Transactions on Engineering Sciences vol 11, © 1996 WIT Press, www.witpress.com, ISSN 1743-3533

Object oriented approach for large circuits with substructures in the computer algebra program Maple V

A. Kugi,[a] K. Schlacher,[a] M. Kaltenbacher[b]
[a]*Department of Automatic Control, Johannes Kepler University, A-4040 Linz Auhof, Austria*
[b]*Department of Electrical Measurement, Johannes Kepler University, A-4040 Linz Auhof, Austria*

This paper is concerned with the symbolic derivation and solution of the minimal set of network equations in MapleV. By means of the table environment of MapleV it is possible to realize the essential features of an object oriented design. All considered linear electric terminals are arranged in a class hierarchy using the principle of multiple inheritance. The network algorithm for setting up the equations is based on the recursive graph searching method depth first. The algorithm is extended by a special substructure technique, in order to be able to calculate complex networks with a higher number of network elements. Finally, the feasibility of the proposed approach will be demonstrated by two application examples.

1 Introduction

During the past decade, there have been some significant advances in the area of computer algebra programs. Thus the symbolic calculation of mathematical models becomes more and more common. Especially in the field of control theory most of the controller design methods require a symbolic description of the controlled plant. Furthermore, these models allow to investigate the effects of parameter variations on the system behavior in an easy manner.

We will present an efficient method for the automatic derivation of the network equations in the computer algebra program MapleV. Thereby only networks consisting of linear terminals are considered. The result can be a system of ordinary differential equations or transfer functions. Without great effort it is also possible to extend the network algorithm to handle systems with nonlinear terminals.

2 Fundamentals of Network Theory

2.1 Kirchhoff Laws

We consider linear networks, which are formed by connecting together different terminals like resistors, inductors, various ideal and controlled sources, operational amplifiers etc.. A connected graph $G = \{N, B\}$ with the finite set of nodes N and the finite set of branches B gives a mathematical description of the network, whereas the end points of a branch must be nodes.

The current state is defined as a point $i = \left(i_1, i_2, \ldots, i_{|B|}\right)^T \in R^{|B|}$, where $|B|$ denotes the cardinality of the set B and i_k determines the current flowing through the kth branch. Now the Kirchhoff current law can be expressed as follows

$$\sum_k d_l^k i_k = 0 \tag{1}$$

with

$$d_l^k = \begin{cases} 1 & \text{if the branch } k \text{ is oriented } \textit{to} \text{ the node } l \\ -1 & \text{if the branch } k \text{ is oriented } \textit{off} \text{ the node } l \\ 0 & \text{otherwise.} \end{cases}$$

A current state i is said to be *admissible*, if i satisfies the Kirchhoff current law. Hence we obtain from eq. (1) that i is admissible, iff $i \in \operatorname{Ker} D$ with the linear map $D : R^{|B|} \to R^{|N|}$ defined in the form $\sum_k d_l^k i_k = x_l$ for some $i \in R^{|B|}$ and $x \in R^{|N|}$.

The voltage state is given by a point $u = \left(u^1, u^2, \ldots, u^{|B|}\right) \in \left(R^{|B|}\right)^*$, where u^k denotes the voltage drop across the kth branch and $\left(R^{|B|}\right)^*$ is the dual space of $R^{|B|}$. Defining the voltage potential v^l for the lth node ($v \in \left(R^{|N|}\right)^*$) the Kirchhoff voltage law takes the form

$$\sum_i d_i^k v^i = d_l^k v^l + d_j^k v^j = u^k \tag{2}$$

for all branches k. A voltage state u is said to be *admissible*, if u satisfies the Kirchhoff voltage law. Eq. (2) shows, that u is admissible, iff $u \in \operatorname{Im} D^*$ with $D^* : \left(R^{|N|}\right)^* \to \left(R^{|B|}\right)^*$ as the dual map of D.

Due to Tellegen's theorem[2, 4], which says that ui is zero, whenever u and i are admissible, a new formulation of the Kirchhoff's laws can be given. Thereby Tellegen's theorem states much more than the fact that the total power of the network is zero. Let Q be a basis of $\operatorname{Ker} D$ and S a basis of $\operatorname{Im} D^*$, then the Kirchhoff's laws are expressed as

$$i = Qx \quad \text{and} \quad uQ = 0 \qquad \text{or} \qquad Si = 0 \quad \text{and} \quad u = yS \tag{3}$$

for some suitable x and y.[11]

It is a well known fact from graph theory, that every connected graph $G = \{N, B\}$ contains a tree $T = \left\{N, \bar{B}\right\}$ with $\bar{B} \subset B$.[1] Consider a connected graph $G = \{N, B\}$ with a tree $T = \left\{N, \bar{B}\right\}$, then Q of eq. (3) follows from the currents of the branches in $B \setminus \bar{B}$ (this formulation will also be used for the network algorithm in section 4) and S can be calculated by means of the voltages of the branches in $\bar{B}$.[11]

2.2 Two–port Description

Fig. 1 shows a two–port with the input port voltage $u_{T,1}$, the output port voltage $u_{T,2}$ and the four terminal currents $i_{T,1}$, $i'_{T,1}$, $i_{T,2}$ and $i'_{T,2}$, which fulfill the Kirchhoff current law $i_{T,1} + i'_{T,1} + i_{T,2} + i'_{T,2} = 0$. Every port must

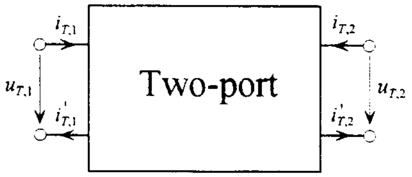

Figure 1: Representation of a two–port.

satisfy the so called port constraints

$$i_{T,1} = i'_{T,1} \qquad \text{and} \qquad i_{T,2} = i'_{T,2},$$

which reduce the number of current variables from three to two. One way to describe the interrelationship among the four terminal variables of a two–port is to write the port voltages in terms of the port currents. In this manner we obtain the impedance matrix $Z = \{Z_{ij}(s)\}$ with

$$\hat{u}_{T,i} = \sum_{j=1,2} Z_{ij}(s)\, \hat{i}_{T,j} \quad , \quad i \in \{1,2\},$$

where the symbol ^ denotes the Laplace transform of the corresponding time signals and s the Laplace variable. The impedances $Z_{ij}(s)$, $i, j \in \{1,2\}$ are defined in the form

$$\begin{aligned} Z_{11}(s) &= \left.\frac{\hat{u}_{T,1}}{\hat{i}_{T,1}}\right|_{\hat{i}_{T,2}=0} , \quad & Z_{21}(s) &= \left.\frac{\hat{u}_{T,2}}{\hat{i}_{T,1}}\right|_{\hat{i}_{T,2}=0} \\ Z_{12}(s) &= \left.\frac{\hat{u}_{T,1}}{\hat{i}_{T,2}}\right|_{\hat{i}_{T,1}=0} , \quad & Z_{22}(s) &= \left.\frac{\hat{u}_{T,2}}{\hat{i}_{T,2}}\right|_{\hat{i}_{T,1}=0} . \end{aligned} \tag{4}$$

Beside the impedance matrix there exist many other possibilities for two–port representation, namely the admittance matrix, two hybrid representations and the transmission matrix. But since all these representations can be easily converted into each other an additional investigation is not necessary.

3 Object Structure of the Terminals in MapleV

Since a computer algebra program like MapleV does not directly support an object oriented design, the *table* environment is used to implement a class concept. Thereby we attached importance to realize some essential features of a class mechanism like they are known from C++[6] or CLOS[12].

An object model describes the structure of an object in a system and contains information about the object name, the object type and the relationship to other objects, the object attributes and data and the methods and operations which can be applied to this object.[9] In MapleV such an object is realized by means of the predefined data type *table*, which has the useful property that its values can be symbolically indexed. The following example illustrates how such an object is created in MapleV.

```
Electric_Object := table ([
   (init) = proc(x: name) global CORRECT;          #initialization
              if type( eval(x), name ) then
                 insert(x, BASIS);
                 x[typ]  := {Electric_Object};     #object type
                 x[chk]  := Electric_Object[chk]; #check operation
              else
                 CORRECT := 1;
                 ERROR(x,` is wrong assigned`)
              fi
            end,
   (chk)  = proc(x: name) true end
]):
```

This Electric_Object serves as a superclass from which we derive all further electric terminals. The initialization program (init) checks if the object name is valid, determines the object type and inserts this object in a globally defined table, called BASIS.

The set of all different objects is arranged in a class hierarchy. The connections between the classes are called inheritance links, because a subclass inherits slots and methods from its superclasses. Fig. 2 illustrates the hierarchy levels of derivation for the electric terminals used in our program. As one can see for special two–port elements the concept of multiple inheritance is used, too.

In the next step we will construct the one–port class. A one–port is initialized by its name and the names of the two end points, the nodes P and M. Thereby a one–port branch is oriented to the node P and off the node M. The function One_Port_Connection checks, if a branch is connected with two nodes and if necessary, inserts the nodes in the global table BASIS.

```
One_Port := table ([
   (init) = proc(x, P, M)                           #initialization
              Electric_Object[init] (x);            #inheritance
              x[typ] := x[typ] union {One_Port};    #object type
              x[p]   := P;                          #node P
              x[m]   := M;                          #node M
```

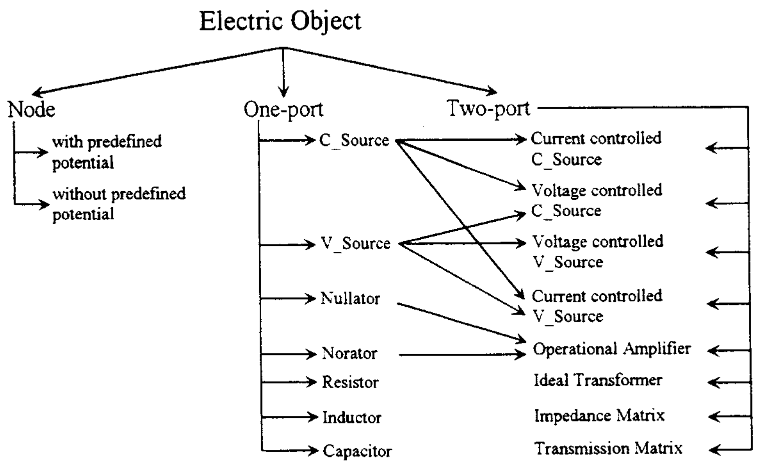

Figure 2: Terminals inheritance graph.

```
             x[con] := One_Port[con];          #connection test
             x[chk] := One_Port[chk];          #check operation
            end,
   (con)  = One_Port_Connection,
   (chk)  = proc (x) true end
]):
```

Now it is rather easy to create new objects without great effort. How this can be done, is demonstrated for a capacitor in the next few lines.

```
Capacitor := table ([
   (init) = proc (x, P, M, c, u0) local x0;       #initialization
             if nargs = 4 then x0 := 0
             else x0 := u0 fi;
             One_Port [init] (x, P, M);           #inheritance
             x[typ] := x [typ] union {Capacitor}; #object type
             x[C]   := convert (c, rational);     #capacity
             x[du]  := x[i]/s/x[C] + x0/s;        #voltage drop
            end,
   (chk)  = proc (x) true end
]):
```

In order to guarantee an easy handling of these objects for the user, a make procedure is implemented in the program. For example a capacitor with the object name $C1$, the nodes $K3$ and $K4$ and the capacity $C1\_value$ is created by calling the procedure

$$\text{Make}(\text{Capacitor}, C1, K3, K4, C1\_value).$$

The make procedure takes its arguments to initialize the object by means of the command line

$$\text{Capacitor[init]}(C1,\ K3,\ K4,\ C1\_value)\ .$$

4 Network Algorithm

In this section we want to present an efficient method for the derivation and solution of the minimal set of network equations.[10] The essential steps of the algorithm are listed below:

1. *Initialization and test phase:*
   - Setting the global and local variables to their initial values.
   - Creation of the nodes of all terminals which are not yet inserted in the global table BASIS.
   - Checking the network graph, if it is connected.
   - Finding all voltage sources that form a loop.
   - Searching the node $K0$ with a predefined voltage potential (e.g. ground). In our case $K0$ is always unique, because we only consider circuits, where the network as a whole satisfies the Kirchhoff current law.
2. *Determination of a tree of the network graph:*

   For this purpose the recursive graph searching algorithm *depth first* is implemented. Thereby the tree must not contain a branch of type current source, controlled current source or norator, because at these terminals the voltage drop cannot be written as a function of the current but it results from the connected network. Moreover if there exists a node that cannot be reached in this manner, the algorithm is terminated.
3. *Calculation of the currents of the tree branches:*

   Starting at the end of the tree it is possible to calculate the currents of the tree branches in terms of the currents of the nontree branches by means of a recursive algorithm.
4. *Calculation of the voltage potential of all nodes:*

   Since the tree was chosen in such a way, that there is a definite interrelationship between the voltage drop and the current at all tree branches the voltage potential of all nodes can be expressed in terms of the currents of the nontree branches. The calculation starts at the node $K0$.

5. *Derivation of the minimal set of network equations:*

   At this step one must distinguish two cases depending on the type of the nontree branches:

   a. If the nontree branches are *not* of type current source, controlled current source or norator, then we obtain an equation according to the fact that the voltage drop of this nontree branch equals the voltage potential difference of the nodes belonging to this branch.

   b. If the nontree branches are of type current source, controlled current source or norator, then additional equations which are induced by this nontree branch must be taken into consideration. So for example an ideal current source will always maintain a specific current I_0 regardless of the voltage drop that may be found across its terminals. In this case the additional equation is trivial and results from the fact, that the current of the current source branch equals I_0 for all times.

   Suppose $G = \{N, B\}$ is a connected network graph with a tree $T = \left\{N, \bar{B}\right\}$, then the minimal number of network equations is $\left|\bar{B}\right|$.

6. *Solution of network equations:*

   The system of linear network equations is symbolically solved by means of internal MapleV procedures.

It turns out, that most of the calculation time is needed for the symbolic solution of the network equations. In a computer algebra program like MapleV an increasing amount of network elements leads to an over-polynomial increase in the calculation time and memory. Thus a network consisting of more than 100 terminals cannot be handled with the so far presented algorithm.

To overcome this restraint we take advantage of the fact, that the calculation time and memory of a symbolic operation depend on the size of data involved in this operation and not primarily on the amount of operations.[3] This is also the reason, why in a computer algebra program the determinant of an $n \times n$ matrix is calculated by means of Cramer's rule, where $(n!)\,n$ symbolic operations are required, and not, as one might expect, with the Gauß elimination, where only n^3 symbolic operations are necessary.

The idea is to split the electric network into several subcircuits, whereas a subcircuit is chosen in such a way, that it can be modeled by a two–port. The network equations are recursively calculated from the top to the innermost subcircuit layer and then the subcircuit solutions are substituted the other way round. In a certain layer a subcircuit is only represented by its two–port parameters, like e.g. the impedances of eq. (4). In order to apply this idea to the network algorithm, the following steps must be additionally executed.

7. *Detection of subcircuits:*

   From the global table BASIS all two–ports of type impedance matrix and transmission matrix must be found out and the relevant port data (node voltage potentials) are stored.

8. *Recursion:*

   Now with the data of the subcircuit we go back to step 1 of our algorithm and determine the solution of the network equations.

9. *Calculation of the substitution table:*

   Using eq. (4) the impedances of the subcircuit are calculated and stored in a substitution table.

5 Applications

5.1 A Simple Example – Active Filter

Fig. 3 illustrates the active filter circuit of investigation. The corresponding

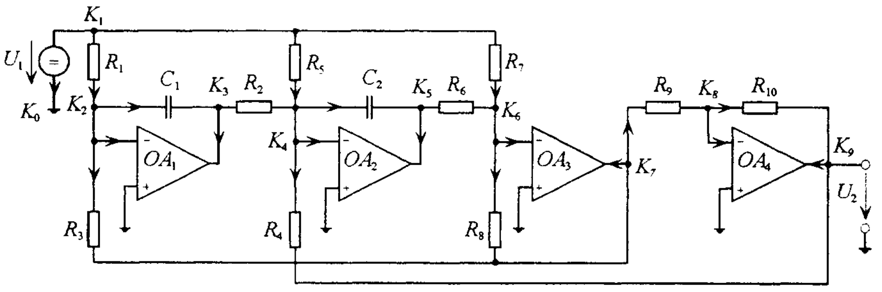

Figure 3: Filter circuit.

input file for the calculation has the form:

```
Make(VSource,U_1, K0, K1, U1);
Make(Resistor,R_1, K1, K2, R1);
Make(Capacitor,C_1, K2, K3, C1);
Make(Opr,OA_1, K2, K0, K3, K0);
Make(Resistor,R_5, K1, K4, R5);
....
```

Now the current of each branch and the voltage potential of each node are available as functions of the voltage source U_1. Thus, the calculation of a transfer function is done with relative ease – so e.g. taking U_2 to be the output of the system the transfer function $G(s) = \frac{U_2}{U_1}$ follows as

$$G(s) = \frac{R_8 R_3 R_{10} \left(R_1 s^2 C_2 R_5 R_6 R_2 C_1 - R_1 R_7 R_2 s C_1 + R_7 R_5\right) R_4}{R_1 R_7 R_5 \left(R_9 R_4 C_2 R_2 s^2 C_1 R_6 R_3 + R_2 s C_1 R_{10} R_8 R_3 + R_9 R_4 R_8\right)} .$$

5.2 Thermal Model for an Injection Moulding Machine

It is a well known fact from literature[8], that the thermal behavior of a system with negligible heat radiation can be modeled by a linear equivalent thermal network. Thereby the thermal impedances depend on the component geometry and material data, whereas also the case of transient heat conduction is considered by means of thermal capacities. A general cylindrical lumped component and its corresponding thermal network is shown in fig. (4) with the thermal impedances R_1, R_2, R_3 and R_m, the thermal capacity C, the heat source P, the temperature of the surroundings θ_u and $\theta(x)$ denotes the temperature at the point x. For the necessary assumptions and formulas we refer to the respective literature[5, 8].

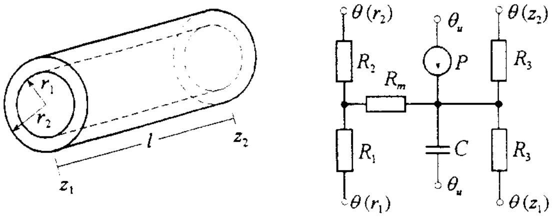

Figure 4: General cylinder component and equivalent thermal network.

This modeling technique was tested on an injection moulding machine. The geometry of the machine was reduced to about 10 cylinder components given in fig. (4). All these components together with the additional thermal impedances for the heat convection lead to an interconnected thermal network with about 60 elements. A symbolic calculation of this network on a 486AT/100MHz with 32MB RAM is only possible using the subcircuit technique discussed in section 4. Using 8 subcircuits the calculation takes 2 minutes.

This modeling technique is an alternative to conventional identification methods. Moreover the great advantage of the symbolic description is, that one can easily investigate the effects of certain system parameters (like in our case e.g. the thermal conductivity) on the system outputs. Comparisons between measurement results of the injection moulding machine and the model output show a very good correspondence.

References

1. Christofides, N. *Graph Theory*, Academic Press, 1975.
2. Chua, L.O., Desoer, C.A. & Kuh, E.S. *Linear and Nonlinear Circuits*, McGraw–Hill, 1987.
3. Davenport, H., Siret, Y. & Tournier, E. *Computer Algebra*, Academic Press, 1988.
4. Hirsch, M. & Smale, S. *Differential Equations, Dynamical Systems and Linear Algebra*, Academic Press, 1974.
5. Kaltenbacher, M. & Saari, J. An asymmetric thermal model for totally enclosed fan–cooled induction motors, *Report 38 of the Helsinki University of Technology*, Espoo, Finland, 1992.
6. Lippmann, B. St. *C++ Primer*, Addison Wesley, 1992.
7. MapleV. *Reference Manual*, Springer Verlag, 1994.
8. Mellor, P., Roberts, D. & Turner, D. Lumped Parameter Thermal Model for Electrical Machines of TEFC Design, *IEE Proceedings–B*, Vol.138, No.5, pp.205–218, 1991.
9. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. & Lorensen, W. *Object-Oriented Modeling and Design*, Prentice Hall, 1991.
10. Schlacher, K. & Kugi, A. Mathematical modeling and computational principles for the analysis and simulation of long–distance energy systems, *Mathematics and Computers in Simulation*, Elsevier Science B.V., Vol.39, pp.565–572, 1995.
11. Schlacher, K. & Scheidl, R. Modeling of Mechatronic Systems by Symbolic Computation, *Proceedings of the EUROSIM 95*, edited by Breitenecker, F. & Husinsky, I., Vienna, Elsevier Science B.V., pp.657–662, 1995.
12. Winston, P. H. & Horn, B. K. P. *LISP*, Addison Wesley, 1989.