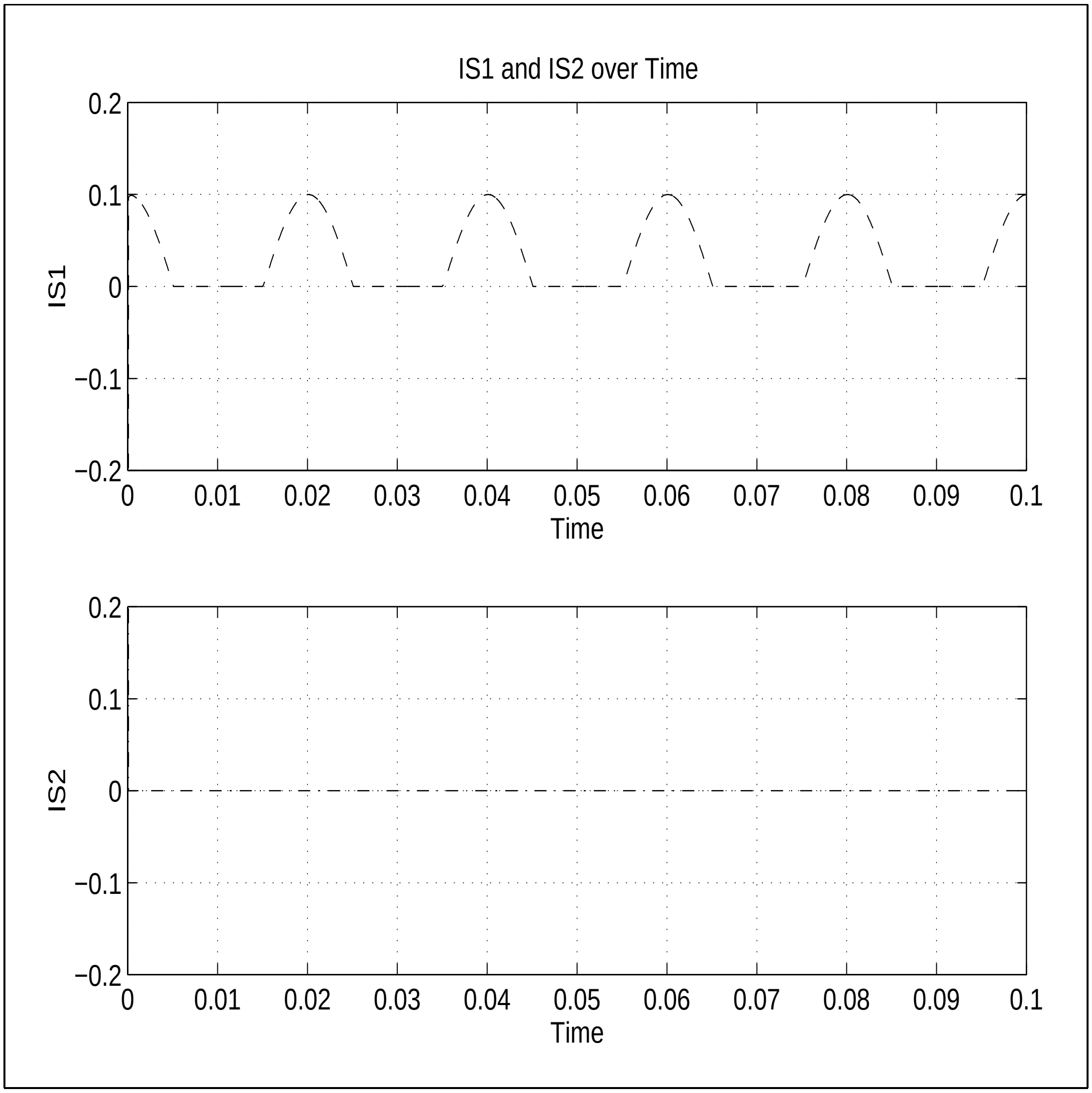


Figure 5.13: Simulation Results Example (5)

CHAPTER 6

The SCR Circuit for Train Speed Control

6.1 Ordinary Two Quadrant Rectifier

The SCR circuit contains a two quadrant rectifier and a thyristor diode pair that achieves the desired behavior. Let us first take a look at the problems caused by a two quadrant rectifier before we discuss the SCR circuit in detail. The circuit shown in Fig. 6.1 is described by the following set of equations:

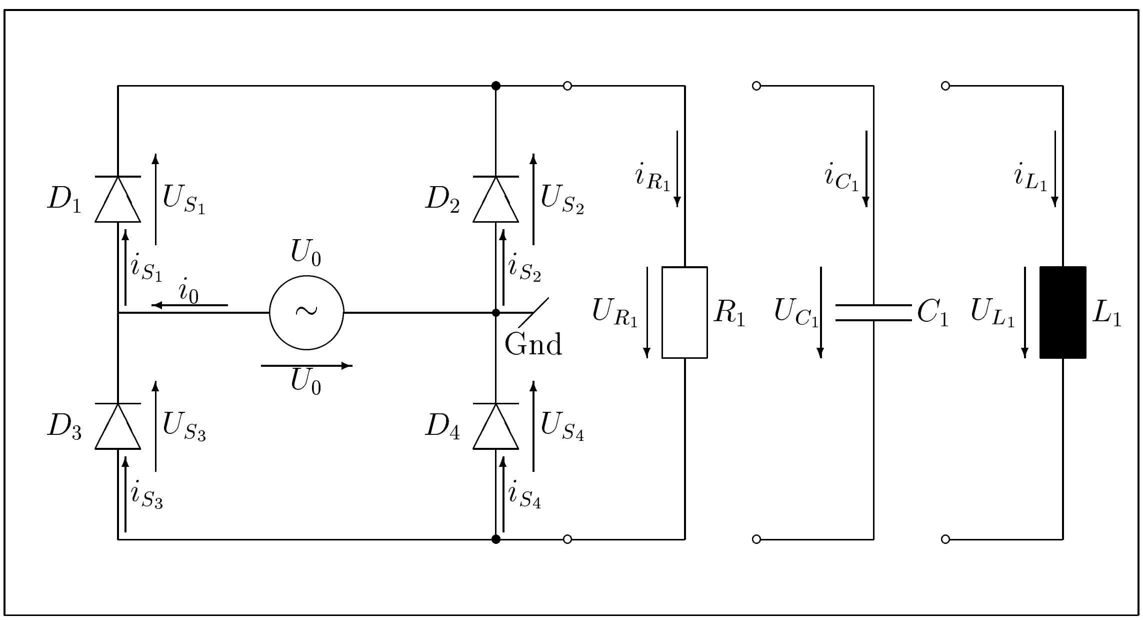


Figure 6.1: Two Quadrant Rectifier Circuit

$$i_{S_1} - i_{S_3} - i_0 = f_1 = 0$$

$$-i_{S_1} - i_{S_2} + i_{R_1} = f_2 = 0$$

$$i_{S_3} + i_{S_4} - i_{R_1} = f_3 = 0$$

$$U_0 - U_{S_1} + U_{S_2} = f_4 = 0$$

$$-U_0 - U_{S_3} + U_{S_4} = f_5 = 0$$

$$-U_{R_1} - U_{S_2} - U_{S_4} = f_6 = 0$$

$$U_{R_1} - R_1 \cdot i_{R_1} = f_7 = 0$$

$$OS1 \cdot i_{S_1} + (1 - OS1) \cdot U_{S_1} = f_8 = 0$$

$$\text{if } [\text{not}(U_{S_1} > 0) \text{ and not } (i_{S_1} > 0)] \text{ then } 1 \text{ else } 0 - New(OS1) = f_{8a} = 0$$

$$OS2 \cdot i_{S_2} + (1 - OS2) \cdot U_{S_2} = f_9 = 0$$

$$\text{if } [\text{not}(U_{S_2} > 0) \text{ and not } (i_{S_2} > 0)] \text{ then } 1 \text{ else } 0 - New(OS2) = f_{9a} = 0$$

$$OS3 \cdot i_{S_3} + (1 - OS3) \cdot U_{S_3} = f_{10} = 0$$

$$\text{if } [\text{not}(U_{S_3} > 0) \text{ and not } (i_{S_3} > 0)] \text{ then } 1 \text{ else } 0 - New(OS3) = f_{10a} = 0$$

$$OS4 \cdot i_{S_4} + (1 - OS4) \cdot U_{S_4} = f_{11} = 0$$

$$\text{if } [\text{not}(U_{S_4} > 0) \text{ and not } (i_{S_4} > 0)] \text{ then } 1 \text{ else } 0 - New(OS4) = f_{11a} = 0$$

$$-U_{L_1} + L_1 \cdot \frac{i_{L_1} - i_{L_{1old}}}{h} = \hat{f}_7 = 0$$
$$-i_{C_1} + C_1 \cdot \frac{U_{C_1} - U_{C_{1old}}}{h} = \tilde{f}_7 = 0$$

If either the inductor L_1 or the capacitor C_1 replaces the resistor R_1 in Fig.6.1, the equation f_7 is replaced by either $\hat{f}_7$ or $\tilde{f}_7$, and U_{R_1} and i_{R_1} are replaced by either U_{L_1} and i_{L_1} or U_{C_1} and i_{C_1}. However, the behavior of the DAE matrix system does not change at all if we use the difference formulae concept.

$$\underbrace{\begin{pmatrix} 1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -R_1 & 0 & 0 & 0 & 0 & 1 \\ OS1 & 0 & 0 & 0 & 0 & 0 & (1-OS1) & 0 & 0 & 0 & 0 \\ 0 & OS2 & 0 & 0 & 0 & 0 & 0 & (1-OS2) & 0 & 0 & 0 \\ 0 & 0 & OS3 & 0 & 0 & 0 & 0 & 0 & (1-OS3) & 0 & 0 \\ 0 & 0 & 0 & OS4 & 0 & 0 & 0 & 0 & 0 & (1-OS4) & 0 \end{pmatrix}}_{A} \cdot \begin{pmatrix} i_{S_1} \\ i_{S_2} \\ i_{S_3} \\ i_{S_4} \\ i_0 \\ i_{R_1} \\ U_{S_1} \\ U_{S_2} \\ U_{S_3} \\ U_{S_4} \\ U_{R_1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -U_0 \\ U_0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Only the seventh row in this matrix equation system changes if the resistor is replaced by an inductor or a capacitor . The elements $A(7,6)$ and $A(7,11)$ would either change to L_1 and $-h$ in the case of an inductor, or to $-h$ and C_1 in the case of a capacitor. The vector on the right side would change in the seventh row to either

Table 6.1: Values for System Determinant

switch position				R_1 case	L_1 case	C_1 case	reason for
OS1	OS2	OS3	OS4	$\det(A)$	$\det(A)$	$\det(A)$	singularity
1	1	1	1	0	0	0	R_1, L_1 or C_1 floating
1	1	1	0	-1	h	$-C_1$	–
1	1	0	1	-1	h	$-C_1$	–
1	1	0	0	0	0	0	i_0 cannot be determined
1	0	1	1	-2	$2h$	$-2C_1$	–
1	0	1	0	$-1+R_1$	$h-L_1$	$h-C_1$	–
1	0	0	1	R_1	$-L_1$	h	–
1	0	0	0	0	0	0	i_0 cannot be determined
0	1	1	1	-2	$2h$	$-2C_1$	–
0	1	1	0	$-1+R_1$	$h-L_1$	$h-C_1$	–
0	1	0	1	R_1	$-L_1$	h	–
0	1	0	0	0	0	0	i_0 cannot be determined
0	0	1	1	0	0	0	i_0 cannot be determined
0	0	1	0	0	0	0	i_0 cannot be determined
0	0	0	1	0	0	0	i_0 cannot be determined
0	0	0	0	0	0	0	i_0 cannot be determined 2 parallel closed switches

$L_1 \cdot i_{L_{1old}}$ or $C_1 \cdot U_{C_{1old}}$. Yet, this change would not influence the solvability of the matrix equation in any way. Table 6.1 describes the values of the determinant of A for all possible sixteen switch positions. The three columns represent the three cases of elements connected to the rectifier.

Table 6.1 shows that singularities occur in the same eight switch cases independently of the connected element. The last chapter showed that the difference formulae concept equalizes the behaviors of resistors, capacitors, and inductors. Hence the independence of the singularities from the connected element is not surprising.

Let us now take a look at the bond graph for the resistive load to explain the singularities that occur in these eight cases. Fig. 6.2 shows the bond graph for the circuit containing the causality requirements mandated by the voltage source.

If we allow either (D_1 and D_2) or (D_3 and D_4) to be conducting simultaneously ($OS1 = OS2 = 0$ or $OS3 = OS4 = 0$), the causality requirements for the corresponding 1– and 0–junctions require that the bond connecting them be left **without** a causality stroke. This behavior is shown in Fig. 6.3 simultaneously for both cases. From another perspective, fixing the causality stroke of the D_1 (or D_3) diode to its conducting position at the associated 1–junction and propagating the resulting causality strokes through the circuit, it becomes evident that the causality of the corresponding D_2 (D_4) diode is already fixed to its non–conducting position. Consequently, if the D_1 (D_3) diode is said to be conducting, the causality of the corresponding D_2 (D_4) diode cannot be chosen independently. This explains the observed singular determinant in seven of the eight cases.

The last case is illustrated in Fig. 6.4. If both diodes D_1 and D_3 are fixed in their non–conducting position, the position of either diode D_2 or diode D_4 can still be chosen freely. However, the last diode's position is predetermined. If all four diodes are

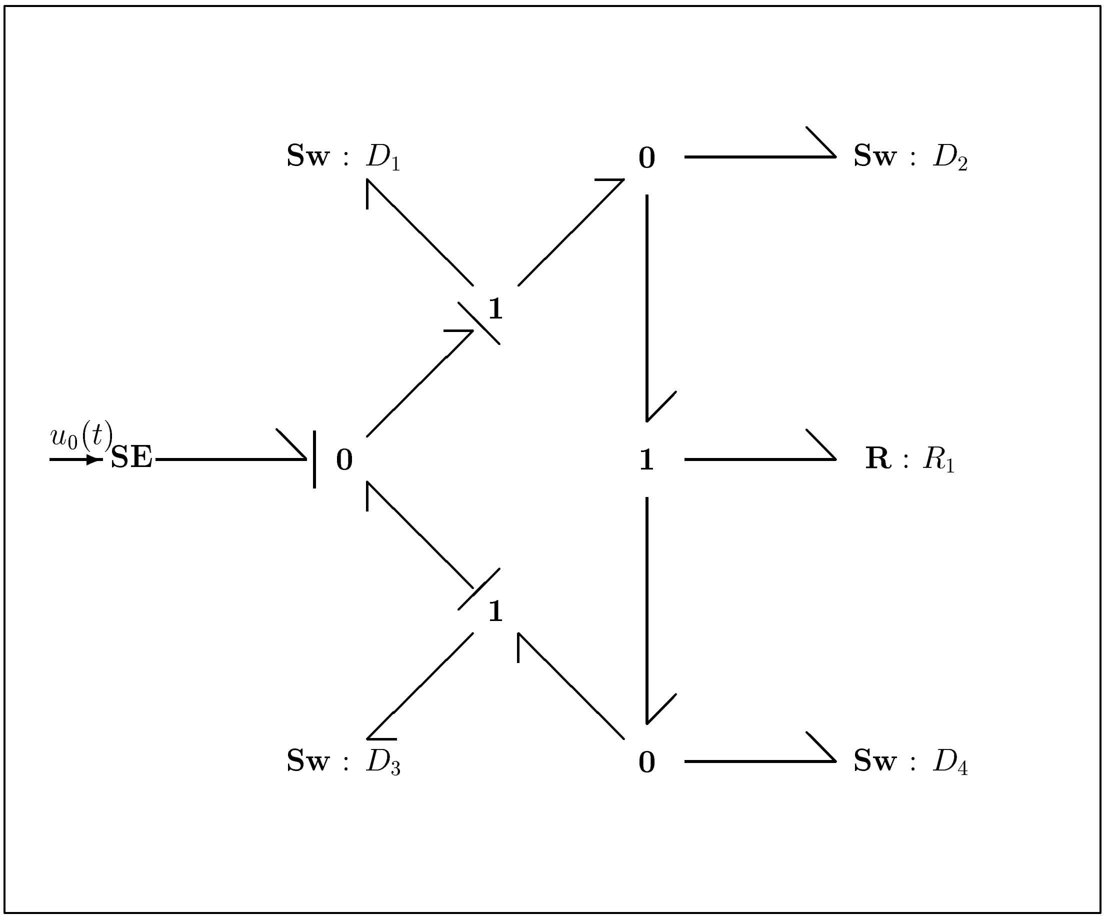


Figure 6.2: Bond Graph (1) Two Quadrant Rectifier

forced into their non–conducting positions, as shown in Fig. 6.4, the 1–junction at the load is left with a single stroke, which is in violation of the causality requirements for 1–junctions.

From a physical perspective, the first seven cases lead to situations, whereby the voltage source is shortcircuited. The eighth case leads to a floating load. If all four diodes are non–conducting, the load is decoupled from the ground, and its potential can no longer be known.

Six of the seven shortcircuit situations can be solved by placing an impedance in series with the voltage source, as done in the subsequent example of the SCR circuit. However, the seventh case (all four diodes are conducting) still leads to a singular determinant, because it exhibits two parallel wires. In this case, it is impossible to compute the current flowing through each of these two wires. Only the sum of currents can be known.

6.2 The SCR Circuit

Reference [10] describes the use of the circuitry shown in Fig. 6.5 to control the motion of a train. The train engine is represented by a negative current source that

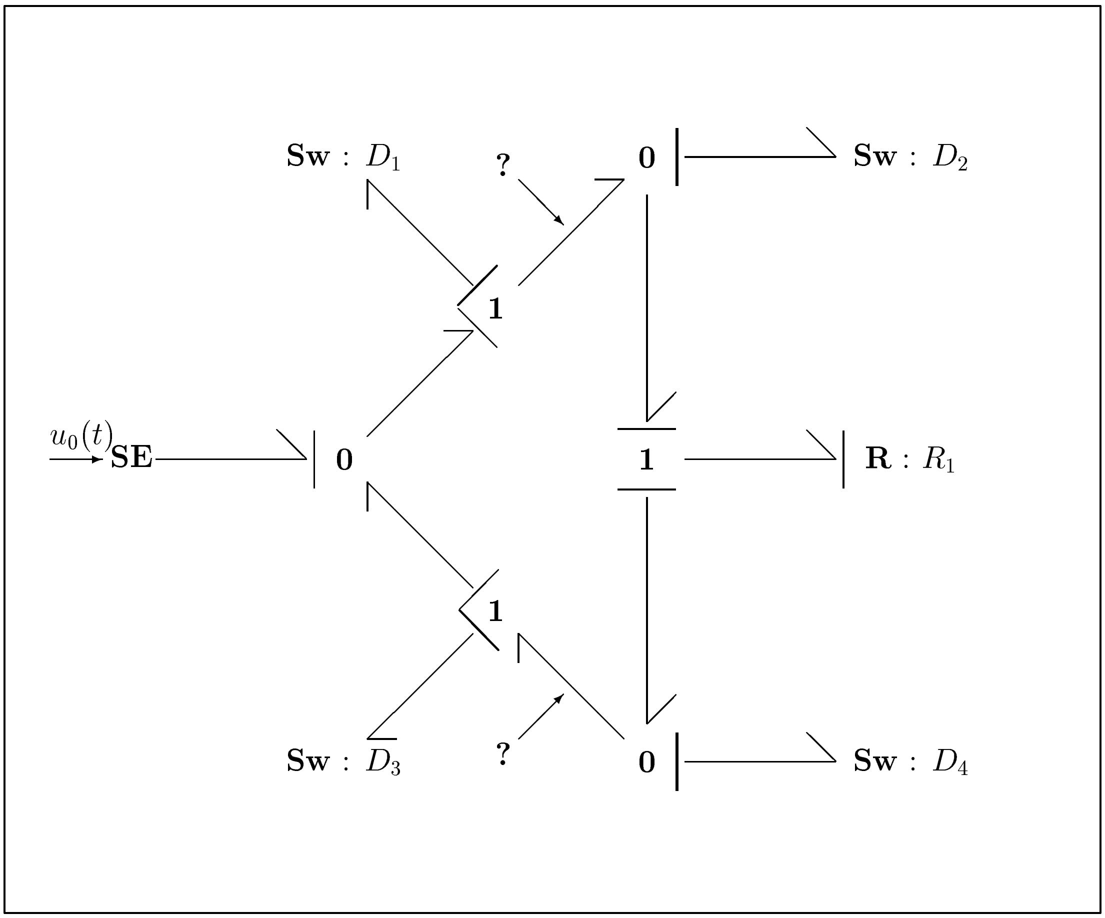


Figure 6.3: Bond Graph (2) Two Quadrant Rectifier

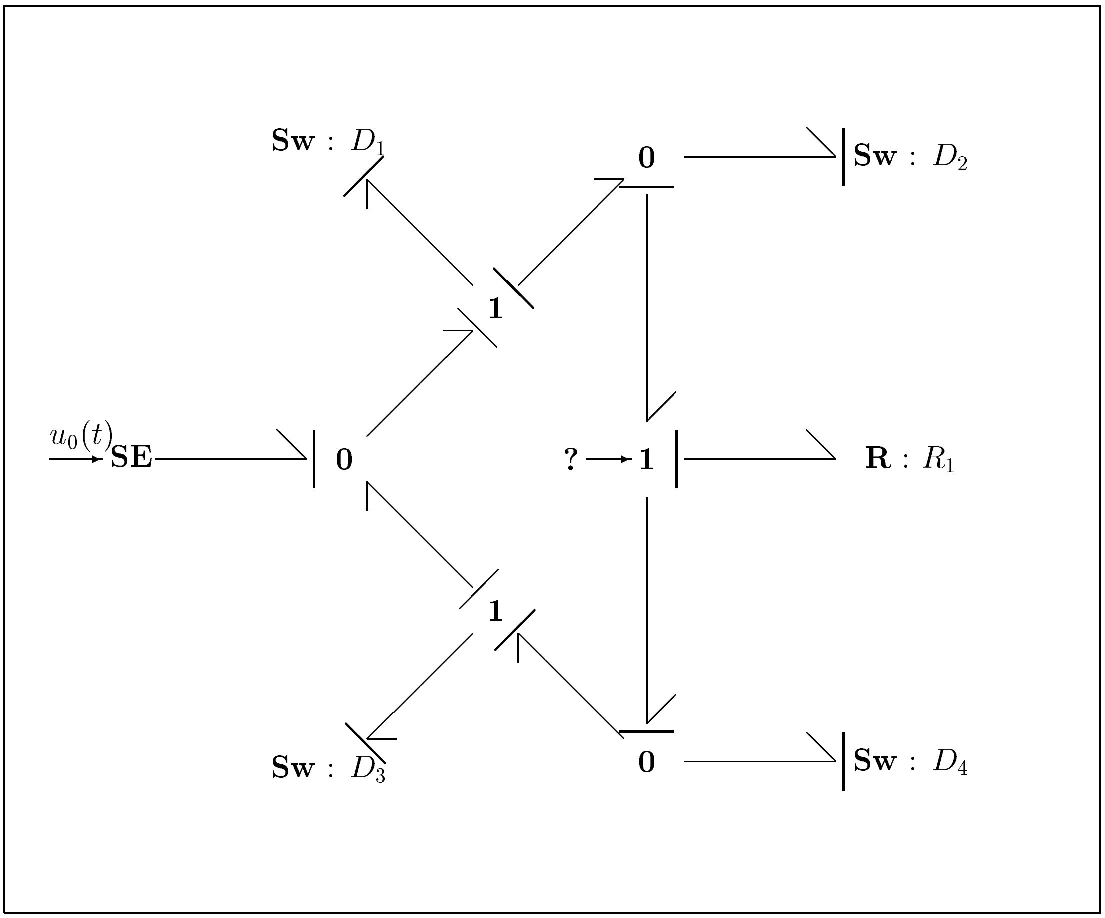


Figure 6.4: Bond Graph (3) Two Quadrant Rectifier

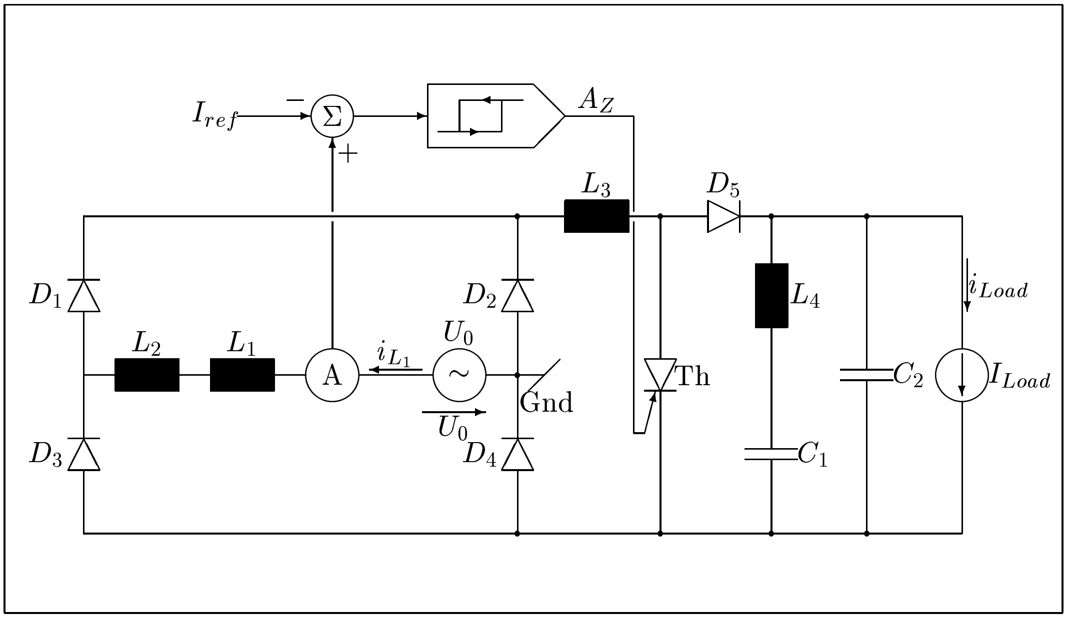


Figure 6.5: SCR Circuit for Train Speed Control (1)

drains current out of the net. The gate control logic is also shown in Fig. 6.5. The line current i_{L_1} is controlled in such a way that it always remains in the vicinity of:

$$I_{ref} = \frac{15 \cdot 10^6}{\hat{U}_0} \sin \omega t \tag{6.1}$$

The width of the tolerance band around I_{ref} is $B_T = 200$ A. The circuit basically operates in one of four modes that are described in the Table 6.2.

Hereby the statement that i_{L_1} is "negative and increasing" means that i_{L_1} is getting closer to zero. Table 6.2 contains only two of the four possible cases for the two switch variables $OS5$ and $OS6$. The circuit model is not really accurate

Table 6.2: Circuit Operation Modes

OP #	I_{ref}	OS5	OS6	i_{L_1}	resulting from	i_{L_1} is
1	> 0	0	1	$i_{L_1} < I_{ref}$	$i_{L_1} < I_{ref} - B_T$	positive, increasing
2	> 0	1	0	$i_{L_1} > I_{ref}$	$i_{L_1} > I_{ref} + B_T$	positive, decreasing
3	< 0	1	0	$i_{L_1} < I_{ref}$	$i_{L_1} < I_{ref} - B_T$	negative, increasing
4	< 0	0	1	$i_{L_1} > I_{ref}$	$i_{L_1} > I_{ref} + B_T$	negative, decreasing

in modeling this behavior. First, the thyristor does not open as soon as the input A_Z becomes zero, but it opens only if the voltage is negative. Yet, the thyristor Th should change its operational mode based on the input A_Z alone. Second, the diode D_5 should close simultaneously with the opening of the thyristor Th. Otherwise the inductor current i_{L_3} would jump to zero, and this cannot happen in a physical system. However, if we change the switch characteristics of the thyristor Th and diode D_5 to an ideal toggle switch, we resolve the above problems. The ideal toggle switch allows only one of the two discrete switch variables $OS5$ and $OS6$ to be 0, while the second is 1. This restriction can mathematically be represented by $OS5+OS6 = 1$. The toggle switch depends only on A_Z, and thus also resolves the first problem. Consequently, equations f_{21a} and f_{22a} were corrected to:

$$\text{if } (A_Z > 0) \text{ then } 1 \text{ else } 0 - New(OS5) = f_{21a} = 0$$

$$\text{if } [\text{not}(A_Z > 0)] \text{ then } 1 \text{ else } 0 - New(OS6) = f_{22a} = 0$$

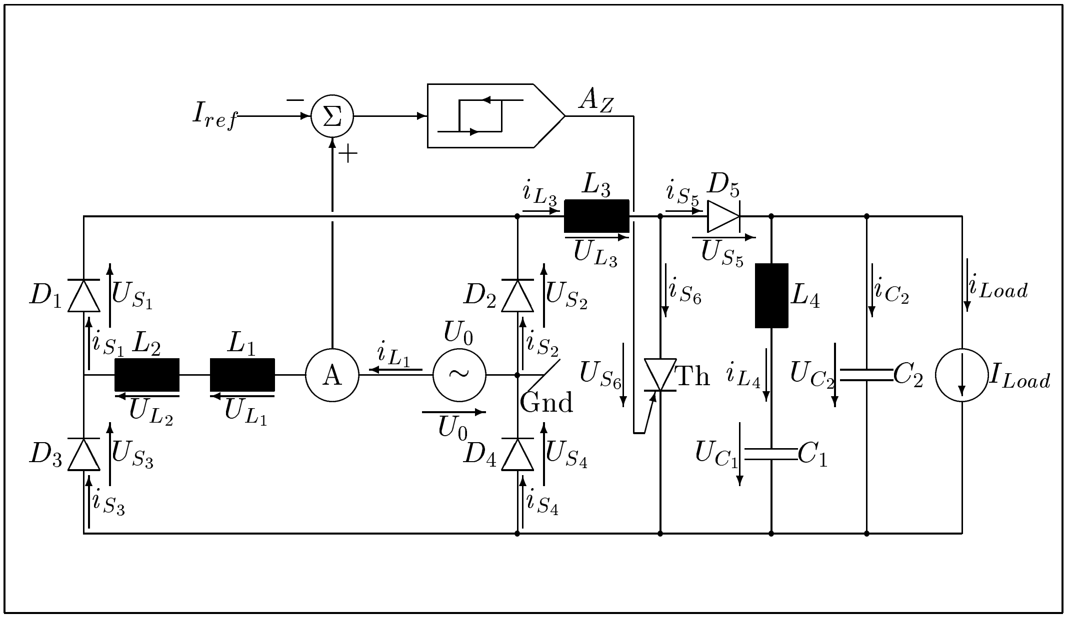


Figure 6.6: SCR Circuit for Train Speed Control (2)

for the simulation part. These equations represent the behavior of the ideal toggle switch controlled by A_Z.

6.3 The Equation System

The following set of equations describes the circuit. The current and voltage variables are all shown in Fig. 6.6. The switch characteristics still contain the uncorrected behavior.

$$i_{S_1} - i_{S_3} - i_{L_1} = f_1 = 0$$

$$
\begin{aligned}
-i_{S_1} - i_{S_2} + i_{L_3} &= f_2 = 0 \\
-i_{S_5} - i_{S_6} + i_{L_3} &= f_3 = 0 \\
i_{S_5} - i_{L_4} - i_{C_2} - i_{Load} &= f_4 = 0 \\
-i_{S_3} - i_{S_4} + i_{S_6} + i_{L_4} + i_{C_2} + i_{Load} &= f_5 = 0 \\
U_{S_1} - U_{S_2} + U_{L_1} + U_{L_2} - U_0 &= f_6 = 0 \\
U_{S_3} - U_{S_4} - U_{L_1} - U_{L_2} + U_0 &= f_7 = 0 \\
U_{S_2} + U_{S_4} + U_{S_6} + U_{L_3} &= f_8 = 0 \\
U_{S_5} - U_{S_6} + U_{L_4} + U_{C_1} &= f_9 = 0 \\
-U_{L_4} - U_{C_1} + U_{C_2} &= f_{10} = 0 \\
-U_{L_1} \cdot h + L_1 \cdot (i_{L_1} - i_{L_{1old}}) &= f_{11} = 0 \\
-U_{L_2} \cdot h + L_2 \cdot (i_{L_1} - i_{L_{2old}}) &= f_{12} = 0 \\
-U_{L_3} \cdot h + L_3 \cdot (i_{L_3} - i_{L_{3old}}) &= f_{13} = 0 \\
-U_{L_4} \cdot h + L_4 \cdot (i_{L_4} - i_{L_{4old}}) &= f_{14} = 0 \\
-i_{L_4} \cdot h + C_1 \cdot (U_{C_1} - U_{C_{1old}}) &= f_{15} = 0 \\
-i_{C_2} \cdot h + C_2 \cdot (U_{C_2} - U_{C_{2old}}) &= f_{16} = 0 \\
OS1 \cdot i_{S_1} + (1 - OS1) \cdot U_{S_1} &= f_{17} = 0 \\
\text{if } [\text{not}(U_{S_1} > 0) \text{ and not } (i_{S_1} > 0)] \text{ then } 1 \text{ else } 0 - New(OS1) &= f_{17a} = 0 \\
OS2 \cdot i_{S_2} + (1 - OS2) \cdot U_{S_2} &= f_{18} = 0
\end{aligned}
$$

$$\begin{aligned}
\text{if}\,[\text{not}(U_{S_2} > 0)\,\text{and not}\,(i_{S_2} > 0)]\,\text{then 1 else 0} - New(OS2) &= f_{18a} = 0 \\
OS3 \cdot i_{S_3} + (1 - OS3) \cdot U_{S_3} &= f_{19} = 0 \\
\text{if}\,[\text{not}(U_{S_3} > 0)\,\text{and not}\,(i_{S_3} > 0)]\,\text{then 1 else 0} - New(OS3) &= f_{19a} = 0 \\
OS4 \cdot i_{S_4} + (1 - OS4) \cdot U_{S_4} &= f_{20} = 0 \\
\text{if}\,[\text{not}(U_{S_4} > 0)\,\text{and not}\,(i_{S_4} > 0)]\,\text{then 1 else 0} - New(OS4) &= f_{20a} = 0 \\
OS5 \cdot i_{S_5} + (1 - OS5) \cdot U_{S_5} &= f_{21} = 0 \\
\text{if}\,[\text{not}(U_{S_5} > 0)\,\text{and not}\,(i_{S_5} > 0)]\,\text{then 1 else 0} - New(OS5) &= f_{21a} = 0 \\
OS6 \cdot i_{S_6} + (1 - OS6) \cdot U_{S_6} &= f_{22} = 0 \\
\text{if}\,[\text{not}((U_{S_6} > 0)\,\text{and}\,(A_Z > 0))\,\text{and not}\,((i_{S_4} > 0)\,\text{and not}\,(OS6 > 0))] & \\
\text{then 1 else 0} - New(OS6) &= f_{22a} = 0
\end{aligned}$$

This system can be written in matrix form. The matrix equation is not included because of its size. In this example, we deal with 22 unknowns in 22 equations, as well as six additional equations that describe the switch characteristics. The determinant of this circuit was examined for all $2^6 = 64$ possible switch cases. The determinant was singular in 14 of these 64 cases, and the next section contains the explanation for these 14 singular cases.

Table 6.3: Singular Cases SCR Circuit

case number	switch position OS1	OS2	OS3	OS4	OS5	OS6	reason for singularity
1	0	0	0	0	0	0	D_1, D_2, D_3, and D_4 are closed i_{L_1} cannot be distributed between D_1–D_2–branch and D_3–D_4–branch, see Fig. 6.7
2	0	0	0	0	0	1	
3	0	0	0	0	1	0	
4	0	0	0	0	1	1	
5	0	0	1	1	1	1	D_3, D_4, D_5, and D_6 are opened L_4, C_1, C_2, and I_{Load} are floating, see Fig. 6.8
6	0	1	1	1	1	1	
7	1	0	1	1	1	1	
14	1	1	1	1	1	1	
8	1	1	0	0	1	1	D_1, D_2, D_5, and D_6 are opened L_3 is floating, see Fig. 6.9
9	1	1	0	1	1	1	
10	1	1	1	0	1	1	
14	1	1	1	1	1	1	
11	1	1	1	1	0	0	D_1, D_2, D_3, and D_4 are opened D_5, Th, L_3,L_4, C_1, C_2, and I_{Load} are floating, see Fig. 6.10
12	1	1	1	1	0	1	
13	1	1	1	1	1	0	
14	1	1	1	1	1	1	

6.3.1 The Cases Causing Singularities

Table 6.3 lists the singular cases. Case 14 is contained three times, because this switch combination is contained in three blocks. The cases are organized into blocks that exhibit a common reason for their singularity. For case 14, that is contained in three blocks, all reasons apply simultaneously.

If the thyristor Th and the diode D_5 are replaced by a toggle switch, as shown in Fig. 6.11, the second and third blocks are eliminated. Thus, we have only two blocks with singular determinants left corresponding to the first and last singular cases of

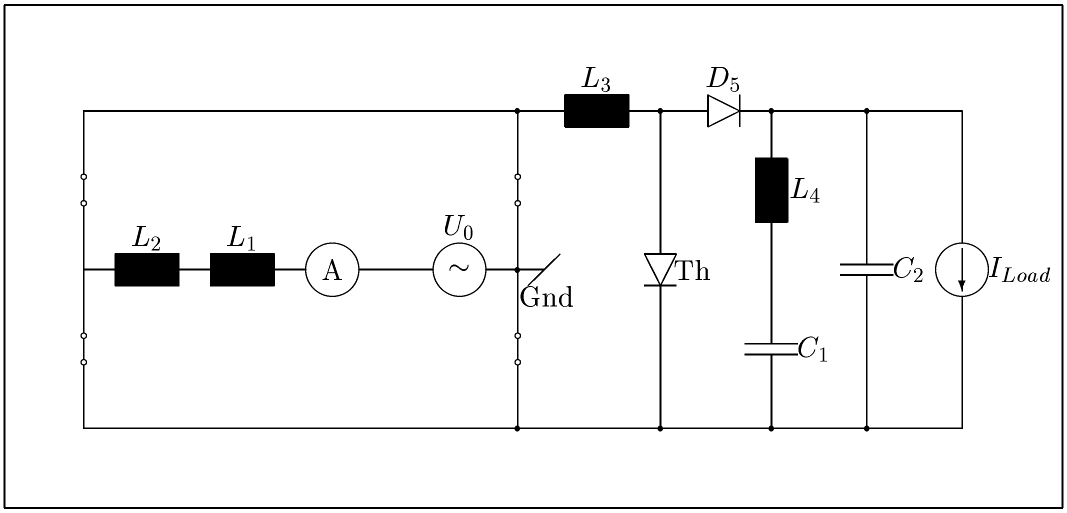


Figure 6.7: Singular Cases SCR Circuit (1)

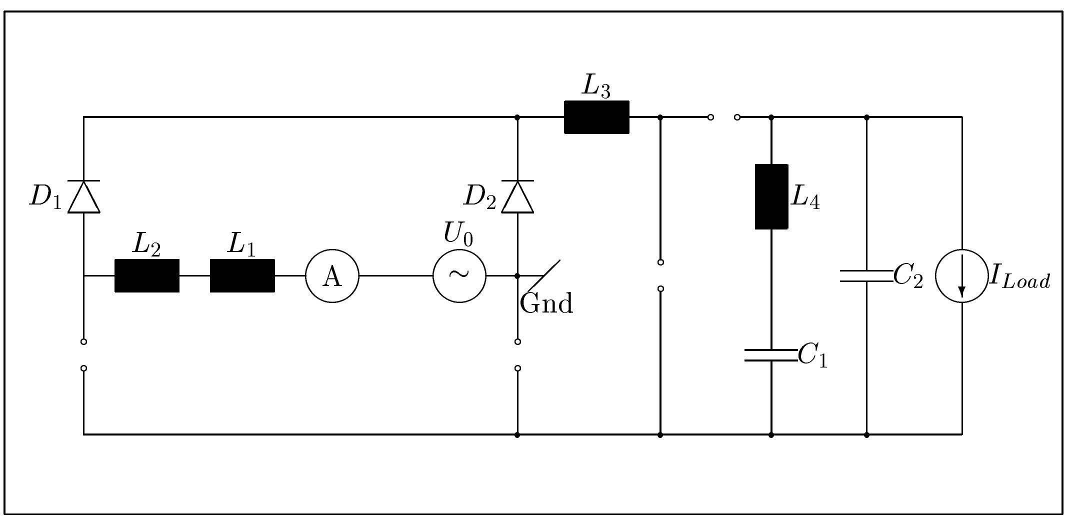


Figure 6.8: Singular Cases SCR Circuit (2)

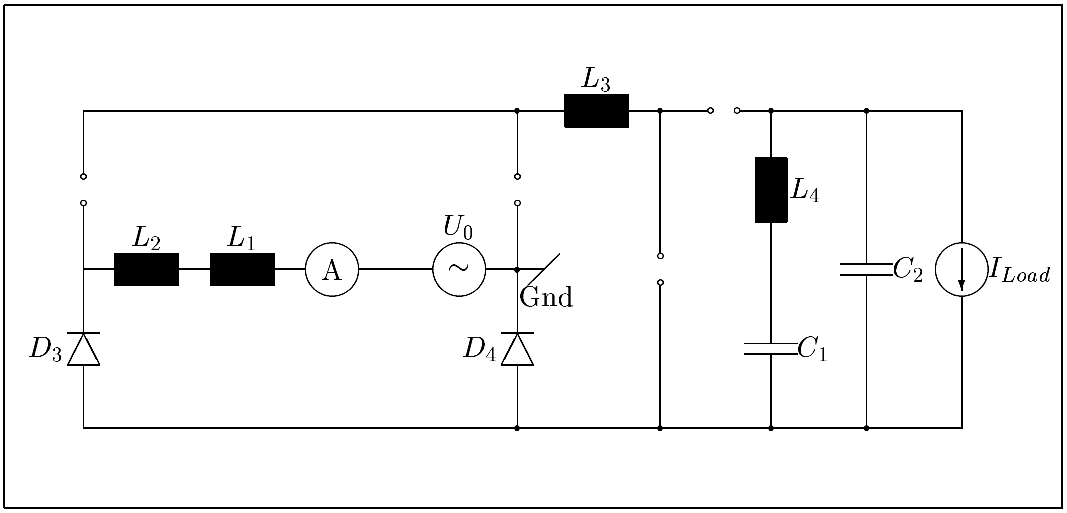


Figure 6.9: Singular Cases SCR Circuit (3)

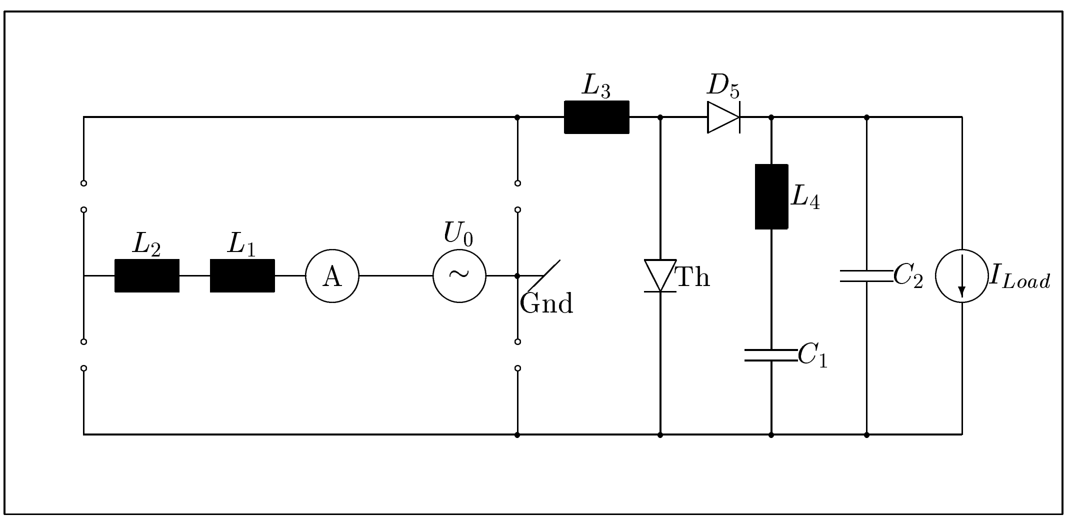


Figure 6.10: Singular Cases SCR Circuit (4)

the previous two quadrant rectifier example.

The cases where $OS5$ is equal to $OS6$ were also eliminated, and thus we are left with two singular cases for each of these blocks. Ideally, the circuit should never reach either of these four switch cases. However at the end of the first positive half–wave of the source voltage U_0, the diodes D_2 and D_3 are conducting before D_1 and D_4 are non–conducting. The corresponding case in which all four diodes are non–conducting occurs at the end of the negative half–wave of the source voltage U_0. Hence, our simulation would end in these cases if we don't modify the simulation program. The four possible singular cases were prevented in the simulation by a surplus switch logic. Connecting all floating elements with a high resistor value to ground and adding a small resistor value in series wherever a current cannot be determined would be another way to prevent singular cases. Thus, the simulation of ideal switch elements makes it necessary to either modify the switch logic, or to add shunt resistors in order to prevent singular cases.

6.4 Simulation Results

Fig. 6.12 shows the plot of i_{L_1} and i_{ref} over time. The inductor current remains indeed in the vicinity of the reference current. The inductor current needs some time at the beginning of each positive and negative half–wave to follow the reference current.

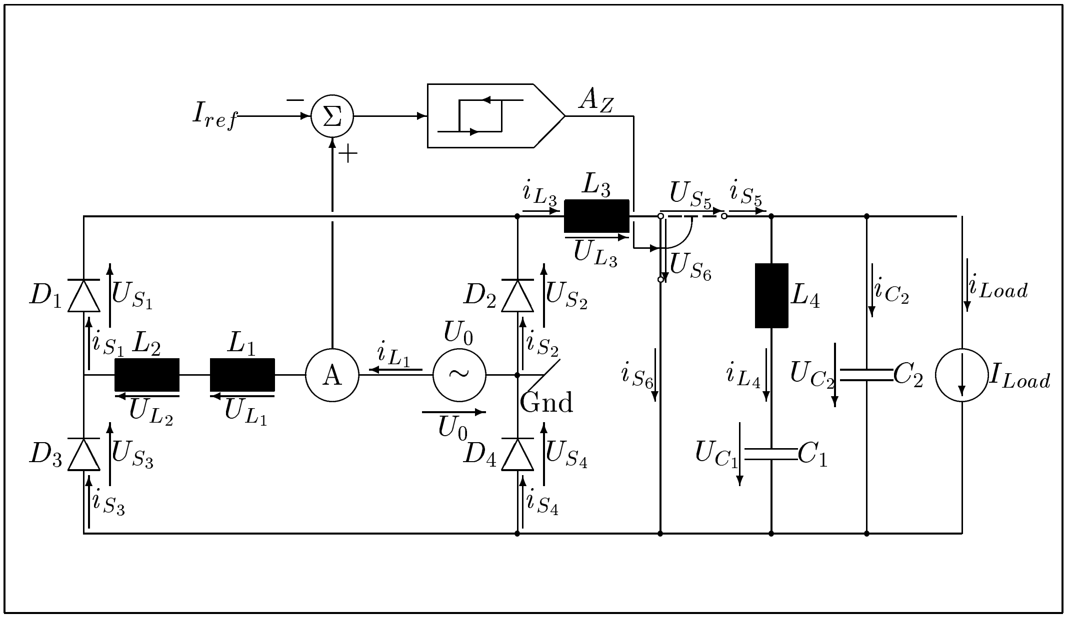


Figure 6.11: SCR Circuit for Train Speed Control (3)

Fig. 6.13 shows the filter voltage $U_F = U_0 - U_{L_1}$. U_F is the potential difference between the node to which the inductor elements L_1 and L_2 are connected and the ground node. The switching caused by the nonlinear control element is quite obvious in this figure. The peak at the onset of the negative half–wave is caused by the inexact determination of the event time. Remember, that the simulation after Fig. 5.8 determines the event time only with the precision of the accuracy constant t_{eps}.

Fig. 6.14 shows the voltage $U_Z = U_{C_2}$. This plot shows the voltage across the current source. The current source models the train engine, and thus, this plot determines the amount of power transferred from the power net to the train engine.

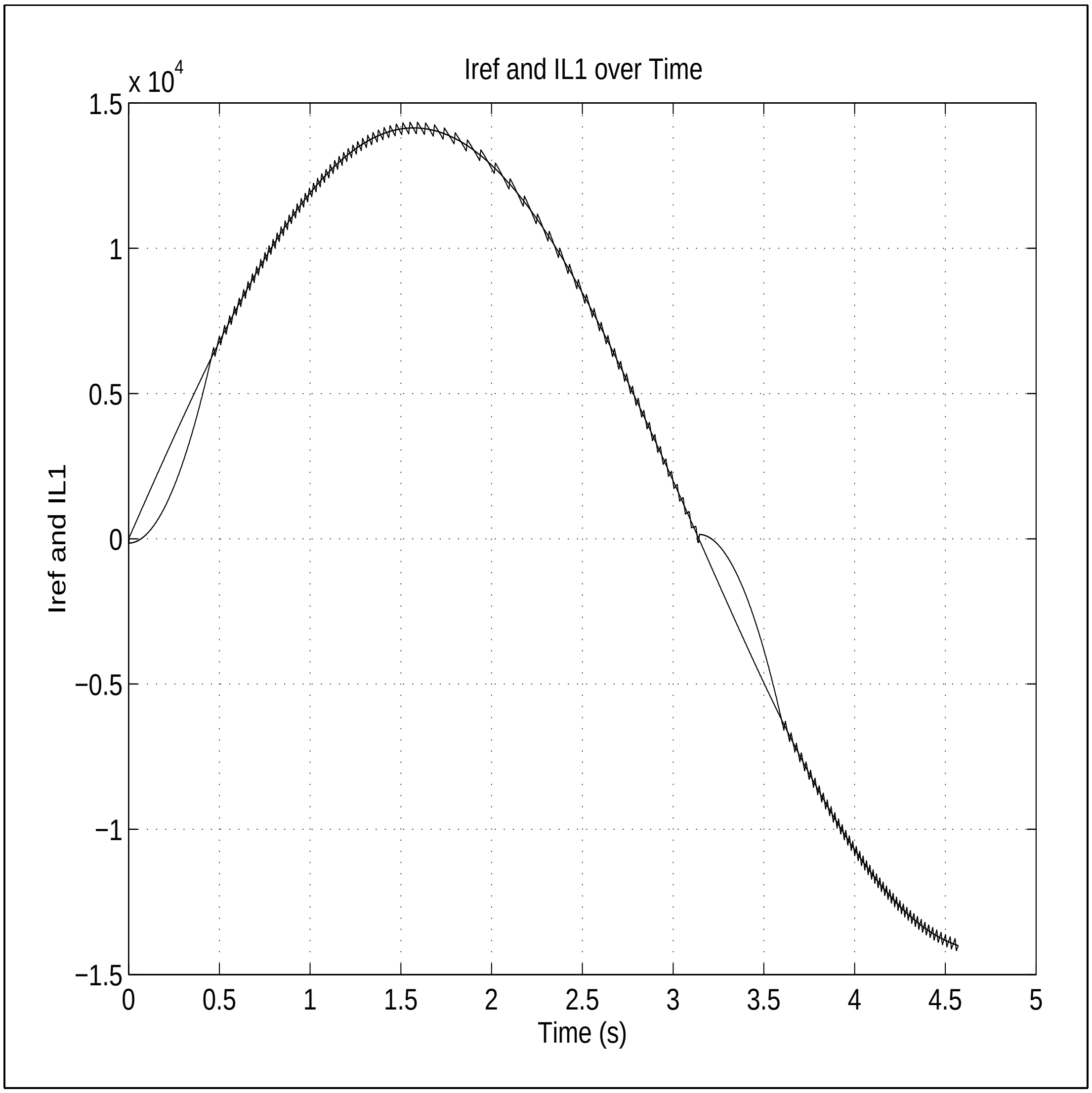


Figure 6.12: Simulation Results SCR Circuit (1)

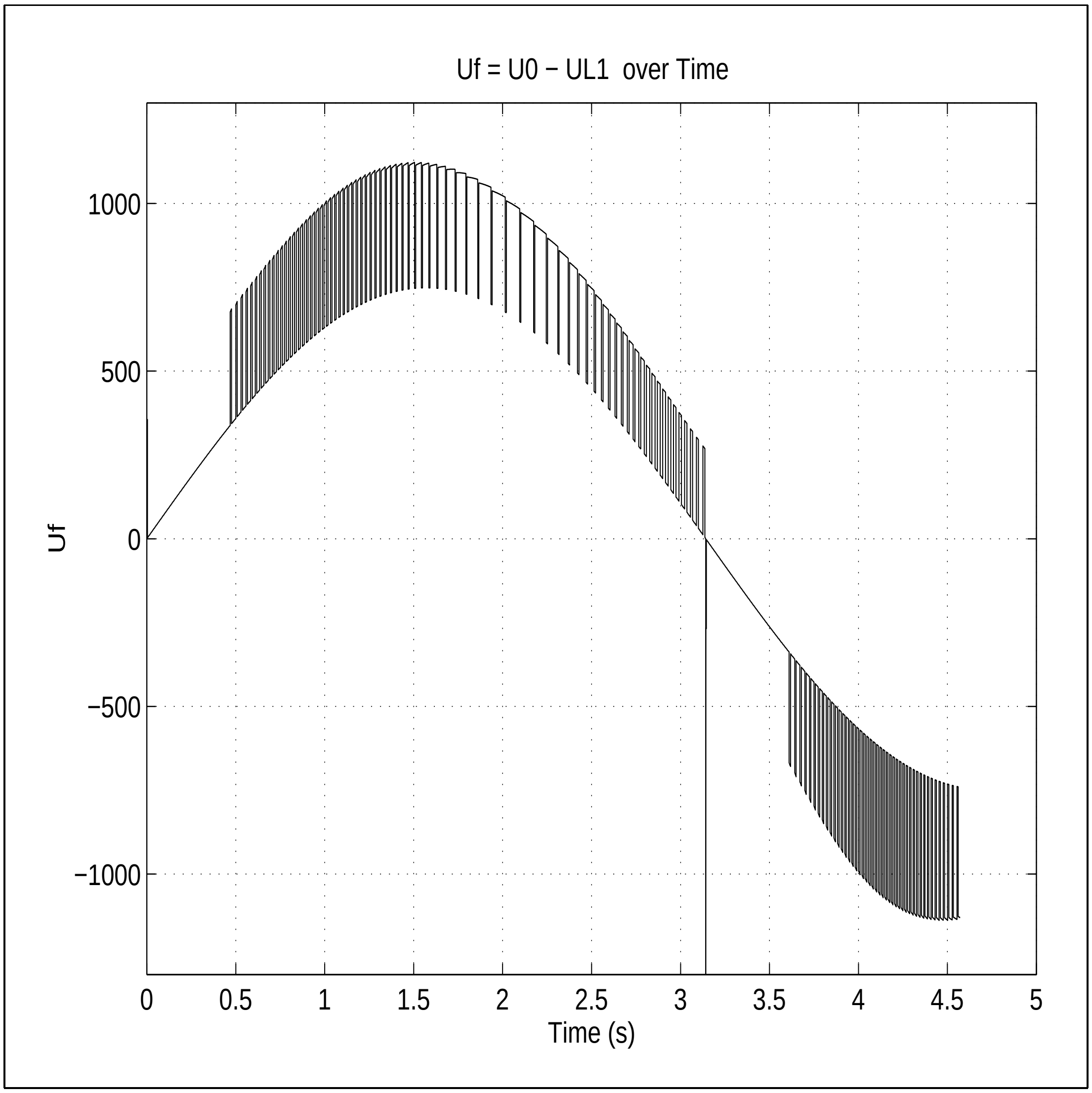


Figure 6.13: Simulation Results SCR Circuit (2)

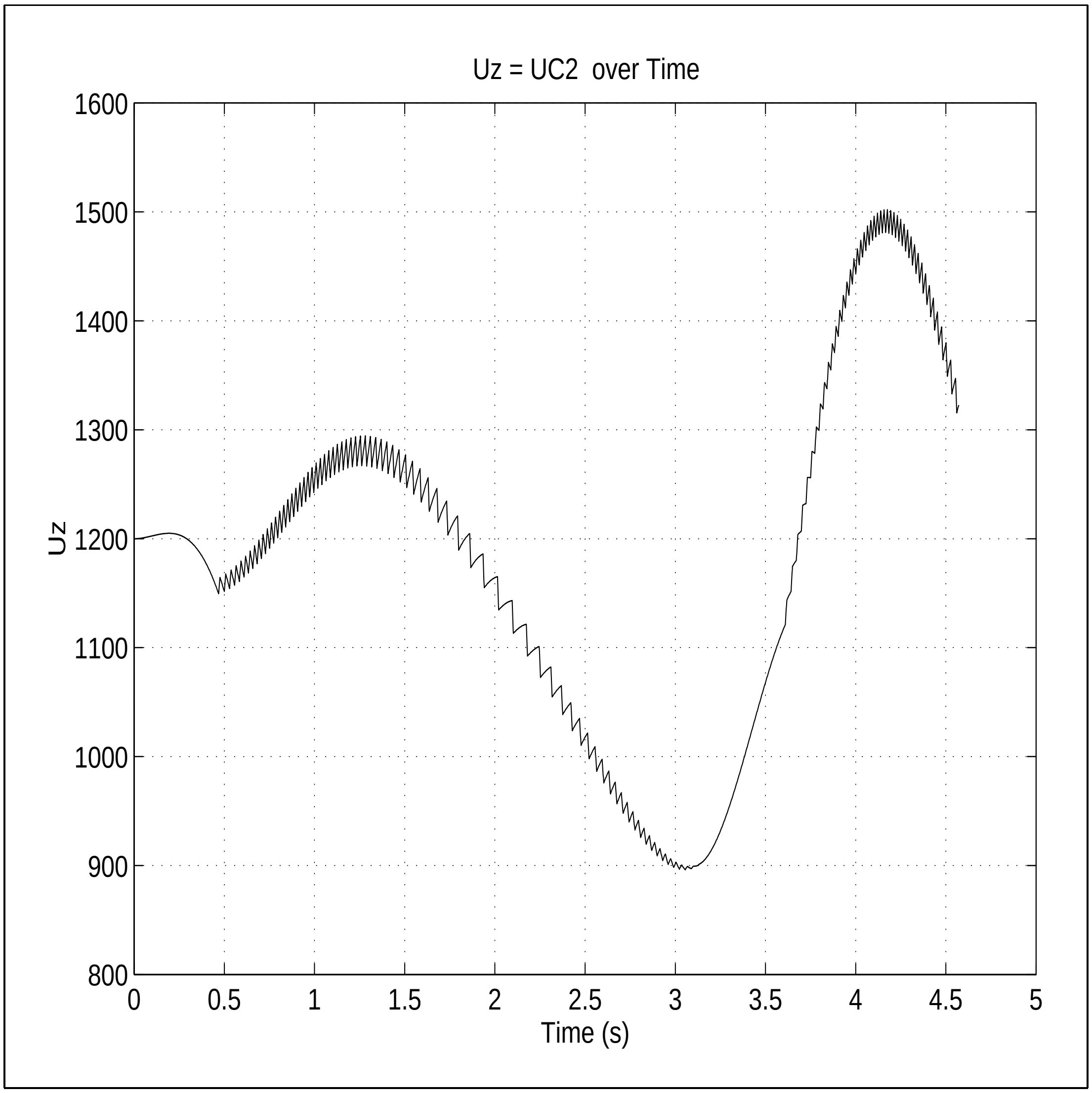


Figure 6.14: Simulation Results SCR Circuit (3)

CHAPTER 7

Discussion

In this thesis, the requirements for a modified Pantelides algorithm were investigated. The search for modified switch equations for a relatively simple example showed that this approach does not bring us closer to the desired goal, to determine a single simulation model for an arbitrary complex conditional index system.

However the use of Difference Formulae allowed us to solve the example circuit in a totally different way. By replacing the differential equations governing the behavior of capacitors and inductors by difference formulae, the resulting bond graph causality constraints for storage elements vanish. The storage elements now behave exactly like resistance elements. The use of difference formulae eliminates causality constraints for storage elements and thereby resolves the higher index problem.

A more complicated example, a SCR controller for train speed control, showed that there are problems remaining. These problems can all be characterized by the nature of ideal switch simulation. The remaining problems in the linear circuit theory are caused by either floating elements or parallel shortings. The potentials and thus the voltages across two pins cannot be calculated in the case of one or more floating elements. The current cannot be distributed between two branches with exactly zero resistance in the case of parallel shortings.

A simulation using the difference formulae will not work in singular cases caused by either of these *ideal switch problems*. Hence these cases must be prevented.

There are several possibilities for preventing singular cases. First, floating elements can be reconnected by introducing a small conductance to ground, whereas the distribution of current in parallel shortened branches is made possible by introducing two small resistances into the two previously shortened parallel paths. Second, the switch logic can be altered by superimposing a switch logic on the circuit that prevents the singular cases on from ever happening. A third possibility is the use of separate models for the singular cases. In this approach, we must replace the non–functional ideal switch model with a non–ideal model. All three approaches somehow modify the properties of ideal switches in order to avoid switch positions that are impossible to simulate with ideal switch elements.

7.1 Comparison with PSpice

One of the most widely used commercial simulation programs for electrical circuits, PSpice, uses difference formulae for the simulation of DAE models. However, this simulation package does not allow the user to simulate ideal switch elements. Instead non–ideal switch elements that result in artificial stiffness and time–consuming simulation runs are used. The important difference between ideal and non–ideal switch element simulations were already explained at the end of chapter 1.
Yet, the difference formulae method is also suited for simulating ideal switches, and by doing so avoids the high simulation time and cost disadvantage. The difference formulae resolve the causality assignment problem for storage elements that was caused by the computational need of the ODE simulation. The use of difference formulae, of course, cannot resolve structural problems that are caused by ideal switch elements, that is the parallel shorting and the floating element problem.

7.2 Issues for Further Research

The simulation programs of the examples were written in Fortran and in Matlab. Implementing the difference formulae concept together with event handling in the object–oriented Dymola/Dymosim environment should be the main issue for further research. An automatic code generation can replace the difficult task of producing correct simulation code manually, as was done in this research effort for implementing the introduced concepts. This research should result in the possibility of modeling a conditional index system in a truly object–oriented fashion. The simulation user can then concentrate his or her efforts on the modeling task.

A comparison of the efficiency of the newly suggested concepts with commercial simulation approaches, such as those embraced by PSpice is another research task that directly results from the previous one. The efficiency should increase as the artificial stiffness problem vanishes with the use of difference formulae for derivatives. However the problems of the ideal switches still cause singularities that must be prevented by either a switch logic if these cases are not significant or use of a non–ideal switch in cases where they are significant. The simulation time should decrease, because the cases with the need for a non–ideal switch element should only make up for a small fraction of the cases that previously caused artificial stiffness.

Further research should be concentrated on resolving problems caused by the nature of ideal switches. Most likely, ideal switches should only be replaced by non–ideal switch elements in singular cases, and thus, the simulation time and cost can be kept as small as possible. However, an automated algorithm for determining these cases and to find the best substitution with non–ideal switch elements is needed.

REFERENCES

[1] François E. Cellier 1991. "Continuous System Modeling", Springer–Verlag New York, 755p.

[2] François E. Cellier, Martin Otter, Hilding Elmqvist 1995. "Bond Graph Modeling of Variable Structure Systems"

[3] Hilding Elmqvist, François E. Cellier, Martin Otter, 1993. "Object–Oriented Modeling of of Hybrid Systems", Proc. ESS'93, SCS European Simulation Symposium, Delft, The Netherlands, pp.xxxi - xli.

[4] Hilding Elmqvist, François E. Cellier, Martin Otter, 1995. "Inline Integration: A New Mixed Symbollicalic/Numeric Approach for Solving Differential–Algebraic Equation Systems", Proc. ESM'95, SCS European Simulation MultiConference, Prague, Czech Republic, pp.xxiii-xxxiv.

[5] Pantelides, C.C. 1988. "The Consistent Initialization of Differential–Algebraic Systems", SIAM J. *Scientific and Statistic Computing*, no. 9 (2): 213-231.

[6] Hilding Elmqvist 1993. "Dymola, Dynamic Modeling Language — User's Manual", Dynasim AB, Research Park Ideaon, Lund, Sweden

[7] Edward E. L. Mitchell and Joseph S. Gauthier 1991, "ACSL: Advanced Continuous Simulation Language – User Guide and Reference Manual", Mitchell & Gauthier Assoc., Concord, Mass.

[8] Stephen Wolfram 1992. "Mathematica Reference Guide", Addison–Wesley Reading, Mass., 305p.

[9] Robert Tarjan 1972, "Depth–First Search and Linear Graph Algorithms", SIAM J. *Journal of Computation*, 1(2), pp 146-160

[10] H.Schlunegger 1977. “Untersuchung eines netzrückwirkungsarmen, zwangskommutierten Triebfahrzeug–Stromrichters zur Einspeisung eines Gleichspannungszwischenkreises aus dem Einphasennetz”, Ph.D. dissertation, no. Diss. ETH 5867: The Swiss Federal Institute of Technology Zurich, Switzerland

[11] Martin Otter 1994. “ Objektorientierte Modellierung mechatronischer Systeme am Beispiel geregelter Roboter”, Ph.D. dissertation, Dept. of Mech. Engr., Ruhr–University Bochum, Germany