UNIVERSITÀ DEGLI STUDI DI BOLOGNA
Facoltà di Ingegneria Elettrica

Corso di Dottorato di Ricerca in Ingegneria Elettrotecnica XIII ciclo

SYMBOLIC TECHNIQUES ADDRESSED TO ELECTRIC CIRCUIT ANALYSIS AND DIAGNOSIS

Tesi di Dottorato di Marcello Artioli

AA 1999-2000

Il Coordinatore	Il Relatore
Prof. F. Negrini	Prof. F. Filippetti

Foreword

This work

1 Aim of the work

The aims of this work are basically:

1. to state the possible contribution of the *Symbolic Calculus* to two of the problems of electrical engineering, that are:
    - analysis of electric circuits;
    - diagnosis of electric circuits;
2. to study a method and an implementation of the suggested contribution.

2 Structure of the work

Accordingly to the aim of the work, the matter exposition has been structured to outline first a general frame about Symbolic Calculus and then to give more details on particular contributions about the mentioned problems.

2.1 Chapter I

A definition of Symbolic Calculus is given, and the state-of-the-art of this engineering approach is presented with its advantages and drawbacks, to identify some fields of interest for the electrical engineering.

2.2 Chapter II

2.2.1 Part One

The new *symbolic formulation* of the known Inhibition Method can avoid or reduce the typical limitations which the Symbolic Calculus suffers from, due to its intrinsic nature. This method offers some other interesting features as well.

2.2.2 Part Two

The symbolic formulation of this method allows several software implementations of a *symbolic processor* oriented to wide linear circuits. The general symbolic formulation and the kernels of the implementations are presented to be a general method that has reflections on circuit diagnosis and analysis.

2.3 Chapter III

2.3.1 Part One

The symbolic approach and an efficient symbolic processor, based on the Inhibition Method, lead to an original contribution in linear electric circuit diagnosis: the *Cycling Verify Method*. It is a symbolic technique to locate and identify faults in linear circuits. The embedded *interval algebra* at the symbolic formulation level allows to take into account parameter tolerances and uncertainties, in a transparent way.

2.3.2 Part Two

Some other remarks are reported about the Inhibition Method and diagnostic techniques, with particular reference to a proposed improvement to multifrequency based techniques.

2.4 Chapter IV

The same symbolic approach and processor lead to contributions to circuit analysis as well.

2.4.1 Part One

An improvement of the *Threshold Technique*, dealing with *switching circuits*, introduces the event-driven principle and a calculation procedure starting from the general symbolic solution. This fact avoids some known problems, like convergence.

2.4.2 Part Two

An original contribution to resistive *piecewise-linear circuit* analysis consists in an elegant symbolic approach to this kind of non-linearity. Once this non-linearity is

(in some way) “hidden” to the symbolic processor, the formulation and the solving process become purely linear.

2.5 Chapter V

A software environment for *virtual instruments* has been investigated to study the possibility to integrate diagnostic procedures and symbolic algorithms in a user-friendly tool for electrical system analysis and diagnosis.

2.6 Conclusions and references

Conclusions are reported at the end of each chapter or at the end of each part of the chapter, if it is subdivided in different main sections: each section has its own conclusion paragraph.
References are organized in the same way. In each block, numbering restarts from “1” to make easier reading.

Chapter I

Goals and views

1 Symbolic calculus

The key sentences that define the Symbolic Calculus (SC) are basically:

- literal calculus;
- operations on formal relationship (equations, inequations, rules, sets, etc.);
- integration of an advanced programming language.

It is clear that SC is, but it is not limited to, literal calculus. It works with abstract quantities and their formal relationships to extend the capabilities of the Numeric Calculus (NC). A numeric program normally takes specified numbers as input and yields numbers as output again, referring to the prefixed unknown quantities. Of course, this numeric output can further collected to produce graphs, for example, or some other kind of report, to have information about quantity courses or about correlations between quantities. But in any case it is a matter of numeric estimation of relations and not of analytical relations. Therefore SC is told to be more general: it can produce the same results, because analytical expressions can be numerically evaluated to produce numbers, but it adds the explicit information about quantity correlations.
Furthermore, SC integrates a programming language with advanced functions not only to calculate or to solve problems, but to manipulate the resulting expressions and formulas, like factorization, coefficient extraction, integration and many other conceptual operation that the user normally does by the abstract and mathematical thinking. In this manner, SC brings the problem formulation near to the natural way of thinking of the user and thus it is a convenient and reliable programming approach to get general solutions. As a matter of fact, a simulation that needs to be repeated many times, can be easily programmed and performed once and for all in symbolic way while the resulting symbolic solution can be further repeatedly evaluated with different parameter values, with a minimal workload.
So, it is possible to summarize the importance of SC [1] mainly in:

- problem formulation in a natural way of thinking;
- extraction of relationship between parameters;
- solution in general terms.

SC has also drawbacks, of course, and thus the state of the art comprises a variety of proposals, depending on the aim and the optimization needed, as can be understood in the next sections.

2 Goals

Analysis and diagnosis of electric circuits make often provision of repeated simulations and calculations that are formally identical but which are performed by different situations of rated values. A general symbolic solution, repeatedly evaluated in numeric terms seems well to be complementary to existing analysis and diagnosis techniques. Examples could be:

- fault dictionary generation: the same network function is measured and plotted when one ore more parameters are varying in a given range;
- sensitivity calculations: basically, the same network function has to be derived with respect to several parameters;
- research of parameter correlations: if an exact relation is needed, then it is the right task for SC; if an approximated relation is needed, then a symbolic expression could be either symbolically simplified or numerically evaluated and then approximated.

Moreover, considering that modern environment for SC are capable of sophisticated manipulations of literal expressions and rules, and that they can connect to other software applications, the direct symbolic formulation of a problem (in terms of abstract quantities, formal equations and inequations, rules, etc.) seems to allow original procedures that are not possible if only numerical.
So the goals of this work are mainly set on three things:

- verify the benefit of SC in electric circuit analysis and diagnosis;
- use of SC to improve some existing technique;
- original contribution of SC to new techniques.

These aims have to be pursued taking into accounts the following drawbacks.

3 Problems

SC environments are complex software packages and their main common problems [2] are:

- memory consumption;
- unmanageability of long expressions.

The internal data structures of symbolic expressions, in fact, are basically character strings of variable length and various format. In a word: symbolic

processors work with unstructured data. The main drawback is not the increased CPU time cost to deal with unstructured rather than structured data, but is the fact that during the generation of a resulting symbolic expression these data proliferate exponentially.
Checking Fig. 1, it is easy to understand that to perform a numerical evaluation of a sum of *n* numbers we need memory space for *n+1* number containers: the first *n* for the given numbers and one that will contain, time after time, the sum of the current addendum and the sum of the preceding numbers. This last container has the same length and memory occupation of the other ones. On the other hand the symbolic evaluation of a sum of *n* expressions still needs *n+1* containers (which are themselves memory wider than numbers, normally), but if no cancellation occurs between the input expressions, the result container will carry an expression which will be long at least as the sum of the lengths of the input expression (without counting control and formatting data).

3, 2, 6	$\frac{1}{sC_1}, R, \frac{1}{sC_2}$
$p1 = 0 + 3 = 3$	$p1 = 0 + \frac{1}{sC_1} = \frac{1}{sC_1}$
$p2 = p1 + 2 = 5$	$p2 = p1 + R = \frac{1 + C_1 Rs}{sC_1}$
$p3 = p2 + 6 = 11$	$p3 = p2 + \frac{1}{sC_2} = \frac{1 + C_1 + C_2 + C_1 C_2 Rs}{sC_1 C_2}$

Fig. 1: Proliferation of symbolic data.

Thus, the main drawback of SC is memory consumption. This fact bounds enormously the dimension of the problem (in our case, of the circuit) to be solved.
Over this disadvantage, that relies on hardware resources, there is a second one that relies on the user side. In fact, complete symbolic expressions for wide circuits are very long, no matter where they come from, obtained by hand or from the symbolic processor: a network function for a hundred-parameter circuit could not fit on a page. Symbolic results are often unusable for a visual inspection by the operator, due to their complexity and unreadability. This is a great penalty, for example, during a design process, where the last tuning of a circuit is frequently delegated to the user and not to an automatic procedure.

There are remedies, of course, to these drawbacks, but they are often incompatible with other needs, as discussed in the next section.

4 Solutions

4.1 Decreasing the number of symbols

The first, obvious, remedy to both memory consumption and user-unreadability is to reduce the number of symbols in the expressions, that is, only the most interesting parameters are left literal, while the other ones are set to their numerical values. In this way, the input expressions are more compact, the program execution is faster, and the final result is not so complicated and more readable. These things are paid, of course, by a loss of information and generality about the solution.

4.2 Symbolic approximation of expressions

The symbolic approximation consists in generating simplified non-exact expressions, maintaining all parameters literal. Relations between parameters are simplified accordingly to particular neglecting criteria. This approach shares the same advantages and disadvantages of all approximated methods, but it offers the possibility to get a complete symbolic solution, easy to be read and manipulated by the user, and more compact to be stored in memory. This feature is still paid by a loss of information (which could be not so irrelevant) plus an increased calculation time, due to the simplification procedures.

4.3 Functional/mathematical disaggregating of the circuit

A large electric network leads, during the solution generation, to very large and complex symbolic expressions, which are difficult (when not impossible) to calculate due to memory constraints.
If the network become decomposed in several smaller networks, then each sub-network can be easily solved in turn by the symbolic processor, that could rebuild the whole general solution from the partial ones, avoiding the simultaneous generation of a great part of memory wasting expressions. The decomposition can be done in terms of functional blocks or following abstract mathematical rules.
The first case is very useful when the circuit input data are given as network layout, that is when the circuit description is near its physical realization: it is also possible to generate a library of symbolic expressions related to the most common functional blocks, in order to re-use previous partial calculation to save memory and CPU time. This is possible due to the generality of the symbolic solution, no matter the parameter values are, which constitutes a kind of abstract model of the block.

The second case operates directly on the circuit description in terms of equations. This allows a great variety of transformations and manipulations to get reduced equations, easier to solve. Of course, all partial solutions have to be anyhow recollected to rebuild the general solution.
It is clear, that this approach lowers memory consumption but rises time consumption, due to the decomposition-recollection process.

4.4 Spreading a single expression over a sub-sequence of shorter expressions

During the generation of the solution expression, partial expression are not colleted and algebraically compound in a new symbolic expression, but they are labeled and organized (following specified criteria) in a kind of database; the new expression is given as relationship between sub-expression, referred as place-holder label. The resulting expression is very compact, and when the sub-expressions are referring to well identified circuit blocks, it can be more easily used in the design phase, for example.
This remedy is similar to the "circuit decomposition", but it differs from it basically in the fact that the final symbolic solution is not immediately and explicitly calculated, while partial or intermediate calculations are saved on mass storage media, and then retrieved when needed.

5 State-of-the-art

Nowadays there are some commercial and non-commercial symbolic processors oriented to electric circuits and they are collected with their main features in Table 1, taken from the work of Fernàndez and Rodrìguez-Vàzquez [2]. This table highlights some relevant aspects about symbolic processors:

- only one is capable of hierarchical analysis by disaggregating circuits;
- very few have capabilities of managing general expressions (they are often limited to s-domain calculations) or capabilities of further symbolic computations (like pole extraction, sensitivity, etc.);
- almost all are stand-alone applications written in a low level language, not a complete symbolic environment;
- they normally need high-level hardware resources (workstations).

Table 1: State-of-the-art for symbolic processors.

	SCAPP	SCYMBAL /SYBILIN	SC	SAPEC	SSPICE	SYNAP	ISAAC	ASAP	SIFTER	RAINIER
Formulation	RMNA & SFG	SFG	MNA	MNA	MNA	MNA	CMNA	SFG	nodal analysis	tree enum.
Analysis domain	s	z / s	s	s	s	s & z	s & z	s	s	s
SAG (expanded format)	no	no	no	no	yes	yes	yes	yes	no	no
SAG (nested format)	no	no	no	no	no	yes	no	yes	no	no
SBG	no	no	no	no	no	no	no	no	yes	yes
SDG	no	no	no	no	yes	no	yes	yes	no	yes
Mismatchings	no	no	no	no	no	yes	yes	yes	no	no
Element lumping	no	no	no	no	no	no	yes	yes	no	no
Nonlinear analysis	no	no	no	no	no	no	weakly nonlin.	no	no	no
Hierarchical analysis	yes	no	no	no	no	no	no	no	no	no
P/Z extraction	no	no	no	no	no	no	no	yes	yes	no
Graphical interface	no	no	yes	yes	no	no	no	yes	no	no
Platforms (WS = workstations)	WS	WS	WS	PC	WS & PC	WS	WS	WS & PC	WS	WS
Implementation language	C	FORTRAN /ADA	C	LISP/ C++	C	LISP/ C++	LISP/ C	C/C++	C	C

6 Synthesis prospect

Trying to overcome the typical limitations of SC, a symbolic processor should hopefully have this features:

- ability for wide circuits, even if running on standard hardware (PCs);
- good compromise approximation-manageability of the symbolic solution;
- tools for extended expression manipulation (to perform other tasks than solving).

Now, it's clear that there is a definite need for a method that should allow SC for wide circuits, but to date, there is not a method without drawbacks. The best proposals are combinations of the previous discussed solutions, depending on the optimization requested. If the generation of an exact symbolic solution, without approximation, is "a must", then the only way seems to be the renounce to the visual inspection of the formulas. This disadvantage can be mitigated if the method is well integrated in a complete symbolic environment, equipped with

advanced functions for the symbolic-algebraic manipulation, like integration, derivation, transformation, plotting, etc.
Thus, a synthesis prospect on symbolic processors devoted to electric circuits recalls the search about a method that has to be:

(1) intrinsically *memory sparing*
(2) and easily implementable in a *symbolic programming environment*.

In a strange manner, an old numeric method seems to be a good candidate, as outlined in the next chapter.

7 References

[1] R. Germundsson. Viewpoint: THE IMPORTANCE OF SYMBOLIC PROGRAMMING. *IEEE Spectrum 1999*.

[2] F. V. Fernàndez, A. Rodrìguez-Vàzquez: SYMBOLIC ANALYSIS TOOLS - THE STATE-OF-THE-ART. *IEEE 1996*.

Chapter II

The old Inhibition Method as new symbolic processor

1 Part One - Formulation

Part One is intended to show a general method that relies on the superposition principle to analyse linear systems and electric circuits, based on hierarchical sequences of more simple topologies with some inhibited elements and on a short recollecting logic to calculate the final solution from the previous partial and more elementary solutions.
In Section 1.1, 1.2 and 1.3, will be mentioned the origin of the method, why it is general and some its global features.
In Section 1.4 will be exposed the non-disciplinary theory of the method. This originally anonymous theoretical formulation can be particularised to other field of interest specific formulations, and due to its intrinsic symbolic constitution this method can be directly used to get symbolic solutions from the system which is being applied to, or even be re-formulated in different algorithmic ways to optimise some aspects like memory or time consumption, appearing to be a good candidate for doing SC.
Implementations of this method will be discussed in Part Two.

1.1 Brief history and overview of the method

The Inhibition Method (IME) is an iterative exact non-inverting method to solve any linear system, derived from the Cross method, which is, on the contrary, a non-exact method and from which IME inherits the fundamental characteristic of decomposing the problem into sub-problems easier to solve. It produces a hierarchical sequence of sub-systems [1], at the end of which only elementary

systems can be found. Therefore, they can fast be solved with the minimum knowledge, that is, few program code lines are needed.
The problem splitting process is founded, in fact, on the possibility (see Fig. 2) to deduce the properties of the system (a) from two more simple sub-systems (b) and (c), obtained from (a) suppressing or inhibiting a component.

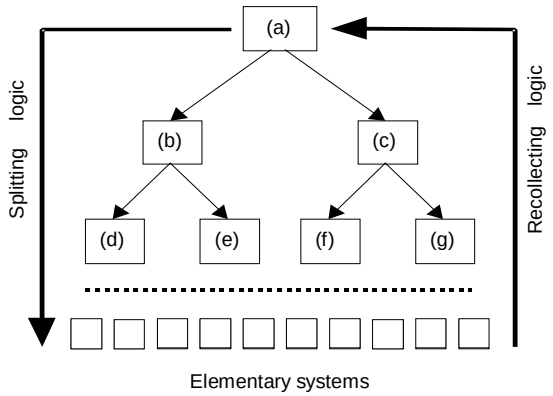


Fig. 2: IME logic.

The same idea can work on the sub-systems (b) and (c), that can be deduced each from an other couple, respectively (d-e) and (f-g), suppressing other components, and so on. At the end of the splitting process we come to sub-systems so simple that they can be solved immediately. For example a linear electric network can be repeatedly split into more simple networks with an inhibited branch until they become one-loop circuits. A similar ground basis is used in [2].
The recollecting logic, which will be demonstrated later in detail, is very simple as well. As a matter of fact, if a variable (a) consists of the sum of other three variables (b), (c) and (x),

$$a = b + c + x\,,$$

where (x) is proportional to (a) through a constant (k),

$$x = k \cdot a\,,$$

it follows that

(Eq. 1) $$a = \frac{b + c}{1 - k}\,,$$

which is the symbolic formula that synthesises the link between a generic system (a) and the two sub-systems (b) and (c) derived from it. The demonstration is not difficult as well, because it is organised following the “Short Didactics” (SD) criteria [3]: the unifying element of several steps of the proof is put in evidence

and collected to shorten and clarify the proof itself, realising a kind of “internal contraction” [4].
The first IME version [5] was only for numeric calculations and it was based on a laborious algorithm which had been working fine because it allowed to go round the heavy memory constraints for huge systems to solve. Here it is presented a new version of the method that really becomes a general method with the use of modern software tools.

1.2 Weakness and strength

From the previous overview of the method, it comes out its intrinsic nature, which carries advantages and disadvantages, of course.
Disadvantages are that:

- the basic algorithm is laborious;
- from the numeric computational point of view is slow.

These are the main reasons because this method is no more used today.
But, as explained in the previous chapter, it has advantages that could contribute to limit common problems in SC, like:

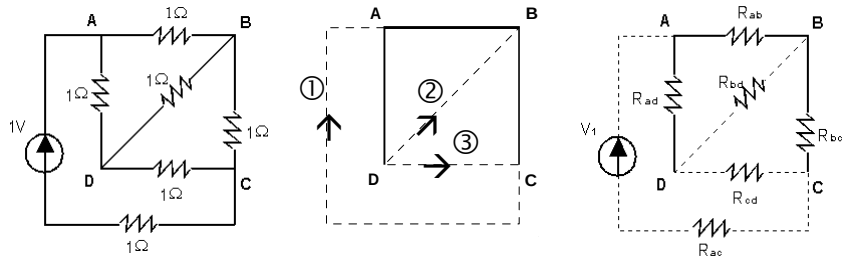
- *disaggregating the circuit*, as can be seen in Fig. 2 and in the introductory example of Section 1.3 below;
- *disaggregating expressions*, as can be understood considering that the very simple formula of (Eq. 1) is repeated at every iteration instead of a single complex expression;
- *minimization of memory consumption*, due to disaggregating-collecting logic which works on few elements at a time.

Furthermore it has not to be underrated that it is a general method for several reasons. As a matter of fact, it can be applied in every field if the system to solve is linear (or made linear). The following anonymous formulation emphasizes this fact, because there is no reference to the physical nature of the system, thus, “anonymous” stands for inter-disciplinary and indicates a general method all over the linear world.
From this formulation, discipline-specific methods can be derived simply by carrying out the related rules, intrinsic to that particular discipline, as could be seen in one of the next sections, where the electric formulation is given. Adding to this some specific knowledge, like the modified nodal analysis, a complete software tool can be realized for electric circuits.
Symbolic programming is a quite recent phase in the evolution of programming but has now reached relevance and importance in engineering [6,7,8,9,10,11]. Other advantages of the suggested contribution relies, in fact, on the advanced features of commercial symbolic processors, like Mathematica™, which are able to widely manipulate expressions, to draw any kind of graphics and to automatically generate program source code.

Due to the presence of such symbolic processors, which allow to manage more abstract quantities other than only numbers, the anonymous formulation can be directly transferred to a computer program to make a general tool for linear system solving and, last but not least, to solve it in a complete symbolic form, giving a more general character to the solution found, leaving away the pure numerical computation field.

1.3 An introductory example

The anonymous formulation is not too complicate, but it could be a little hard to understand because it is really abstract. An introductory example should show the method basics in a practical way. Let's consider the very simple resistive circuit Fig. 3. This is a 3-loop circuit: a possible tree could be that one depicted Fig. 3b with numbered main loops.



(a): resistive circuit (b): tree, co-tree, loops (c): symbolic names

Fig. 3: Sample circuit.

IME performs a decomposition of the starting circuit into three elementary (one-loop) circuits corresponding to the main loops, shown in Fig. 4, where each loop is supplied with an unitary voltage source in the co-tree branch, regardless of the originary supply value. These three elementary circuits are first solved independently as reported (symbolically) in Fig. 5, and then the procedure continues rebuilding the complete solution of the starting circuit, following a recollecting logic. This logic uses "special" quantities to link and deduce the general solution from the different elementary ones: *inhibition quantities*. In this case, as depicted in Fig. 6 with slashed greyed symbols, inhibition quantities are voltage sources. Their values have to be calculated to get just zero current in the branch (seat) where they are placed, in order to render the starting circuit a virtual one-loop circuit, that is, a three-loop circuit with two co-tree branches inhibited (Fig. 4 and Fig. 6 compared). This is called *inhibition level 3*. The next generation of circuits (*inhibition level 2*) basically consists of the same three-loop circuit with only one co-tree branch inhibited, as shown in Fig. 7. The solution of these circuits is deduced from level 3 via *Inhibition Theorem* (see previous Section 1.1 or following Section 1.4.3). In the same way, level 1, which is the final level, with no inhibited branches, can be deduced from level 2, giving the final solution for the starting circuit of Fig. 3.

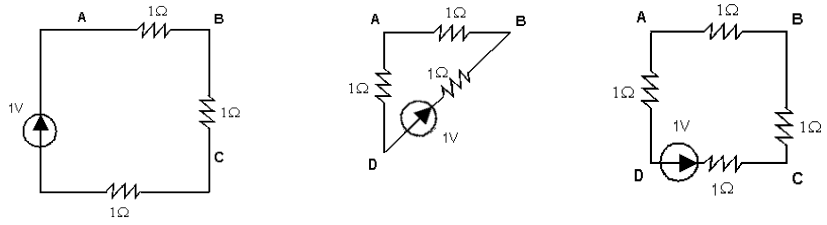


Fig. 4: Elementary one-loop circuits from the sample circuit.

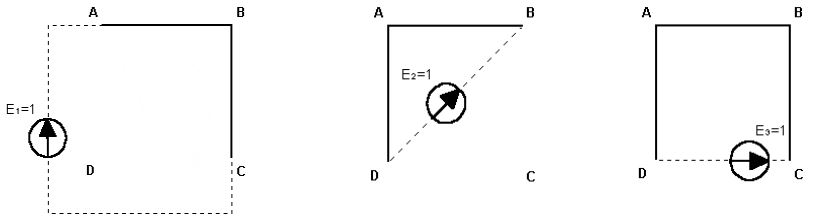
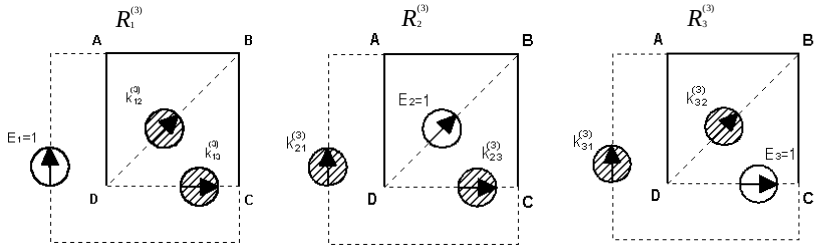


Fig. 5: Simplified schemes for the elementary one-loop circuits.



$[I_1]_1^{(3)} = \frac{1}{3}$

$k_{12}^{(3)} = -R_{ab} \cdot [I_1]_1^{(3)} = -\frac{1}{3}$

$k_{23}^{(3)} = -(R_{ab} + R_{bc}) \cdot [I_1]_1^{(3)} = -\frac{2}{3}$

$[I_2]_2^{(3)} = \frac{1}{3}$

$k_{21}^{(3)} = -R_{ab} \cdot [I_2]_2^{(3)} = -\frac{1}{3}$

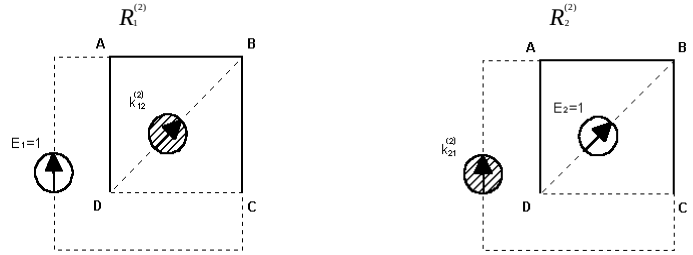
$k_{23}^{(3)} = (R_{ab} + R_{ad}) \cdot [I_2]_2^{(3)} = \frac{2}{3}$

$[I_3]_3^{(3)} = \frac{1}{4}$

$k_{31}^{(3)} = -(R_{ab} + R_{bc}) \cdot [I_3]_3^{(3)} = -\frac{1}{2}$

$k_{32}^{(3)} = -(R_{ab} + R_{ad}) \cdot [I_3]_3^{(3)} = \frac{1}{2}$

Fig. 6: IME elementary regimes of the sample circuit (inhibition level 3).



$k_{12}^{(2)} = \left[k_{12}^{(3)} - k_{13}^{(3)} \cdot k_{32}^{(3)}\right] \cdot \alpha_1^{(3)}$

$k_{21}^{(2)} = \left[k_{21}^{(3)} - k_{13}^{(3)} \cdot k_{31}^{(3)}\right] \cdot \alpha_2^{(3)}$

Fig. 7: Hierarchical deduction of circuits “less inhibited” (inhibition level 2).

1.4 Anonymous (symbolic) formulation

IME can be used in solving any physical linear system. Here we'll talk about linear systems in general, because the intention is to give a more abstract formulation of the method, with no reference to real systems, from which we get the notion of "cause", "effect" and "seat". The word "seat" will indicate the place where some causes act to produce some effects, in general. In Table 2 and Table 3 it is reported the glossary needed to understand the formulation. In many cases, the word or the definition used in the anonymous formulation is followed by the correspondent one used in the electrical formulation, both in terms of branch current and node voltage. In this manner it is possible to figure out the abstract formulation with the help of suitable entities taken from the more usual electrical topics.
To get a more compact dissertation some other symbolic representations are given in Table 4, understand all figures in the following sections.

Table 2: Fundamental glossary.

	Seat	Cause	Effect	Supplied seat	Free seat	Inhibited seat
Anonymous	j	Y_j	X_j	$Y_j \neq 0$	$Y_j = 0$	$Y_j = k \rightarrow X_j = 0$
Electrical (branch current)	j branch	E_j voltage source	I_j branch current	$E_j \neq 0$	$E_j = 0$	$E_j = k \rightarrow I_j = 0$
Electrical (node voltage)	j node	I_j current source	V_j node voltage	$I_j \neq 0$	$I_j = 0$	$I_j = k \rightarrow V_j = 0$
Comment	j = 1, 2, .. n, where n is the total number of seats	The cause in the seat j	The effect in the seat j	A seat is *supplied* when both cause and effect are present on it.	A seat is *free* when the cause is not present but the corresp. effect is.	A seat is *inhibited* when a quantity, dimensionally homogeneous with the causes, is present and set to a value so that it cancels the effect in the seat .

Table 3: Fundamental definitions.

Name and symbol	Anonymous	Electrical (branch current)	Electrical (node voltage)
Regime $R(\cdots)$	State of the system individuated by the indication of the (given) causes, supplied seats, free seats and inhibited seats.	State of the circuit individuated by the set of currents and voltages	
Multi-supplied regime	Regime with more than one supplied seat.	(State of the) circuit with more than one branch having a voltage source	(State of the) circuit with more than one node having a current source
Single-supplied regime	Regime with only one supplied seat.	(State of the) circuit with only one branch having a voltage source	(State of the) circuit with only one node having a current source
Unitary regime	Single-supplied regime where the supplying cause has unitary value.	(State of the) circuit with only one branch having a voltage source set to "one"	(State of the) circuit with only one node having a current source set to "one"
Principal regime of order *v* $R_j^{(v)}$	Unitary regime supplied in the *j-th* seat, whose free seats are all and only the seats with index greater than *v*. This means that the first *v* seats are all inhibited but one, which is unitary supplied.		
Inhibition quantity $k_{jl}^{(v)}$	Considering a principal regime of order *v* supplied in *j*, it is the quantity, in the seat *l*, dimensionally homogeneous with the causes set to a value so that it cancels the effect in the same seat (the effect is not present) (*).	.. it is the voltage source in the branch *l*, set to the value so that the current is zero in the branch *l*.	.. it is the current source in the node *l*, set to the value so that the node voltage is zero.
Inhibition level of order v	The set of all principal regimes of order *v*.		
Inhibition sequence	The sorted set of all inhibition levels, sorted by decreasing values of the order *v*.		

(*) This definition is tautological with respect to the inhibited seat, but it is given anyway to underline that the inhibition quantity itself is an effect as well, even if it causes the principal effect cancellation.

Table 4: Useful representations utilized in the anonymous formulation.

Generic quantity	$Q_j^{(v)}$	It is the value assumed by the quantity *Q* in the regime of order *v*, supplied in the *j-th* seat. It follows the same notation of the regime, as shown in Table 3.
Unitary regime	1 | 1 2 | $k_{12}^{(2)}$ 3 | | $R_1^{(2)}$	A regime is depicted by a table: the last row shows the regime name; the other (numbered) rows indicate the seats; inhibited seats have greyed background; free seats are empty and supplied seats have supply value reported.
Non-unitary regime	1 | 5 2 | $5\,k_{12}^{(2)}$ 3 | | $5\,R_1^{(2)}$	If, for the previous regime, the supplying value is not set to 1, let's say 5, and then it will be depicted by the table on the left, due to the linearity between causes and effects (it is a multiple of an unitary regime).

Level 3	Q	Seat	$k_{\_1}$	$k_{\_2}$	$k_{\_3}$
$R_1^{(3)}$	$Q_1^{(3)}$	1		$k_{12}^{(3)}$	$k_{13}^{(3)}$
$R_2^{(3)}$	$Q_2^{(3)}$	2	$k_{21}^{(3)}$		$k_{23}^{(3)}$
$R_3^{(3)}$	$Q_3^{(3)}$	3	$k_{31}^{(3)}$	$k_{32}^{(3)}$	
Level 2					
$R_1^{(2)}$	$Q_1^{(2)}$	1		$k_{12}^{(2)}$	
$R_2^{(2)}$	$Q_2^{(2)}$	2	$k_{21}^{(2)}$		
Level 1					
$R_1^{(1)}$	$Q_1^{(1)}$	1			

Fig. 8: Sample inhibition sequence tableau with symbols.

1.4.1 Inhibition sequence tableau

This tableau gives a complete view of all principal regimes of the same inhibition sequence and of all effect values in the respective regimes:

- in the first column are indicated all principal regime names, grouped by decreasing inhibition order;
- the column headed with "Seat" reports the supplied seat index for more readability;
- the columns between regime names and seat index contain effect values;
- the columns on the right of the seat index column contain inhibition quantities.

In Fig. 8 it's reported a tableau for a 3-seat system with only one effect quantity.

1.4.2 ICS operator

ICS stands for Inhibition-Compensation-Separation and indicates a particular kind of decomposition from a given unitary regime into other two, due to the superposition principle.

Let the given unitary regime be $R_j^{\cdot\cdot}$, where "·" has to be understood as non specified inhibited seats.

The first two steps of the ICS operator, applied to a generic free seat *m* are equivalent to add and subtract the correspondent inhibition quantity $k_{jm}^{\cdot\cdot m}$ in the same seat (Fig. 9). The third step consists in separating the two quantities in two different regimes (Fig. 10). The inhibition quantity $k_{jm}^{\cdot\cdot m}$ is left in the original regime (which now becomes $R_j^{\cdot\cdot m}$) and its opposite $-k_{jm}^{\cdot\cdot m}$ now acts as supplying quantity in the seat *m* of a second regime that, superimposed to the first, gives back the original one.

	0			0			0			0			0			0	
j	1		j	1		j	1		j	1		j	0				
m	0		m	$k_{jm}^{\cdot\cdot m} - k_{jm}^{\cdot\cdot m}$		m	0		m			m	$-k_{jm}^{\cdot\cdot m}$				
	0			0			0			0			0				
	$R_j^{\cdot\cdot}$	=		$R_j^{\cdot\cdot}$			$R_j^{\cdot\cdot}$	=		$R_j^{\cdot\cdot m}$	+		$-k_{jm}^{\cdot\cdot m} \cdot R_m^{\cdot\cdot}$				

Fig. 9: Adding and subtracting the same inhibition quantity in the seat *m*.

Fig. 10: Separating the two compensating regimes.

1.4.3 Inhibition theorem

1.4.3.1 Proof

It is derived from the relationship of Fig. 10 applying the ICS operator again on the last regime. If we inhibit its seat *j* by adding the inhibition quantity $-k_{jm}^{\cdot\cdot m} \cdot k_{mj}^{\cdot\cdot j}$ (remembering that this regime is a multiple of $R_m^{\cdot\cdot}$ through the factor $-k_{jm}^{\cdot\cdot m}$) and if we compensate it with $k_{jm}^{\cdot\cdot m} \cdot k_{mj}^{\cdot\cdot j}$, which will be then separated and made acting as supply cause in the next regime $k_{jm}^{\cdot\cdot m} \cdot k_{mj}^{\cdot\cdot j} \cdot R_j^{\cdot\cdot}$, we will get the symbolic equation:

$$R_j^{\cdot\cdot} = R_j^{\cdot\cdot m} - k_{jm}^{\cdot\cdot m} \cdot R_m^{\cdot\cdot j} + k_{jm}^{\cdot\cdot m} \cdot k_{mj}^{\cdot\cdot j} \cdot R_j^{\cdot\cdot}.$$

The superposition principle allow to arrange this symbolic equation as any other equation, and thus:

$$R_j^{\cdot\cdot} - k_{jm}^{\cdot\cdot m} \cdot k_{mj}^{\cdot\cdot j} \cdot R_j^{\cdot\cdot} = R_j^{\cdot\cdot m} - k_{jm}^{\cdot\cdot m} \cdot R_m^{\cdot\cdot j},$$

collecting the common factor:

$$(1 - k_{jm}^{\cdot\cdot m} \cdot k_{mj}^{\cdot\cdot j}) \cdot R_j^{\cdot\cdot} = R_j^{\cdot\cdot m} - k_{jm}^{\cdot\cdot m} \cdot R_m^{\cdot\cdot j},$$

and at the end:

(Eq. 2) $$R_j^{\cdot\cdot} = (R_j^{\cdot\cdot m} - k_{jm}^{\cdot\cdot m} \cdot R_m^{\cdot\cdot j}) \frac{1}{1 - k_{jm}^{\cdot\cdot m} \cdot k_{mj}^{\cdot\cdot j}}$$

as reported in Fig. 11, with the obvious definition of $\alpha_{jm}^{\cdot\cdot}$.

	0			0			0			0							
j	1		j	1		j	0		j	$-k_{jm}^{\cdot\cdot m} \cdot k_{mj}^{\cdot\cdot j}$							
m	0		m			m	$-k_{jm}^{\cdot\cdot m}$		m	0							
	0			0			0			0							
	$R_j^{\cdot\cdot}$	=		$R_j^{\cdot\cdot m}$	+		$-k_{jm}^{\cdot\cdot m} \cdot R_m^{\cdot\cdot}$	+		$-k_{jm}^{\cdot\cdot m} \cdot k_{mj}^{\cdot\cdot j} \cdot R_j^{\cdot\cdot}$							

$$R_j^{\cdot\cdot} = [R_j^{\cdot\cdot m} - k_{jm}^{\cdot\cdot m} \cdot R_m^{\cdot\cdot j}] \cdot \alpha_{jm}^{\cdot\cdot}, \qquad \alpha_{jm}^{\cdot\cdot} = \frac{1}{1 - k_{jm}^{\cdot\cdot m} \cdot k_{mj}^{\cdot\cdot j}}$$

Fig. 11: Proof of the Inhibition Theorem.

1.4.3.2 Goal

If we consider that:

- it has been proved that any unitary regime can be deduced from two regimes which have one more inhibited seat each;
- regimes with higher order of inhibition are easier to study, because inhibition practically means suppression;
- the theorem can be applied on the new two regimes, getting other more simple (inhibited) regimes, and so on, until they are no more simplifyable;

then the goal is to move the study of the (real) given system to regimes of maximum inhibition and thus of maximum solution simplicity.

1.4.4 Fundamental formula

The proof is valid for any unitary regime, and thus for any principal regime of order *v*. The fundamental formula can be deduced directly substituting *m* by *v* in (Eq. 2). Starting from:

j	1
m	0
	0
	$R_j^{\cdot\cdot}$

with *m*=*v* we get: $$R_j^{(v-1)} = \left[R_j^{(v)} - k_{jv}^{(v)} \cdot R_v^{(v)}\right] \cdot \boldsymbol{\alpha}_{jv}^{(v)}$$

where: $$\alpha_{jv}^{(v)} = \frac{1}{1 - k_{jv}^{(v)} k_{vj}^{(v)}}$$

which can be particularized for any quantity or inhibition quantity:

(Eq. 3) $$Q_j^{(v-1)} = \left[Q_j^{(v)} - k_{jv}^{(v)} \cdot Q_v^{(v)}\right] \cdot \alpha_{jv}^{(v)},$$

(Eq. 4) $$k_{jr}^{(v-1)} = \left[k_{jr}^{(v)} - k_{jv}^{(v)} \cdot k_{vr}^{(v)}\right] \cdot \alpha_{jv}^{(v)}.$$

1.4.5 Topological rule

The previous expressions are not properly user friendly, but it is possible to give a topological formula of them, avoiding the annoying use of indexes. We need to add a new column to the tableau, called the *auxiliary column*, where to put the values of $\alpha_{jv}^{(v)}$, in correspondence with $R_j^{(v)}$, as depicted in Fig. 12.

These values have to be calculated through the product between the inhibition quantity $k_{jv}^{(v)}$ (adjacent cell to $\alpha_{jv}^{(v)}$, on the left) and the inhibition quantity $k_{vj}^{(v)}$ (symmetrically located to previous cell, with respect to the greyed diagonal of the tableau).

After a look to Fig. 13, where two examples (*a,* for inhibition quantities and *a',* for generic quantities) are reported, it is easy to understand the topological rule:

$$a = \left[A - B \cdot C \right] \cdot D$$

E*very cell "a" can be calculated from the homologous cell "A" (same row and column in the immediately upper level) subtracting the product between the right end of row cell "B" and the lower end of column cell "C", and multiplying this difference by the value of cell "D" adjacent to "B" in the auxiliary column.*

Level 3	Q	Seat	$k_{\_1}$	$k_{\_2}$	$k_{\_3}$	α
$R_1^{(3)}$	**A'**	1		**A**	**B**	**D**
		2				
$R_3^{(3)}$	**C'**	3		**C**		***
Level 2						
$R_1^{(2)}$	**a'**	1		**a**	$\alpha_{12}^{(2)}$	
		2			**	
Level 1						
		1		*		

Fig. 12: Tableau with auxiliary column and symbol positions (topological formula).

$k_{12}^{(2)}$	**a**
$k_{12}^{(3)}$	**A**
$k_{13}^{(3)}$	**B**
$k_{32}^{(3)}$	**C**
$\alpha_{13}^{(3)}$	**D**

$Q_1^{(2)}$	**a'**
$Q_1^{(3)}$	**A'**
$k_{13}^{(3)}$	**B**
$Q_3^{(3)}$	**C'**
$\alpha_{13}^{(3)}$	**D**

$$a = \left[A - B \cdot C \right] \cdot D \qquad a' = \left[A' - B \cdot C' \right] \cdot D$$

Fig. 13: Correspondence between symbols for the topological formula.

1.4.6 Solving mono-supply regimes

The topological rule allows to deduce all effects in a given inhibition level when all principal regimes of the immediately upper level are known. The level of maximum inhibition has to be studied directly, of course, but its regimes are the most simple ones (elementary regimes).
When all elementary regimes are known, then the iterative application of the topological formula leads to the final evaluation of the Level 1 quantities. This Level 1 corresponds to the regime, mono-supplied in seat 1.
So, the basic algorithm is constituted by four steps:

1. Choice, enumeration and orientation of seats.
2. Direct solution of elementary regimes.
3. Calculation of inhibition quantities.
4. Calculation of effects.

For completeness, some remarks are here reported:

- the choice of the seats is normally arbitrary;
- the enumeration of the seats is arbitrary, except for the supplied seat, which has to be indexed with "1";
- orientation of seats (if necessary) is normally arbitrary;
- the inhibition quantities have to be calculated first, because they are needed for the effect calculation;
- effects can be calculated separately, because the calculus is row independent.

1.4.7 Solving multi-supply regimes

It is possible to solve regimes with more than one supplied seat because they can be reduced to mono-supply regimes, due to the superposition principle, following a similar procedure (not reported here) based on the ICS operator as well. The final formula (given for the same three-loop circuit) is a very simple expression:

(Eq. 5) $$R(Y_1, Y_2, Y_3) = F_1 \cdot R_1^{(1)} + F_2 \cdot R_2^{(2)} + F_3 \cdot R_3^{(3)}$$

where the following quantities

$$F_3 = Y_3,$$
$$F_2 = Y_2 - k_{32}^{(3)} F_3,$$
$$F_1 = Y_1 - k_{31}^{(3)} F_3 - k_{21}^{(2)} F_2$$

are called "auxiliary causes". We notice that all needed quantities in this formula are present or can be directly derived from the tableau calculated for the mono-supply regime, with a minimal calculation overhead.

1.4.8 Flexibility

There some general feature of the method that make it very flexible.

1.4.8.1 Change of supply

If the auxiliary causes are left symbolic, then we get expressions which are function of the supply values only. Once the full tableau is completed, then this fact allows to solve the originary system with the supply values configuration changed (not the seat displacement, of course), without redoing the entire tableau calculations: we simply need to recalculate the auxiliary causes by introducing the new supply values and then to use (Eq. 5) to get the requested effects corresponding to the new configuration.

1.4.8.2 Inverse matrix calculation

Once the full tableau is completed, inverse problem solving becomes immediate. An inverse problem is of the kind: "find the causes so that the given effects are produced" and it is depicted for a three-seat system, with the usual symbols, below:

$$\begin{aligned} X_1 &= b_{11}Y_1 + b_{12}Y_2 + b_{13}Y_3 \\ X_2 &= b_{21}Y_1 + b_{22}Y_2 + b_{23}Y_3 \\ X_3 &= b_{31}Y_1 + b_{32}Y_2 + b_{33}Y_3 \end{aligned}$$

The coefficients b_{ij} constitute the inverse matrix of the originary system. The element displacement in this matrix can be interpreted as follows (see Fig. 14)

- row (1): effects in regime $R(1,0,0)$;
- row (2): effects in regime $R(0,1,0)$;
- row (3): effects in regime $R(0,0,1)$.

Thus, the inverse problem is reduced to the calculation of the last two regimes by a supply change (see previous section) each, because the first one corresponds to $R_1^{(1)}$, already calculated at the end of the full tableau.

coefficients interpreted as:	Calculation needed:
$b_{11}=[X_1]_{Y_1=1,Y_2=0,Y_3=0}$ $b_{21}=[X_2]_{Y_1=1,Y_2=0,Y_3=0}$ $b_{31}=[X_3]_{Y_1=1,Y_2=0,Y_3=0}$	$R(1,0,0)$ (no extra calculation)
$b_{12}=[X_1]_{Y_1=0,Y_2=1,Y_3=0}$ $b_{22}=[X_2]_{Y_1=0,Y_2=1,Y_3=0}$ $b_{32}=[X_3]_{Y_1=0,Y_2=1,Y_3=0}$	$R(0,1,0)$ (one supply change)
$b_{13}=[X_1]_{Y_1=0,Y_2=0,Y_3=1}$ $b_{23}=[X_2]_{Y_1=0,Y_2=0,Y_3=1}$ $b_{33}=[X_3]_{Y_1=0,Y_2=0,Y_3=1}$	$R(0,0,1)$ (one supply change)

Fig. 14: Regime correspondence for inverse problem.

1.5 Example

Let us consider a 3-loop circuit with one dependent voltage generator of Fig. 15. The mutual induction between L1 and L2 is not considered (even if it is possible), and the parameter values are not given because we work in a completely symbolic way.
If the tree is composed by the branches 4 and 5, then the elementary regimes are those summarized in Fig. 16.
If we apply the procedure explained in the previous section we get the full inhibition sequence until the final solution row, which is here partially reported only for the quantity i_1:

$$i_1 = \frac{E\left(1+\alpha Cs+CL_2s^2\right)}{R_1+\left(L_1+L_2+\alpha CR_1\right)s+\left(CL_2R_1+\alpha CL_1\right)s^2+CL_1L_2s^3}$$

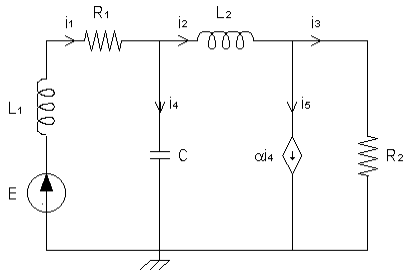


Fig. 15: Sample circuit.

	i_1	i_2	i_3	k_1	k_2	k_3
$R_1^{(3)}$	$\frac{1}{Z_{L1}+Z_C+R_1}$	0	0		$-(Z_C+\alpha)\cdot i_1$	$\alpha\cdot i_1$
$R_2^{(3)}$	0	$\frac{1}{Z_{L2}+Z_C+\alpha}$	0	$-Z_C\cdot i_2$		$-\alpha\cdot i_2$
$R_3^{(3)}$	0	0	$\frac{1}{R_2}$	0	0	

Fig. 16: Elementary regimes of the sample circuit

1.6 Conclusions for Part One

Symbolic programming can be used either for new methodologies or to renew old ones. In this paper an obsolete method from the point of view of numerical computing (called IME) has been reformulated symbolically in a complete abstract way, because its intrinsic nature of hierarchical problem decomposer helps to avoid some typical limitations of the symbolic analysis of electric circuits.

As a matter of fact, among the Mathematica environment, it allows to solve quite wide networks, to deduce (semi-)symbolic relationship between circuit parameters (very useful in circuit diagnosis, for example), or to utilize inhibition tableau data to easily deduce other quantities (related to the circuit, of course) like sensitivities and inverse matrix.

Furthermore, it is a flexible method because it is interdisciplinary and general in finding solutions. Also it allows to solve “inverse problems”, even if it is a non-inverting method.

Part One showed the theoretical basis of the method and some features, while the software implementation is shown in Part Two of this chapter.

1.7 References to Part One

[1] X. Tan, C. J. R. Shi: HIERARCHICAL SYMBOLIC ANALYSIS OF LARGE ANALOG CIRCUITS WITH DETERMINANT DECISION DIAGRAMS. *IEEE 1998.*

[2] W. M. G. Van Bokhoven, D. M. W. Leenaerts: EXPLICIT FORMULAS FOR THE SOLUTIONS OF PIECEWISE LINEAR NETWORKS. *IEEE Trans. on Circuits and System*, vol.46, n.9, 1999.

[3] M. Artioli, F. Ciampolini, F. Filippetti: SHORT DIDACTICS (SD): A FEASIBLE PROPOSAL TO SHORTEN LEARNING TIMES. Accepted at *EPE E = T_e M^2 Tomorrow's Education in Electrical Technologies:revisited Methods & Tools for renewed Motivation*, Liège 2001.

[4] M. Artioli, F. Ciampolini, F. Filippetti: INHIBITION METHOD: A "SHORT DIDACTICS" EXAMPLE FOR AN ENGINEERING APPROACH TO DIDACTICS. Accepted at *EPE E = T_e M^2 Tomorrow's Education in Electrical Technologies:revisited Methods & Tools for renewed Motivation*, Liège, 2001.

[5] F. Ciampolini: A METHOD FOR LINEAR ELECTRIC CIRCUIT ANALYSIS (in Italian), *L'Elettrotecnica*, n.10, 1968.

[6] R.Germundsson: VIEWPOINT: THE IMPORTANCE OF SYMBOLIC PROGRAMMING. *IEEE Spectrum 1999*.

[7] F. Filippetti, M. Martelli: SYMBOLIC TECHNIQUES ADDRESSED TO THE FAULT DIAGNOSIS OF LARGE LINEAR ELECTRIC CIRCUITS. *IEEE Mediterranean Electrotechnical Conf. MELECON'96*, Bari 1996.

[8] F. Filippetti, M. Martelli, M Artioli: IMPROVEMENTS TO THE FAULT DIAGNOSIS OF LARGE LINEAR ELECTRIC CIRCUITS VIA IME METHOD. *Int. Symp. on Theoretical Electrical Engineering ISTET'97*, Palermo 1997.

[9] F. Filippetti, M. Martelli:. NETWORK SENSITIVITY ANALYSIS OF LARGE CIRCUITS IN SYMBOLIC FORM. *European Conf. on Circuit Theory and Design ECCTD'95*, Istanbul 1995.

[10] A. Liberatore, S. Manetti: SAPEC A PERSONAL COMPUTER PROGRAM FOR THE SYMBOLIC ANALYSIS OF ELECTRICAL CIRCUITS, *IEEE ISCAS'88*, Helsinki 1988.

[11] F. V. Fernàndez, A Rodrìguez-Vàzquez: SYMBOLIC ANALYSIS TOOLS - THE STATE-OF-THE-ART. *IEEE 1996*.

2 Part Two - Implementation

This part is intended to show several software implementations of the general method exposed on Part One. They all are realised in a symbolic environment, based on the Mathematica™ program. Even though they can be used to approach any linear system, only electric circuits will be reported as test examples. The first implementation has a compact recursive algorithm, fast, but not suitable for wide circuits, because recursion needs a large amount of memory. The second is directly derived from a still compact but iterative (in closed form) algorithm, introducing a function for memory swapping onto mass storage media, to allow dealing with very wide circuits. The third one, not fully tested yet, is based on a matrix formulation: simple and elegant for a fast execution. For all these three implementations, the user has only to write down a PSPICE-like "net list" description (but even in symbolic, numeric or mixed way), because an initialisation function can translate it into the equations constituting the starting linear system. It is also possible to write down the system directly. From this point onwards, the procedure is completely transparent to the user which can collect, at the end, a symbolic expression (or semi-symbolic, if the starting system was too), ready for further calculations among the Mathematica environment.

2.1 Kernel of the implementations

The kernel of all proposed implementations is a procedure that takes an *n*-rank rectangular matrix as input and gives an *(n-1)*-rank rectangular matrix as output. They represent all data needed to describe the system regimes, respectively, of level *n* and *n-1* (with *n* and *n-1* seats inhibited): the less inhibited is calculated from the more inhibited as explained in Part One. In older implementations of this method there were several kind of matrices, each one devoted to different kind of quantities: effects, inhibition quantities, auxiliary quantities and so on. In these new implementations they are collected and compacted in a single matrix structure to reduce memory allocations and to take advantage of the built-in list/matrix oriented functions of Mathematica, which allow a faster execution with a single call on wide data structures rather than multiple calls on narrow data structures. There are several ancillary procedures as well, i.e. for translating data from/to the user or to perform the task in different ways, with the selected optimisation. All procedures are organised in a Mathematica package and they are called from higher level functions which are the shell of a multi-algorithmic tool for linear system (and linear electric circuit, of course) analysis.

2.1.1 Input matrix

As input matrix we refer not to the data delivered by the user, but to the matrix compiled by the symbolic processor starting from the problem description made by the user. Considering an electric circuit the description can be made either

through an input file in a PSPICE-like form or through an equation system directly written using whatever method you like. For example, in Fig. 17 a very simple circuit is shown, just to indicate two possible descriptions, reported in Fig. 18: a “net list” description and an equation system in terms of loop currents and an extra equation due to the presence of the current source.

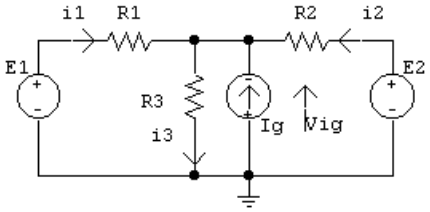


Fig. 17: Sample circuit.

```
V_E1,  1, 0          {
V_E2,  2, 0               E1==R1*i1+R3*i3,
I_Ig,  0, 3               E2==R2*i2+R3*i3,
R_R1,  1, 3      ↔        Vig==R3*i3,
R_R2,  3, 2               i1+i2+Ig-i3==0
R_R3,  0, 3          }
```

Fig. 18: “Net list” and equations of the sample circuit.

In the net list symbols, the first character defines the component type and the characters following the underscore define the related symbolic name, utilized during the symbolic analysis.
The package has two particular ancillary functions: the first one to translate the net list into a system of linear equations (following in this case only the Modified Nodal Analysis - MNA) and the second one (reported as example in Fig. 19) to translate the equation system into the input matrix. In this case the user can utilize any methodology to write equations.

```
IMEStartingTableauFromSystem[A_]:=Block[
     {
          G, K, LenK, alfa
     },
     G=DiagonalMatrix[1/DiagonalElements[A]];
     K=SetDiagonalElements[A/DiagonalElements[A],Null];
     LenK=Length[K];
     alfa=Table[{1/(1-K[[i,LenK]]*K[[LenK,i]])},{i,LenK}];
     alfa[[LenK]]={Null};
     Return[AppendRows[G,K,alfa]];
     ];
```

Fig. 19: Input matrix translation function.

This matrix *mat* is a rectangular $n \times (n+n+1)$ matrix and has the following shape:

$$mat = \left[\begin{array}{c|c|c} n_x n \text{ effects} & n_x n \text{ inhibition quant.} & n_x 1 \text{ auxiliary quant.} \end{array}\right]$$

Acting in this manner, the input matrix can be considered a kind of interface between different circuit representation or even the circuit formalisation “made by hand” by the user. Due to the fact that IME works with every linear system in an absolutely abstract and symbolic way, it does no matter wherever the equations come from.

```
IMEShrink[mat_,gnum_]:=Block[
     {
          v=Length[mat],
          w=Length[mat[[1]]],
          newmat,
          newel,
          endcol,
          endrow
     },
     endrow=v;
     endcol=w-1;
     newmat=Table[0,{i,endrow-1},{j,endcol}];
          newel[r_,c_]:=(mat[[r,c]]-
mat[[r,endcol]]*mat[[endrow,c]])*mat[[r,w]];
     Do[
          newmat[[i,gnum+j]] =newel[i,gnum+j];
          newmat[[j,gnum+i]]=newel[j,gnum+i];
          ,{i,2,v-1},{j,i-1}];
     If[gnum>0,
          (* calc effects, if any *)
          Do[
          newmat[[i,i]]=newel[i,i];
          Do[
               newmat[[i,j]]=newel[i,j];
               ,{j,endrow,gnum}];
          ,{i,endrow-1}];
          ];
     Do[
               newmat[[i,endcol]]=1/(1-newmat[[i,endcol-
1]]*newmat[[endrow-1,gnum+i]]);
          ,{i,endrow-2}];
     Return[newmat];
];
```

Fig. 20: Kernel code of the recursive and iterative implementation.

2.1.2 Recursive and iterative kernel

The method can be activated in different ways, because there are different procedure for the same task depending on the optimisation needed. They all share the same data structure for maximum interoperability, but the recursive and the

iterative implementation share the same kernel as well, whose code is reported in Fig. 20. It is a C-like translation of the formula written in the introduction and applied to the proper elements of the input matrix *mat*.

```
IMEShrinkByMatrix[t_]:=Block[
     {
     tempTableau
     },

     tempTableau=(
     TakeMatrix[t,{1,1},{-2,-3}]-
            Dot[TakeMatrix[t,{1,-2},{-2,-2}],TakeMatrix[t,{-
1,1},{-1,-3}]])*
   Flatten[TakeMatrix[t,{1,-1},{-2,-1}]];
     Return[
          Together[
               AppendRows[tempTableau,
     Map[List,
       1/(1-Flatten[TakeColumns[tempTableau,-1]]*
               First[TakeMatrix[
                        tempTableau,{-1,-
Length[tempTableau]},{-1,-1}]])]]
          ]
     ];
];
```

Fig. 21: Matrix kernel code.

2.1.3 Matrix kernel

The kernel of the matrix formulation works on a mathematical relationship between matrices and not between matrix elements like the previous kernel does. This approach yields in a very compact code, faster to be executed (in interpreted and not compiled symbolic environment) and takes advantage of the optimised matrix manipulation function of Mathematica. Matrix operands and operators in this kernel are given as follows. The logic and the topological formula both explained in Part One suggest to consider the input matrix structured as depicted in the first three rows of Fig. 22: the tableau of inhibition level *n* is subdivided in a rectangular matrix $\overline{A}$, a row vector $\overline{B}$ and two column vectors $\overline{C}$ and $\overline{D}$. It is easy to verify that the submatrix $\overline{a}$, representing the tableau of inhibition level *n-1* (without auxiliary quantity column, in the last row of Fig. 22), can be calculated with this formula:

$$\overline{a} = \left[\overline{A} - \overline{B} \cdot \overline{C}\right] * \overline{D}$$

where $\cdot$ and $*$ are respectively the dot matrix product and the per scalar product between matrices.

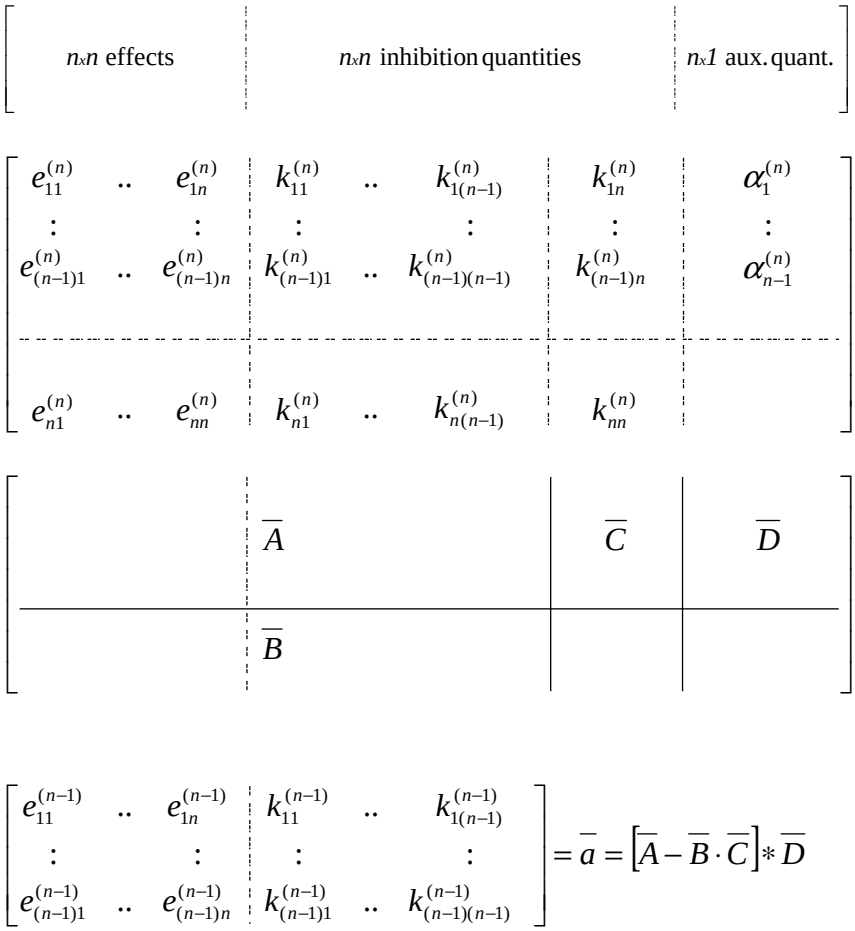


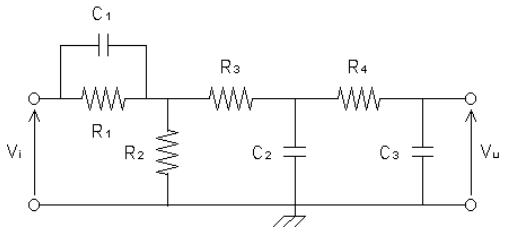
Fig. 22: Matrix structures for the matrix kernel.

The code to implement this kernel is in Fig. 21. We emphasise that the symbolic processor allows to work with a very simple algorithm, once the topological rule is given in terms of symbolic matrix formula.

2.2 Implementations

Although the three implementations do not constitute a stand-alone tool, they represent a complete package to handle linear (systems and) circuits. Properly, IME solves only the "core", that is, it calculates all *effects* which are present in the input equation or which are indicated in the PSPICE-like batch file containing the net list description, and it does not implement functions for post-processing

the outputs because the Mathematica environment itself offers the ability to (symbolically or not) manipulate results for expression simplification, approximation, diagram and graphics creation. In addition, IME implements functions to manipulate input data like net list descriptions, traditional equations or matrix equations to give to the user the greatest reliability in thinking and programming solutions. It is also possible to link the IME processor to several packages for other purposes like sensitivity calculations [1] or fault detection (fault dictionaries [2], Cycling Verify method [3,4, Chapter III]) to extend the range of IME applications. These applications share the same disadvantages and advantages of all interpreted environment: not the fastest execution, no memory optimisation, but great flexibility and code maintenance for further expansions.
The mentioned disadvantages are mitigated by the intrinsic characteristics of IME theoretical formulation and by the particular implementations proposed, but in comparison with other symbolic processors there are some other remarks [5,6]. Specific benchmarks for symbolic processors do not exist (yet), so a truly comparison is not possible [6], but it could be interesting to outline some features and results referring to a practical example, already used in [2], and reported in the next section.




```
..
V_Vi, 1, 0
C_C1, 1, 2
R_R1, 1, 2
R_R3, 2, 3
C_C2, 0, 3
R_R4, 3, 4
C_C3, 0, 4
..
```

Fig. 23: Sample circuit and its “net list” .

```
{
   -(i2*R2) + i1*(R2 + R1/(1 + C1*R1*s))           == Vi,
   -(i1*R2) + i2*(R2 + R3 + 1/(C2*s)) - i3/(C2*s)  == 0,
    i3*(R4 + 1/(C2*s) + 1/(C3*s)) - i2/(C2*s)      == 0
}
```

Fig. 24: Equations written in terms of loop currents.

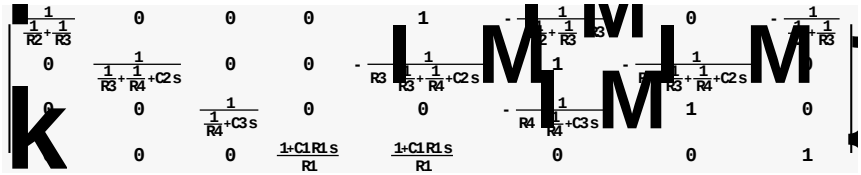


Fig. 25: Input matrix (with no auxiliary column) in terms of MNA.

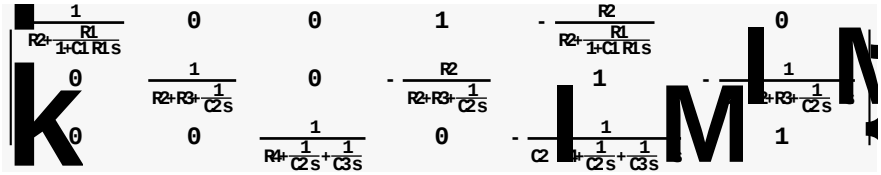


Fig. 26: Input matrix (with no auxiliary column) in terms of loop currents.

2.3 Circuit example

Let us consider the circuit of Fig. 23. Its description can be given in a PSPICE-like manner or through directly written equations like in Fig. 24. In both cases, this description will be automatically translated into the input matrix by an ancillary function. The input matrix, in symbolic form, is reported in Fig. 25 and Fig. 26, starting respectively from the "net list" description and from loop current equations (without the auxiliary column for readability reasons). Now, IME and other calculations will be activated either by commands in a batch file or interactively by the user, which can work on them from the desktop. A great advantage of this package is that the user has not to worry about the input form and that he has (virtually) no constraints on managing the output. Examples of further elaboration on the circuit solution are reported in Fig. 27, Fig. 28 and Fig. 29, with particular interest to circuit diagnostics.

```
H(s,R1,R2,R3,R4,C1,C2,C3) = Vu/Vi =
      (R2 + C1*R1*R2*s)/(R1 + R2 + (C1*R1*R2 + C2*R1*R2 + C3*R1*R2 +
      C2*R1*R3 + C3*R1*R3 + C2*R2*R3 + C3*R2*R3 + C3*R1*R4 +
      C3*R2*R4)*s + (C1*C2*R1*R2*R3 + C1*C3*R1*R2*R3 +
      C1*C3*R1*R2*R4 + C2*C3*R1*R2*R4 + C2*C3*R1*R3*R4 +
      C2*C3*R2*R3*R4)*s^2 + C1*C2*C3*R1*R2*R3*R4*s^3)
```

Fig. 27: Network function *H=Vu/Vi* with all symbolic parameters.

```
SensC2 = D[H,C2]*C2/H =
      -((C2*((R1*R2 + R1*R3 + R2*R3)*s + (C1*R1*R2*R3 + C3*R1*R2*R4
      + C3*R1*R3*R4 + C3*R2*R3*R4)*s^2 + C1*C3*R1*R2*R3*R4*s^3))/
          (R1 + R2 + (C1*R1*R2 + C2*R1*R2 + C3*R1*R2 + C2*R1*R3 +
      C3*R1*R3 + C2*R2*R3 + C3*R2*R3 + C3*R1*R4 + C3*R2*R4)*s +
            (C1*C2*R1*R2*R3 + C1*C3*R1*R2*R3 + C1*C3*R1*R2*R4 +
      C2*C3*R1*R2*R4 + C2*C3*R1*R3*R4 + C2*C3*R2*R3*R4)*s^2 +
      C1*C2*C3*R1*R2*R3*R4*s^3))
```

Fig. 28: Symbolic sensitivity of *H* to the parameter *C2*.

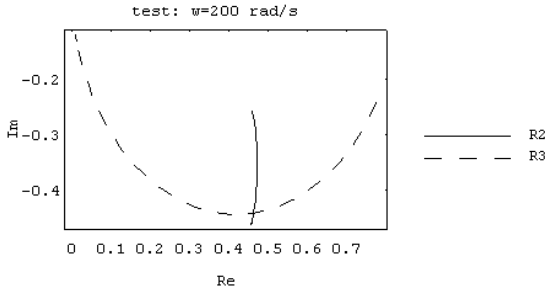


Fig. 29: Fault dictionary in terms of real and imaginary part of *H*, related to *R2* and *R3*, varying from 0.1 to 10 times the rated value.

2.4 Conclusions for Part Two

Part Two pointed out some features related to the software implementation of a method for the symbolic analysis of linear electric circuit, explained in detail in Part One. The proposed implementation consists in a Mathematica package containing three different but interchangeable versions of the method plus a function library to help the user preparing input data. No post-processing function is given, because the Mathematica environment itself offers a great variety of functions to manipulate results in symbolic, numeric or graphical way. Thus the implementation covers only the software engine to efficiently solve wide linear circuits, due to the intrinsic nature of the method, based on a hierarchical decomposition of the starting problem into smaller and easier sub-problems and on a further recollection of them.

2.5 References to Part Two

[1] F. Filippetti, M. Martelli:. NETWORK SENSITIVITY ANALYSIS OF LARGE CIRCUITS IN SYMBOLIC FORM. *European Conf. on Circuit Theory and Design ECCTD'95,* Istanbul 1995.

[2] F. Filippetti, M. Martelli: SYMBOLIC TECHNIQUES ADDRESSED TO THE FAULT DIAGNOSIS OF LARGE LINEAR ELECTRIC

CIRCUITS. *IEEE Mediterranean Electrotechnical Conf. MELECON'96*, Bari 1996.

[3] M Artioli, F. Filippetti: LINEAR ANALOG CIRCUIT DIAGNOSIS BASED ON SYMBOLIC ANALYSIS AND REDUCED OBSERVABLE POINT SET. *International Symposium on Theoretical Electrical Engineering ISTET'99*, Magdeburg 1999.

[4] M Artioli, F. Filippetti: SINGLE AND MULTIPLE FAULT DIAGNOSIS BASED ON SYMBOLIC ANALYSIS AND REDUCED SET OF OBSERVABLE POINTS FOR LINEAR ANALOG CIRCUITS. *IEEE International Conference on Circuits and Systems ICECS2000*, Beirut 2000.

[5] A. Liberatore, S. Manetti: SAPEC A PERSONAL COMPUTER PROGRAM FOR THE SYMBOLIC ANALYSIS OF ELECTRICAL CIRCUITS, *IEEE ISCAS'88*, Helsinki 1988.

[6] F. V. Fernàndez, A Rodrìguez-Vàzquez: SYMBOLIC ANALYSIS TOOLS - THE STATE-OF-THE-ART. *IEEE 1996*.

Chapter III

Reflection on circuit diagnostics

Symbolic Calculus (SC) is nowadays becoming a real way to circuit analysis due to powerful symbolic processors and even commercial ones running on a normal PC [1,2,3]. They often use a large amount of memory, but there are algorithms to treat wide circuits too. Related to this, several Authors showed that SC allows to define new diagnostic techniques or to improve existing ones [4,5,6,7,8,9], because the symbolic solution of a circuit is one of its general representation that can be particularized under several conditions as needed, taking into account i.e. measured values, parameters tolerances or that can be further processed to simply obtain other characterizations for each situation. In particular, the IME method [Chapter II], based on the superposition principle, yields the general symbolic solution of linear circuits by means of hierarchical sequences of more elementary analyses of smaller circuits (derived from the starting one, suppressing some elements and thus more simple), working with a unique four-term formula. This "circuit splitting" logic [Chapter I] is intended for relatively wide electric circuits and it is performed mathematically [3], e.g. from the admittance matrix, or physic-functionally [8,10].

1 Part One - Cycling Verify Method

1.1 The basic idea

Part One is intended to show the possibility to perform fault location and identification in case of single or double fault for linear analog circuits with a symbolic algorithm that allows to use few observable points. Under some hypothesis [11], the fault location is made through a first group of measures for the analysis and a second one for the validation. A group of test equations, obtained from the symbolic solution of the circuit, is cyclically solved in turn for each group of parameters under test, leaving the other ones at their rated value. A

validation equation, still obtained from the same symbolic solution, has the task to validate the faulty or non-faulty situation for those parameters.

1.2 Single fault diagnosis - Problem definition

1.2.1 Definition

We consider the simple case of a single fault due to a variation from the rated value of a component; we exclude short or open circuits which correspond to circuit topology modifications. Assuming to have:

- a symbolic solution of a linear circuit in terms of node voltage (if in terms of loop current it would be equivalent) which allows a first test point expressed as:

$$tv(p_1, p_2, .., p_m)$$

where $(p_1, p_2, .., p_m)$ are the circuit parameters

- a measured test voltage $\overline{tv}$ at a fixed frequency to detect the fault;
- a symbolic expression still derived from the symbolic circuit solution and related to a validation point (on a second observable node) expressed as:

$$vv(p_1, p_2, .., p_m)$$

- a measured validation voltage $\overline{vv}$ at the same fixed frequency.

If we assume that the fault occurrence is possible only among the parameters to which the test voltage has sensitivity, we will verify the sensitivity of the test voltage to each parameter to exclude that ones that don't affect the test voltage over a definite threshold. If the affecting parameters are

$$(p_1, p_2, .., p_n),$$

the excluded ones are

$$(p_{n+1}, p_{n+2}, .., p_m)$$

and they will be taken at their rated value, thus the symbolic expressions are function of only:

$$(p_1, p_2, .., p_n).$$

We emphasise the strategic importance to have a symbolic solution, because all sensitivities can be directly calculated through algebraic manipulations by the symbolic processor.

Now, it is possible to define the following equations:

(Eq.1) $$tv(p_1, p_2,.., p_n) = \overline{tv}$$

(Eq.2) $$vv(p_1, p_2,.., p_n) = \overline{vv}$$

They are respectively called *test equation* and *validation equation*. Of course, they are satisfied when the values of all parameters and the measured values are referring to the same situation being measured.

1.2.2 Diagnosis

If the rated value of the affecting parameters are

$$(p_1, p_2,.., p_n),$$

and the actual values are

$$(\overline{p}_1, \overline{p}_2,.., \overline{p}_n),$$

then, according to the definitions of the previous subsection, it will be:

$$tv(\overline{p}_1, \overline{p}_2,.., \overline{p}_n) = \overline{tv}.$$

If we assign the rated value to all but one parameters we can write an equation in one variable. Leaving the first parameter as the unique unknown we get an equation of the type:

$$tv(x_1, \overline{p}_2,.., \overline{p}_n) = \overline{tv}.$$

Solving this equation with respect to x_1 we obtain the value of p_1 as if it were the parameter that actually affects the fault. Cycling this calculations through the *n* parameters at the end we get *n* vectors, which can be rearranged in the *matrix of the possible single faults*:

$$\begin{bmatrix} x_1 & p_2 & p_3 & .. & p_n \\ p_1 & x_2 & p_3 & .. & p_n \\ p_1 & p_2 & x_3 & .. & p_n \\ .. & .. & .. & .. & .. \\ p_1 & p_2 & p_3 & .. & x_n \end{bmatrix}.$$

Each of them is the solution of an equation and contains *n-1* rated values and one hypothetical faulty value. If the circuit behaves as faulty and only one parameter can be out of its rated value (saying x_1), only one of the previous *n* vectors will satisfy the validation equation:

$$vv(x_1, p_2, p_3, .., p_n) = \overline{vv}$$
$$vv(p_1, x_2, p_3, .., p_n) \neq \overline{vv}$$
$$vv(p_1, p_2, x_3, .., p_n) \neq \overline{vv}$$
$$..$$
$$vv(p_1, p_2, p_3, .., x_n) \neq \overline{vv}$$

Thus, checking the validation equations, we get the information about the faulty parameter, because the *i-th* satisfied equation corresponds to the *i-th* component (parameter) of the row.

1.2.3 Example

Let's consider the very simple capacitive-resistive circuit of Fig.1, where the impedance values are reported in ohm. We have to work with complex numbers, but Mathematica function are able to transparently treat these kind of numbers, allowing the symbolic solution of complex equations of real unknown. Parameter values are meaningless, because they are intended just for doing simply demonstrative calculations.

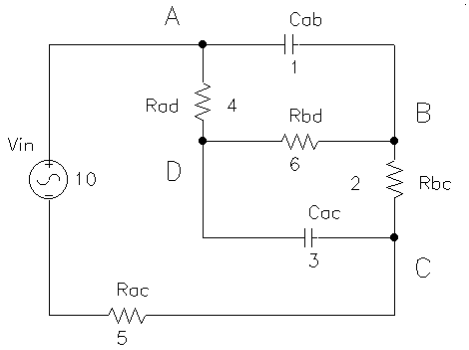


Fig.1: RC circuit example.

Impedances are split in their two real parameters *R* and *X*; fixing the supply frequency to 1, and taken *Vbd* as test voltage

$$tv(R_{ab}, X_{ab}, R_{bc}, R_{cd}, R_{ac} X_{ac}, R_{ad}, R_{bd}),$$

and *Vbc* as validation voltage

$$vv(R_{ab}, X_{ab}, R_{bc}, R_{cd}, R_{ac} X_{ac}, R_{ad}, R_{bd}),$$

and solving symbolically the circuit we get from Mathematica the expressions for *tv* (reported below) and *vv*, in terms of complex numbers:

```
tv(Xab,Rbc,Rcd,Xac,Rad,Rbd)==(Rbd Vin (Rab+I Xab) (Rad Rbc+(Rab+I
    Xab) (-Rcd-I Xcd)))/((-Rbc (Rab+I Xab)-Rbd (Rab+Rac+Rbc+I Xab))
    (Rad Rbc+(Rab+I Xab) (-Rcd-I Xcd))-(-Rac (Rab+I Xab)-Rad
    (Rab+Rac+Rbc+I Xab)) (Rbc Rbd+(Rab+I Xab) (Rbc+Rbd+Rcd+I Xcd)))
```

If we simulate a fault by imposing that *Cab* has value 2 instead of 1, we get the (simulated) measured test and validation voltage:

$$\overline{tv} = 1.7902 + 0.063\ \mathrm{I},$$

$$\overline{vv} = 1.7903 + 0.0235\ \mathrm{I}.$$

Cycling the test equation through the parameters list, we obtain the following equations in a single unknown, from an automated procedure:

$$\left\{1.79025 + 0.0630976\,\mathrm{I} == \frac{60.\ (948.482 + 6.28319\,\mathrm{I}\,\mathrm{Rab})}{(31723.6 - 1796.99\,\mathrm{I}) + (8527.34 - 106.814\,\mathrm{I})\,\mathrm{Rab}},\right.$$

$$1.79025 + 0.0630976\,\mathrm{I} == \frac{60.\ \mathrm{I}\ (-150.796 + \mathrm{Xab})}{(-70. - 5051.68\,\mathrm{I}) + (1357.17 - 17.\ \mathrm{I})\,\mathrm{Xab}},$$

$$1.79025 + 0.0630976\,\mathrm{I} == \frac{60.\ (1. + 473.741\,\mathrm{Rbc})}{(14197.2 - 1332.04\,\mathrm{I}) + (8763.21 - 232.478\,\mathrm{I})\,\mathrm{Rbc}},$$

$$1.79025 + 0.0630976\,\mathrm{I} == \frac{60.\ (948.482 + 18.8496\,\mathrm{I}\,\mathrm{Rcd})}{(31723.6 - 1796.99\,\mathrm{I}) + (8290.47 - 320.442\,\mathrm{I})\,\mathrm{Rcd}},$$

$$1.79025 + 0.0630976\,\mathrm{I} == -\frac{60.\ \mathrm{I}\ (50.2655 - 1.\ \mathrm{Xcd})}{(-72. - 1683.89\,\mathrm{I}) + (439.823 - 17.\ \mathrm{I})\,\mathrm{Xcd}},$$

$$1.79025 + 0.0630976\,\mathrm{I} == \frac{60.\ (1. + 236.871\,\mathrm{Rad})}{(7093.12 - 1017.88\,\mathrm{I}) + (6157.63 - 194.779\,\mathrm{I})\,\mathrm{Rad}},$$

$$1.79025 + 0.0630976\,\mathrm{I} == \frac{56908.9}{(5672.89 - 728.849\,\mathrm{I}) + (5210.15 - 213.628\,\mathrm{I})\,\mathrm{Rac}},$$

$$\left.1.79025 + 0.0630976\,\mathrm{I} == \frac{9484.82\,\mathrm{Rbd}}{(4726.41 - 515.221\,\mathrm{I}) + (4499.54 - 213.628\,\mathrm{I})\,\mathrm{Rbd}}\right\}$$

each giving, in turn, the hypothetical value for one parameter if it were faulty. For each situation we get a vector with the parameter values taken from the rows of the matrix of the possible single faults:

$$\begin{pmatrix}
0.08\,\mathrm{I} & -0.2 & 2. & 0 & -0.05 & 4. & 5. & 6. \\
0 & -0.08 & 2. & 0 & -0.05 & 4. & 5. & 6. \\
0 & -0.2 & 2. - 0.1\,\mathrm{I} & 0 & -0.05 & 4. & 5. & 6. \\
0 & -0.2 & 2. & -0.007 + 0.08\,\mathrm{I} & -0.05 & 4. & 5. & 6. \\
0 & -0.2 & 2. & 0 & 0.02 + 0.007\,\mathrm{I} & 4. & 5. & 6. \\
0 & -0.2 & 2. & 0 & -0.05 & 4. - 0.3\,\mathrm{I} & 5. & 6. \\
0 & -0.2 & 2. & 0 & -0.05 & 4. & 5. + 0.1\,\mathrm{I} & 6. \\
0 & -0.2 & 2. & 0 & -0.05 & 4. & 5. & 5. - 0.8\,\mathrm{I}
\end{pmatrix}$$

They have to be validated by the equation, which corresponds to (Eq. 2),

$$vv(..) = 1.7903 + 0.0235\ \mathrm{I}.$$

The cyclical verification of the previous equation for each row of the matrix yields the validation vector

```
{True,True,False,False,False,False,False,False}
```

where the “True” at the first two places indicates that the first complex parameter (or equivalently the first two real ones) is the faulty component, as expected.

1.3 Double fault diagnosis - Problem definition

1.3.1 Definition

The problem definition is the same as the previous one. The only difference is that the fault is due to a variation from the rated value of two components; we still exclude short or open circuits which correspond to circuit topology modifications, and do not consider parameters tolerances or sensitivities.
Therefore we have the nearly the same assumptions as in the case of single fault:

- a symbolic solution of a linear circuit in terms of node voltage (if in terms of loop current it would be equivalent) which allows the first two test points to be expressed as:

  $$tv1(p_1, p_2,.., p_n)$$
  $$tv2(p_1, p_2,.., p_n)$$

  where $(p_1, p_2,.., p_n)$ are the circuit parameters
- two measured test voltages $\overline{tv1}$ and $\overline{tv2}$ at a fixed frequency to detect the fault;
- a symbolic expression still derived from the symbolic circuit solution and related to a validation point expressed as:

  $$vv(p_1, p_2,.., p_n)$$

  on a third observable node;
- a measured validation voltage $\overline{vv}$ at the same fixed frequency.

1.3.2 Diagnosis

Like in the previous case, if the rated value of the affecting parameters are $(p_1, p_2,.., p_n)$, and the real values are $(\overline{p}_1, \overline{p}_2,.., \overline{p}_n)$, then it will be:

$$tv1(\overline{p}_1, \overline{p}_2,.., \overline{p}_n) = \overline{tv1},$$
$$tv2(\overline{p}_1, \overline{p}_2,.., \overline{p}_n) = \overline{tv2}.$$

If we assign the rated value to all but two parameters we can write two equations in two variables. Leaving the (i.e.) first two parameters as the unique unknowns we get an equation system of the type:

$$tv1(x_1, x_2, \overline{p}_3,.., \overline{p}_n) = \overline{tv1}$$
$$tv2(x_1, x_2, \overline{p}_3,.., \overline{p}_n) = \overline{tv2}$$

Solving this system with respect to x_1 and x_2 we obtain the value of p_1 and p_2 as if they were the parameters that actually affect the fault. Cycling this calculation through all the possible couples of parameters, at the end we get a collection of vectors, which can be rearranged in the *matrix of the possible double faults*:

$$\begin{bmatrix} x_1 & x_2 & p_3 & p_4 & p_5 & .. & p_n \\ x_1 & p_2 & x_3 & p_4 & p_5 & .. & p_n \\ x_1 & p_2 & p_3 & x_4 & p_5 & .. & p_n \\ .. & .. & .. & .. & .. & .. & .. \\ p_1 & x_2 & x_3 & p_4 & p_5 & .. & p_n \\ p_1 & x_2 & p_3 & x_4 & p_5 & .. & p_n \\ .. & .. & .. & .. & .. & .. & .. \\ p_1 & p_2 & .. & .. & .. & x_{n-1} & x_n \end{bmatrix}.$$

Each of them is the solution (if it exists) of an equation system and contains *n-2* rated values and two hypothetical faulty values. If the circuit behaves as faulty and only two parameters can be out of their rated values (saying x_1 and x_2), only one of the previous vectors will satisfy the validation equation:

$$vv(x_1, x_2, p_3, p_4, .., p_n) = \overline{vv}$$
$$vv(p_1, x_2, x_3, p_4, .., p_n) \neq \overline{vv}$$
$$vv(p_1, p_2, x_3, x_4, .., p_n) \neq \overline{vv}$$
$$..$$
$$vv(p_1, p_2, p_3, .., x_{n-1}, x_n) \neq \overline{vv}$$

Thus, checking the validation equations, we get the information about the faulty parameters, because the *i-th* satisfied equation corresponds to the *i-th* component couple.

1.3.3 Example

Let's consider the very simple resistive circuit of Fig. 30. Parameter values (reported in ohm in the figure) are still meaningless, because they are intended just for doing simply demonstrative calculations, like in the previous example.

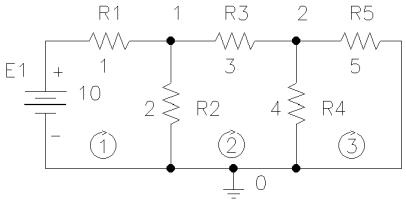


Fig. 30: Circuit example.

*

At this time we will take loop currents instead of voltages as test functions:

$$I1 \text{ as } ti1(R_1, R_2, R_3, R_4, R_5),$$
$$I2 \text{ as } ti2(R_1, R_2, R_3, R_4, R_5).$$

Taken the voltage on node 1 as validation function,

$$V1 \text{ as } vv(R_1, R_2, R_3, R_4, R_5),$$

and solving symbolically the circuit we get from Mathematica the equations for *ti1*, *ti2* and *vv;* for example the corresponding equation for *ti1* is:

```
ti1(R1,R2,R3,R4,R5)==(10 (R2 R4 + R3 R4 + R2 R5 + R3 R5 + R4 R5)) /
     (R1 R2 R4 + R1 R3 R4 + R2 R3 R4 + R1 R2 R5 + R1 R3 R5 + R2 R3
     R5 + R1 R4 R5 + R2 R4 R5)
```

If we simulate a fault by imposing that *R2* has value 6 instead of 2 and *R5* has value 7 instead of 5 we get the (simulated) measured test currents and validation voltage:

$$\bar{ti}1 = \frac{1270}{493}, \quad \bar{ti}2 = \frac{660}{493}, \quad \bar{vv} = \frac{3660}{493}.$$

If we apply the single fault checking procedure, assuming for example *ti1* as test point and *vv* as validation point, it fails; that is, no validation equation is satisfied for the single fault. Therefore it is possible to continue looking for the double fault.

Cycling the test system through the parameter couples

```
{{R1,R2},{R1,R3},{R1,R4},{R1,R5},{R2,R3},{R2,R4},{R2,R5},{R3,R4},{R
      3,R5},{R4,R5}},
```

we obtain equations like the following two ones, given for the possible couple {R3, R5}:

$$\left\{\frac{10\,(1 + R3 + 2\,R5 + R3\,R5)}{1 + 2\,R3 + 3\,R5 + 2\,R3\,R5} == \frac{110}{21},\ \frac{10\,(1 + R5)}{1 + 2\,R3 + 3\,R5 + 2\,R3\,R5} == \frac{20}{7}\right\}$$

For each situation we give a vector with the parameter values taken from the rows of the matrix of the possible double faults, partially reported:

$$\begin{pmatrix} 1.16 & 5.65 & 3. & 4. & 5. \\ 2.92 & 2. & -0.37 & 4. & 5. \\ 2.92 & 2. & 3. & -0.93 & 5. \\ 2.92 & 2. & 3. & 4. & -0.89 \\ 1. & 6. & 3.32 & 4. & 5. \\ 1. & 6. & 3. & 5.18 & 5. \\ 1. & 6. & 3. & 4. & 7. \\ .. & .. & .. & .. & .. \end{pmatrix}$$

They have to be validated by the equation

$$vv(..) = \frac{3660}{493}$$

The cyclical verification of the previous equation for each row of the matrix yields the validation vector:

```
{False,False,False,False, False,False,False,True, ..}
```

where the “True” at the seventh place indicates the seventh parameter couple,

that is *R2* and *R5* are the faulty components, as expected.

1.4 Multiple fault

It might be possible that this method could be extended to the more general case of multiple fault, because, if the double fault checking procedure fails, it should be possible in the same manner to continue looking for three faulty parameters, and so on. The main task wold be to build all the three (or *k*) components combinations and to solve the right three (or *k*) test equations with respect to those parameter variables in turn. The procedure calculation time grows with the multiplicity *k* of the checked fault, but it should still remains a fast procedure due to presence of the symbolic solution. It is possible that two or more rows satisfy the validation equation, that is the procedure can only individuate a group of possible faulty components. To solve this ambiguity we have to change the test and/or the validation point; if the solution is still ambiguous, changing the diagnostic technique could be needed [8,10,11]

1.5 Interval algebra for real circuit tolerances

In the previous sections we outlined the ground basis of this method to point out its formal structure. To handle real circuits we need to take into account tolerances [6,14]: this can be done without changing the formal structure of the method, due to its intrinsic symbolic formulation. Rated values and parameter values can be passed to the symbolic processor as range of values and the further calculations are performed in a transparent way by the "interval algebra" embedded in the symbolic processor. For example, a parameter value (*R4*) is expressed in Mathematica form as:

```
R4=Interval[{3.8,4.2}]
```

for a value of 4 with a ±5% tolerance. Test equations look like the following:

```
    (Interval[{19.99, 20.00}]*(R1*Interval[{2.99, 3.00}] +
Interval[{53.99, 54.10}])) / (R1*Interval[{122.90, 123.00}] +
                Interval[{477.98, 478.01}])
                          ==
                  Interval[{1.67, 1.89}],
```

which is automatically generated by the test procedure and solved by the symbolic processor for:

```
{R1→Interval[{1.01807,1.95378}]}
```

1.6 Conclusions for Part One

Considering a symbolic solution of a circuit and the Interval Algebra embedded in the symbolic processor, the basic idea to have a group of test equations through which it is possible to check in turn all the correspondent parameter

groups, seems to yield a simple algorithm for fault location and identification. In our elementary examples the groups count one or two equations to detect respectively one or two parameters (single or double fault case), but it could be hopefully extended to a higher number of parameters. In this case a higher number of observable nodes is needed. In case of ambiguity it is possible to change the test and validation group.

1.7 References to Part One

[1] A.Liberatore, S.Manetti: SAPEC A PERSONAL COMPUTER PROGRAM FOR THE SYMBOLIC ANALYSIS OF ELECTRICAL CIRCUITS. *IEEE Int. Symp. on Circuits and Syst. ISCAS'88*, Helsenki June 1988.

[2] A.Liberatore, S.Manetti: NETWORK SENSITIVITY ANALYSIS VIA SYMBOLIC FORMULATION. *IEEE Int. Symp. on Circuits and Syst. ISCAS'89*, Portland June 1989.

[3] X.Tan, C.J.R. Shi: HIERARCHICAL SYMBOLIC ANALYSIS OF LARGE ANALOG CIRCUITS WITH DETERMINANT DECISION DIAGRAMS. *Int. Symp. on Circuit and System ISCAS'98.*

[4] M. Artioli, F. Filippetti, M. Martelli: IMPROVEMENTS TO THE FAULT DIAGNOSIS OF LARGE LINEAR ELECTRIC CIRCUITS VIA IME METHOD. *Int. Symp. on Theoretical Electrical Engineering ISTET'97*, Palermo June 1997.

[5] J.W.Bandler, A.E.Salama: FAULT DIAGNOSIS OF ANALOG CIRCUITS. *Proc. IEEE*, August 1985, vol.73 n.8.

[6] H.T.Sheu, Y.H.Chang: HIERCHICAL FREQUENCY-DOMAIN ROBUST COMPONENT FAILURE DETECTION SCHEME FOR LARGE SCALE ANALOGUE CIRCUITS WITH COMPONENT TOLERANCES. *Proc. IEEE – Circuits Devices Syst.*,Februar 1996, vol.143 n.1

[7] J.L.Huertas: TEST AND DESIGN FOR TESTABILITY OF ANALOG AND MIXED-SIGNAL INTEGRATED CIRCUITS: THEORETICAL BASIS AND PRAGMATICAL APPROACHES. *Tutorial, European Conf. on Circuit Theory and Design ECCTD'93*, Davos July 1993.

[8] F. Filippetti, M. Martelli: NETWORK SENSITIVITY ANALYSIS OF LARGE CIRCUITS IN SYMBOLIC FORM. *European Conf. on Circuit Theory and Design ECCTD'95*, Istanbul Turkey August 95.

[9] F. Filippetti, M. Martelli: SYMBOLIC TECHNIQUES ADDRESSED TO THE FAULT DIAGNOSIS OF LAGE LINEAR ELECTRIC CIRCUITS. *IEEE Conf. on Industrial Applications in Power Systems Computer Science and Telecomunications MELECON'96*, Bari Italy May 1996.

[10] R.Voorakaranam, S.Charabarti, J.Hou, A.Gomes, S.Cherubal, A.Chatterjee, W.Kao: HIERARCHICAL SPECIFICATION-DRIVEN ANALOG FAULT MODELING FOR EFFICIENT FAULT SIMULATION AND DIAGNOSIS. *IEEE International Test Conference*, 1997.

[11] M. Artioli, F. Filippetti: LINEAR ANALOG CICUIT DIAGNOSIS BASED ON SYMBOLIC ANALYSIS AND REDUCED OBSERVABLE POINTS SET. *Int. Symp. on Theoretical Electrical Engineering ISTET'99*, Magdeburg September 1999.

[12] G.Fedi, R.Giomi, A.Luchetta, S.Manetti, M.C.Piccirilli: SYMBOLIC ALGORITHM FOR AMBIGUITY GROUP DETERMINATION IN ANALOG FAULT DIAGNOSIS. Proc. European Conf. on Circuit Theory and Design ECCTD'97, Budapest September 1997

[13] G.Fedi, R.Giomi, A.Luchetta, S.Manetti, M.C.Piccirilli: ON THE APPLICATION OF SYMBOLIC TECHNIQUES TO THE MULTIPLE FAULT LOCATION IN LOW TESTABILITY ANALOG CIRCUITS. *IEEE Transactions on Circuits and Systems,* October 1998, vol.45 n.10.

[14] M.W.Tian, C.J.R. Shi: RAPID FREQUENCY-DOMAIN ANALOG FAULT SIMULATION UNDER PARAMETER TOLERANCES. *Design Automation Conference DAC'97*, Anaheim June 1997.

2 Part Two – Multifrequency testing

Part Two discusses some symbolic procedures for improving the fault diagnosis of linear analogue circuits. The procedure, based on IME, belongs to the SAT (simulation after test) diagnostic techniques, and concerns the parameter identification approach and the fault verification approach. An Example is presented by using a simple network to show in detail the feasibility of the procedure.

2.1 Introduction

The fault diagnosis in analogue circuits, in particular the fault detection and location, can be developed using several techniques [1], all involving the circuit simulation, both the simulation before test approach (SBT) and the simulation after test approach (SAT).
In these techniques the circuit simulation plays a very important role but often needs of an excessive CPU time as the circuit must be simulated many times in correspondence of different fault situations.
The symbolic solution of the circuit, that usually can be obtained only for small circuits, allows us to overcome these difficulties.
Many symbolic techniques are developed to analyse linear circuits [2-4]. IME easily allows us to get the symbolic responses also of large linear electric circuits [5,6,7, Chapter II and III]. Besides, if a circuit response to an assigned system of sources is known, IME allows us to get the response to any different system of sources simply working up a succession of expressions corresponding to the first circuit analysis; therefore a further circuit analysis is not needed.
IME method features allow to easy realise diagnostic tasks utilising symbolic elaborations: in particular by combining the use of the *substitution theorem* [8] and of the *multifrequency testing*.
But the question of resources amount (in terms of required hardware and software) still remains:, e.g.:a 100×100 matrix representing an electrical linear network is quite usual, but its completely symbolic inversion was unsuccessful on a 486DX computer, equipped with 8Mb RAM and a popular calculation package.
The IME method [7,9], like other methods [2,3,4], seems to allow flexibility and reliability in reaching symbolic solutions. It is based on a simple calculation rule and allows to split the analysis problem into a sequence of hierarchical elementary circuit analyses, which can be solved separately reducing the computation complexity and consequently the requested amount of memory. These features avoid the well known problems arising in case of symbolic analysis.
The comparison with a classical inversion method (CME): the minors method, in term of memory employment (static, dynamic and total memory) is reported in the next table, for an $n \times n$ matrix.

Table 5:CME and IME Memory allocation.

	CME	IME
Static	n^2	$\frac{n^2(n+3)}{2}$
Dynamic	$\frac{n(n-1)(2n-1)}{6}$	0
Total	$\frac{n(n+1)(2n+1)}{6}$	$\frac{n^2(n+3)}{2}$
Allocations	$n!$	$3n$

The comparison in terms of computing time is reported in Fig. 2. The performance in solving a circuit under several percentage of symbolic elements,(from a completely numerical circuit elements 0%, gradually to a completely symbolic ones 100%) is clearly depicted. It is clear that IME method becomes very suitable as the symbolic elements are increasing.

2.2 Diagnostic procedure

Starting from the IME method ability to provide any requested response of a circuit in complete symbolic form, improvements can be obtained in the diagnostic techniques.
In particular the IME method has been used to perform the network sensitivity analysis of large circuits in [7], and to improve the procedures based either on the fault dictionary, or on the parameters identification by using the substitution theorem in [8].

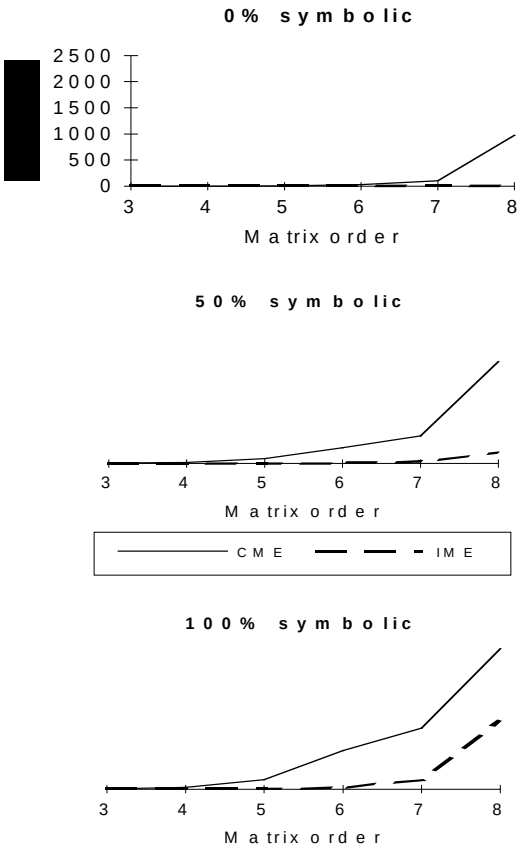


Fig. 31:Speed performance of IME and CME.

About the last application we have to emphasise some drawbacks. In fact all the voltages (or all the currents in the case of mesh analysis) must be observable to detect a possible fault related to any circuit component, this is not the best way to face a diagnostic problem.
On the contrary if we restrict the attention on the localisation of the fault, only few circuit values are needed.
The symbolic linear equations that the IME method provides, look like the following [8]:

$$\begin{aligned} V_1 &= F_1 (I_1, I_2, I_3, ...) \\ V_2 &= F_2 (I_1, I_2, I_3, ...) \\ V_3 &= F_3 (I_1, I_2, I_3, ...) \\ &..... \end{aligned} \qquad \text{(Eq. 1)}$$

where the V are the measured voltages and the I are the current sources simulating the considered faults. If we find, from the solution of the linear system (Eq. 1), that the current I_2 is unequal to 0, this means that the fault is localised on the branches convergent to the node 2. Therefore in order to localise a faulted branch or some possible faulted branches (a faulted zone), we need to know only the voltages of the nodes in which current sources are inserted.
After getting this result, to determine the exact faulted branch (or the exact faulted branches) and to evaluate the perturbed values of the faulted components, the multifrequency testing method can be used with reduced effort. In fact the multifrequency testing method, by using for example as test point the node 1, provide symbolic non linear equations that look like the following [8]:

$$
\begin{aligned}
& V_1(\omega_1) = M_1\,(p, \omega_1\,) \\
\text{(Eq. 2)} \qquad & V_1(\omega_2) = M_2\,(p, \omega_2\,) \\
& V_1(\omega_3) = M_3\,(p, \omega_3\,) \\
& \ldots\ldots\ldots\ldots
\end{aligned}
$$

where *p* represents the parameters vector. Obviously the equation number, and consequently the number of test points or test frequencies [10] depends on the parameters vector order. As higher is the order of *p* as harder is the solution of the non-linear system (Eq. 2).
If the faulted zone is known, it is possible to replace the parameters out of the faulted zone with the numerical nominal values, and to leave as unknowns the parameters into the faulted zone. In this way the solution of (Eq. 2) is certainly easier and quite immediate if only one parameter is unknown [10].
Therefore a good idea is to use IME method as depicted in [8] to localise the fault zone, and then to use again the IME features to provide the system (Eq. 2) in order to apply the multifrequency testing method in reduced form, as previously explained. The following application example shows this technique.

2.3 Example

As an example, the simple circuit already considered in [1,8] has been chosen (Fig. 32). Let's suppose the faulted zone include only the node 2 and Vo be the observed variable. Recalling the previous considerations we have to consider as unknowns the parameters R3, R4, C2 and C3 pertaining to the node 2, and we have to assign the nominal values to the unfaulted components R1, C1, R2.

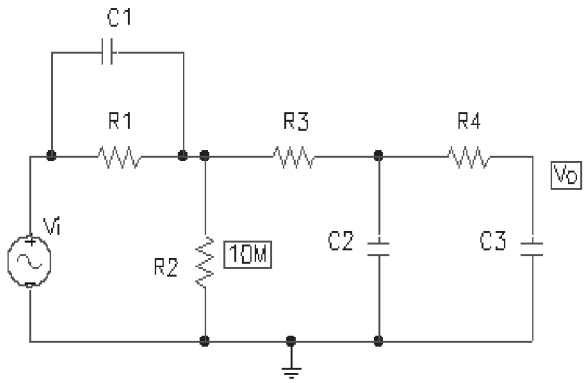


Fig. 32: Sample circuit.

To perform the diagnosis by applying the multifrequency testing method the following steps are needed:

1) A circuit analysis in the Laplace domain to provide the transfer function of the observed voltage Vo in symbolic form;

2) Construction of the system (Eq 2). Due to the presence of four unknowns and one test point, we have to perform two measurements at two different frequencies $V_o(\omega_1)$ and $V_o(\omega_2)$. The corresponding equations

$$V_o(\omega_1) = M_1 (p, \omega_1)$$
(Eq. 3) $$V_o(\omega_2) = M_2 (p, \omega_2)$$

can be split into the Real and Imaginary part, giving the four non-linear equations needed to evaluate the 4 unknowns.
The nominal values of our test circuit are $R_1 = R_4 = 1M\Omega$, $R_2 = 10M\Omega$, $R_3 = 2M\Omega$, $C_1 = 0.01\mu F$, $C_2 = C_3 = 0.001\mu F$. The chosen test frequencies are 10 and 200 rad/sec. At these frequencies for +30-percent deviations in the circuit element R_3 and C_2, from the measurements we have obtained:

$$V_o(10) = 0.902 - 0.073\ I$$
$$V_o(200) = 0.318 - 0.430\ I$$

Therefore the system (Eq. 3) becomes:

$$+0.902 = Re[M_1 (p, 10)]$$
$$-0.073 = Im[M_2 (p, 10)]$$
$$+0.318 = Re[M_3 (p, 200)]$$
$$-0.430 = Im[M_4 (p, 200)]$$

The equations automatically provided by IME are here reported in Mathematica format:

```
(0.90190330342889 == (10000000*
      (11000000 - 1.*10^7*C2*R3 - 1.*10^7*C3*R3 - 1.*10^7*C3*R4 -
        100000000000000*C2*C3*R4 - 1100000000*C2*C3*R3*R4))/
    ((11000000 - 1.*10^7*C2*R3 - 1.*10^7*C3*R3 - 1.*10^7*C3*R4 -
        100000000000000*C2*C3*R4 - 1100000000*C2*C3*R3*R4)^2 -
      (1.*10^6*I + 10000000000000*I*C2 + 10000000000000*I*C3 +
        110000000*I*C2*R3 + 110000000*I*C3*R3 + 110000000*I*C3*R4 -
        1.*10^8*I*C2*C3*R3*R4)^2) -
   (1.*10^6*(-(1.*10^6) - 10000000000000*C2 - 10000000000000*C3 -
       110000000*C2*R3 - 110000000*C3*R3 - 110000000*C3*R4 +
       1.*10^8*C2*C3*R3*R4))/
    ((11000000 - 1.*10^7*C2*R3 - 1.*10^7*C3*R3 - 1.*10^7*C3*R4 -
        100000000000000*C2*C3*R4 - 1100000000*C2*C3*R3*R4)^2 -
      (1.*10^6*I + 10000000000000*I*C2 + 10000000000000*I*C3 +
        110000000*I*C2*R3 + 110000000*I*C3*R3 + 110000000*I*C3*R4 -
        1.*10^8*I*C2*C3*R3*R4)^2),
 -0.07336423102824995 == (1.*10^6*
      (11000000 - 1.*10^7*C2*R3 - 1.*10^7*C3*R3 - 1.*10^7*C3*R4 -
        100000000000000*C2*C3*R4 - 1100000000*C2*C3*R3*R4))/
    ((11000000 - 1.*10^7*C2*R3 - 1.*10^7*C3*R3 - 1.*10^7*C3*R4 -
        100000000000000*C2*C3*R4 - 1100000000*C2*C3*R3*R4)^2 -
      (1.*10^6*I + 10000000000000*I*C2 + 10000000000000*I*C3 +
        110000000*I*C2*R3 + 110000000*I*C3*R3 + 110000000*I*C3*R4 -
        1.*10^8*I*C2*C3*R3*R4)^2) +
   (10000000*(-(1.*10^6) - 10000000000000*C2 - 10000000000000*C3 -
       110000000*C2*R3 - 110000000*C3*R3 - 110000000*C3*R4 +
       1.*10^8*C2*C3*R3*R4))/
    ((11000000 - 1.*10^7*C2*R3 - 1.*10^7*C3*R3 - 1.*10^7*C3*R4 -
        100000000000000*C2*C3*R4 - 1100000000*C2*C3*R3*R4)^2 -
      (1.*10^6*I + 10000000000000*I*C2 + 10000000000000*I*C3 +
        110000000*I*C2*R3 + 110000000*I*C3*R3 + 110000000*I*C3*R4 -
        1.*10^8*I*C2*C3*R3*R4)^2),
 0.3183183203701878 == (10000000*
      (11000000 - 4.*10^9*C2*R3 - 4.*10^9*C3*R3 - 4.*10^9*C3*R4 -
        400000000000000000*C2*C3*R4 - 4400000000*C2*C3*R3*R4))/
    ((11000000 - 4.*10^9*C2*R3 - 4.*10^9*C3*R3 - 4.*10^9*C3*R4 -
        400000000000000000*C2*C3*R4 - 4400000000*C2*C3*R3*R4)^2 -
      (2.*10^7*I + 200000000000000*I*C2 + 200000000000000*I*C3 +
        2200000000*I*C2*R3 + 2200000000*I*C3*R3 + 2200000000*I*C3*R4 -
        8.*10^11*I*C2*C3*R3*R4)^2) -
   (2.*10^7*(-(2.*10^7) - 200000000000000*C2 - 200000000000000*C3 -
       2200000000*C2*R3 - 2200000000*C3*R3 - 2200000000*C3*R4 +
       8.*10^11*C2*C3*R3*R4))/
    ((11000000 - 4.*10^9*C2*R3 - 4.*10^9*C3*R3 - 4.*10^9*C3*R4 -
        400000000000000000*C2*C3*R4 - 4400000000*C2*C3*R3*R4)^2 -
      (2.*10^7*I + 200000000000000*I*C2 + 200000000000000*I*C3 +
        2200000000*I*C2*R3 + 2200000000*I*C3*R3 + 2200000000*I*C3*R4 -
        8.*10^11*I*C2*C3*R3*R4)^2),
 -0.43017487687401 == (2.*10^7*
      (11000000 - 4.*10^9*C2*R3 - 4.*10^9*C3*R3 - 4.*10^9*C3*R4 -
        400000000000000000*C2*C3*R4 - 4400000000*C2*C3*R3*R4))/
    ((11000000 - 4.*10^9*C2*R3 - 4.*10^9*C3*R3 - 4.*10^9*C3*R4 -
        400000000000000000*C2*C3*R4 - 4400000000*C2*C3*R3*R4)^2 -
      (2.*10^7*I + 200000000000000*I*C2 + 200000000000000*I*C3 +
        2200000000*I*C2*R3 + 2200000000*I*C3*R3 + 2200000000*I*C3*R4 -
        8.*10^11*I*C2*C3*R3*R4)^2) +
   (10000000*(-(2.*10^7) - 200000000000000*C2 - 200000000000000*C3 -
       2200000000*C2*R3 - 2200000000*C3*R3 - 2200000000*C3*R4 +
       8.*10^11*C2*C3*R3*R4))/
    ((11000000 - 4.*10^9*C2*R3 - 4.*10^9*C3*R3 - 4.*10^9*C3*R4 -
        400000000000000000*C2*C3*R4 - 4400000000*C2*C3*R3*R4)^2 -
      (2.*10^7*I + 200000000000000*I*C2 + 200000000000000*I*C3 +
        2200000000*I*C2*R3 + 2200000000*I*C3*R3 + 2200000000*I*C3*R4 -
        8.*10^11*I*C2*C3*R3*R4)^2))
```

The Laplace variable *s* has been replaced by I*ω and then ω by the corresponding test frequencies 10 and 200 rad/sec. About the parameter vector *p* the elements out of the faulted zone (R_1 C_1 R_2) have been replaced by their nominal values, remaining as symbols the other elements.

2.4 Conclusions for Part Two

A symbolic and semi-symbolic technique applied to the fault diagnosis of linear electric circuits has been discussed. This technique, based on the circuit analysis method IME, is pertinent to the parameter identification approach.
The generation of suitable symbolic equations through IME is easy for wide circuits too, while solving them can be very hard in this case, due to the non-linearity of the expressions and the high number of unknowns. Thus, an application of this diagnostic technique based on the substitution theorem allows first to restrict the faulted zone, and then the multifrequency testing method can be applied with a reduced number of unknowns providing an efficient way to determine the perturbed values of the faulted components.
A simple example is developed in detail, to show the features and the improvement obtained by utilising this technique

2.5 References to Part Two

[1] J.W. Bandler, A. Salama: FAULT DIAGNOSIS OF ANALOG CIRCUITS - *Proc.IEEE*, August 1985, vol.73,n.8, pag 1279/1325.

[2] A. Liberatore, S. Manetti: SAPEC A PERSONAL COMPUTER PROGRAM FOR THE SYMBOLIC ANALYSIS OF ELECTRIC CIRCUITS, *1988 IEEE ISCAS*, Helsinki, June 1988.

[3] A. .Liberatore, S. Manetti: NETWORK SENSITIVITY ANALYSIS VIA SYMBOLIC FORMULATION, *1989 IEEE ISCAS*, Portland, June 1989.

[4] M. M. Hassoun, P. M. Lin: A NEW NETWORK APPROACH TO SYMBOLIC SIMULATION OF LARGE SCALE NETWORKS, *1989 IEEE ISCAS*, Portland, June 1989.

[5] F. Filippetti, M. Martelli: A METHODOLOGY FOR THE SYMBOLIC ANALYSIS OF LINEAR ELECTRIC CIRCUITS, *Jornadas Hispano Lusas de Ingenieria Electrica*, July 1990, Vigo (Spain).

[6] F. Filippetti, M. Martelli: SYMBOLIC TECHNIQUES TO GENERATE THE EQUATIONS OF REPEATING STRUCTURE CIRCUITS, *SMACD'94*, October 1994, Sevilla (Spain).

[7] F. Filippetti, M. Martelli: NETWORK SENSITIVITY ANALYSIS OF LARGE LINEAR CIRCUITS IN SYMBOLIC FORM, *ECCTD '95*, August 1995, Istanbul (Turkey)

[8] F. Filippetti, M. Martelli: SYMBOLIC TECHNIQUES ADDRESSED TO THE FAULT DIAGNOSIS OF LARGE LINEAR ELECTRIC CIRCUITS, *IEEE Conf. on Industrial Applications in Power Systems Computer Science and Telecomunications MELECON'96*, Bari Italy May 1996.

[9] F. Ciampolini: A METHOD TO ANALYZE LINEAR ANALOG CIRCUITS (In Italian), *L'Elettrotecnica* N. 10, Oct. 1963.

[10] J. L. Huertas, TEST AND DESIGN FOR TESTABILITY OF ANALOG AND MIXED-SIGNAL INTEGRATED CIRCUITS: THEORETICAL

BASIS AND PROGRAMMATICAL APPROACHES *Selected Topics in Circuits and Systems*, Elsevier Science Publishers 1993.

Chapter IV

Reflection on circuit analysis

Starting from the same basic idea, that is, particularizing a general symbolic expression imposing definite transformation rules, two circuit analysis procedure are proposed. The first one for switching circuits and the second one for piecewise-linear circuits.

1 Part One - A procedure for switching circuits

Part One is intended to show the feasibility of implementing symbolic techniques for the analysis and design of switching circuits. In particular, the symbolic approach, based on symbolic analysis in the Laplace transform domain, is applied to DC-DC switching converters. Several topologies, including resonant ones [1], are considered as examples showing the method implementation and its related features and advantages.
To study switching circuits, one needs analysis methods even more efficient and flexible. Several time-domain analysis methods have been proposed, for example in [2-9], and recent versions of PSPICE allow switched networks with non-ideal switches to be treated, leading sometimes to non-accurate solutions and/or convergence problems.
In general, the above methods are acceptable for some switched networks but they suffer of the most common problems of such an approach:

- discontinuities at the switching instants and possible Dirac impulses are not considered;
- difficulty of implementing circuit equations in computer programs;
- restriction to only one topology change at each switching instant.

More general drawbacks are:

- the switching times of internally controlled switches must be determined;
- the correct topology after switching must be determined.

The aim of this chapter is to propose a general analysis method free from the above mentioned limitations.
The method is based on a symbolic processor which can determine any circuit response in a completely symbolic form, and also on the intrinsic behaviour of DC-DC converters. A large-signal time-domain analysis for this kind of circuits is based on these facts:

- at any given time each switch is in ON or OFF state and the network contains only linear components arranged in a corresponding topology;
- the equations for each topology are integrated until an event causes a state change for one or more switches;
- eventually the topology is updated and the procedure is cyclically repeated till a specified time or the occurrence of specified conditions.

The circuit behaviour is usually well known and there should not be any problem to individuate the topological configurations of all ON-OFF combinations of the switches. As it will be explained in the following subsections, this idea of "configuration" can be extended, using the proposed method, to the occurrence of internal events which determine a substantially different behaviour of the network, even if they do not determine a topology change. This new situation can have the same topology of the previous one, but with different parameter characteristics. In this sense, the proposed method can be addressed as an "event-driven method".

1.1 Symbolic technique

After each commutation or, more generally, after each proper event, a configuration change occurs, so that the behaviour of the network can be seen as a sequence of circuit configurations. Each configuration can be handled as a linear circuit with initial conditions given by the final values assumed in the previous configuration [2,6,7]. Thus, it is possible a symbolic approach for this kind of converters by means of a suitable symbolic processor [10]. Through symbolic techniques, one can automatically get a mathematical description for each circuit configuration in the Laplace transform domain [11,12], and, once the symbolic processor has solved each configuration equations, the behaviour of the whole converter can be studied and designed through the method explained in the next subsection.
In Fig. 33, the popular buck converter is shown.

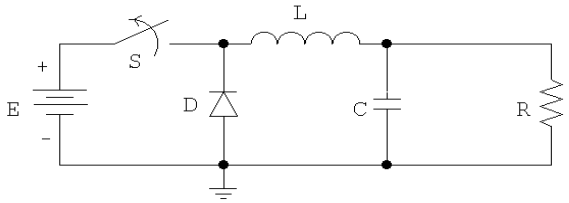


Fig. 33: Ideal buck converter.

In this circuit there are two switches: the external one S and the internal one D, and four configurations are possible. From the initial condition where S is ON and D OFF, when S is switched OFF the voltage polarity across L changes and D is switched ON in accordance to it. Definitely, only the two configurations of Fig. 34 and Fig. 35 are involved in the circuit behaviour. We remember, of course, that all parameters in these schemes are ideal. For this circuit in the configuration, shown in Fig. 34, the symbolic processor gives the following symbolic expression for the inductor current in the *s*-domain:

$$I_L(s) = \frac{E}{s}\frac{(1+CRs)}{(R+Ls+CLRs^2)} + \frac{Li_{L0}(1+CRs)}{R+Ls+CLRs^2} - \frac{v_{C0}CR}{R+Ls+CLRs^2}$$

where i_{L0} and v_{C0} are the inductor current and capacitor voltage just before S is turned ON.

We emphasise that through the symbolic analysis it is easy to put in evidence the most relevant terms in an expression: in our example the forced response term and the natural response terms are separated. The same expression is still valid for the configuration of Fig. 35 as well. As a matter of fact, the two configurations are similar, except for the voltage source *E*, that can be easily set to zero by the symbolic processor.

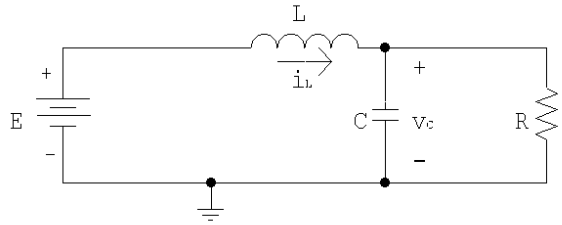


Fig. 34: Buck converter when the switch S is ON.

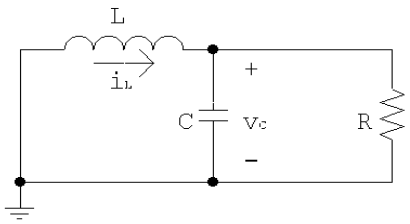


Fig. 35: Buck converter when the switch S is OFF.

1.2 Proposed method

The concept is to monitor the conditions for ON/OFF states of the switches and use the linear model accordingly, to provide segments of responses and build the circuit evolution. Model updating is performed automatically after each segment, whose length is pre-set. The proposed method is summarised in the basic scheme shown in Fig. 36. Starting from a known initial configuration, the circuit response is computed from symbolic solutions. At the switching instant, the actual values of the variables i_L and v_C become the initial values of the new situation. Switching can occur due to external commands or internal events, i.e., a switching element could switch as a consequence of a previous switching. Both cases can be managed by rules suitably formalised and related to the switching strategy: for example, a state change when a current is zero, if we are dealing with a kind of resonant converter.

Because of the limited number of possible configurations and well known behaviour of DC-DC converter circuits, no general technique has been introduced to move from a configuration to another one. We referred, instead, to rules that link the different configurations for each circuit studied, but it is possible to implement general techniques [9-11] as well, that allow one to find the configuration following any previous one as a result of a switching event, without any "*a priori*" knowledge of the circuit evolution.

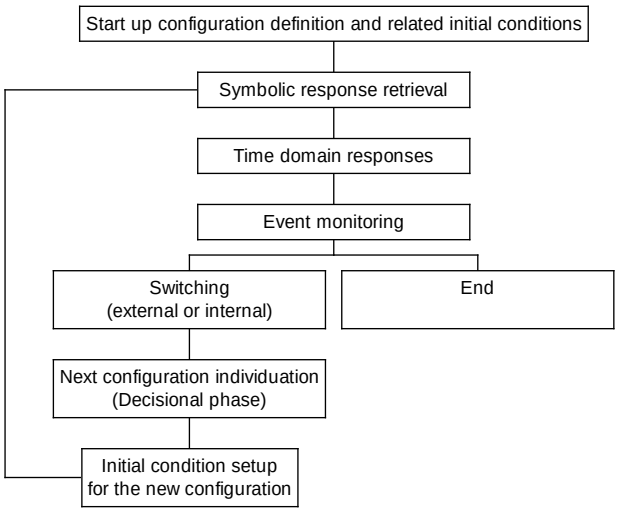


Fig. 36: Basic scheme of the proposed method.

Since the circuit transition to a new configuration can be caused by switching or by an intrinsic event (like a voltage that becomes zero) a variable monitoring is performed to ascertain whether some rules are violated or not and then to process the next situation. With this approach the buck converter was examined and some of the results are depicted in Fig. 37 for the circuit with the following parameter values, already reported in [11]: E = 201 V, L = 3 mH, C = 4.2 μF, R =20 Ω, switching frequency f_s = 40 kHz, duty cycle = 0.4.

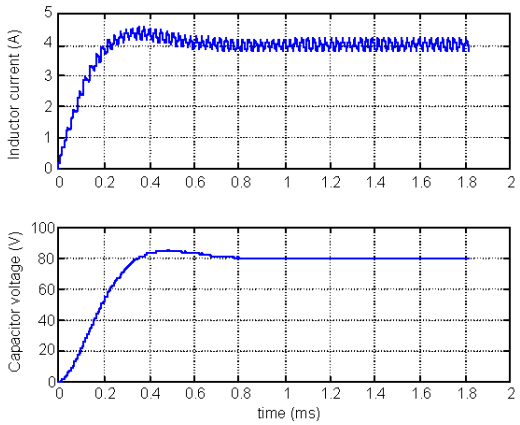


Fig. 37: Buck converter waveforms for inductor current and capacitor voltage.

1.3 Advantages

The symbolic processor (in particular the IME processor [14,15, Chapter II]) allows us to generate and manipulate expressions. These capabilities produce several advantages:

1. the method uses linear techniques only and therefore a solution is always reached;
2. prediction of the next configuration by monitoring that suitable electrical signals verify particular conditions; this decisional step is performed by activating an inferential engine based on a set of production rules related to the circuit behaviour: this leads to a simple approach to the circuit;
3. symbolic expressions allow one to easily study the influence of parameter variations, simulating the circuit only by substituting the new values;
4. semi-symbolic expressions can be automatically generated, where only the most relevant parameters are left in symbolic form to get simple constitutive relationships; for example for design purposes, parasitic elements can be introduced to study their influence;
5. nonlinearities can be introduced with PWL characteristics (in this case the "event" is the transition across two different linearity regions) or with parameters that gradually vary while the circuit switches towards another configuration.

To highlight the features of the method, some cases are presented herein. In particular, Fig. 38 shows the waveforms of the inductor currents and capacitor voltage when the inductance is changed from 3 mH to 2 mH and to 1 mH: as the inductance decreases, the current ripple gets bigger and the voltage ripple across the capacitor increases accordingly.

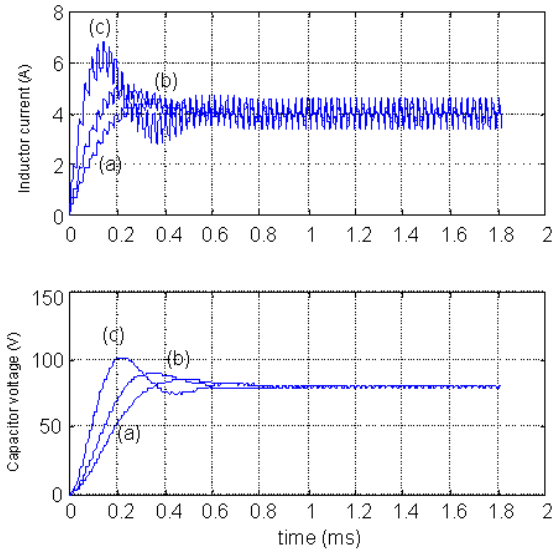


Fig. 38: Buck converter waveforms when the inductance is set to the values 3 mH (a), 2 mH (b) and 1 mH (c).

Then, we can consider the possibility that L is a saturable inductor with the characteristic given in Fig. 39. Now, Fig. 40 shows inductor currents and capacitor voltages when the duty cycle is 0.2, 0.5 and 0.8, and when the capacitance is 500 nF highlighting the saturation effect. We notice that a duty cycle approaching unity produces an evident ripple in the inductor current which is reflected also on a higher ripple in the voltage across the capacitor.

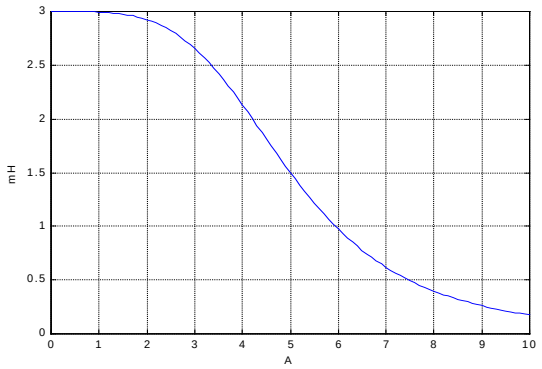


Fig. 39: Saturable inductor characteristic.

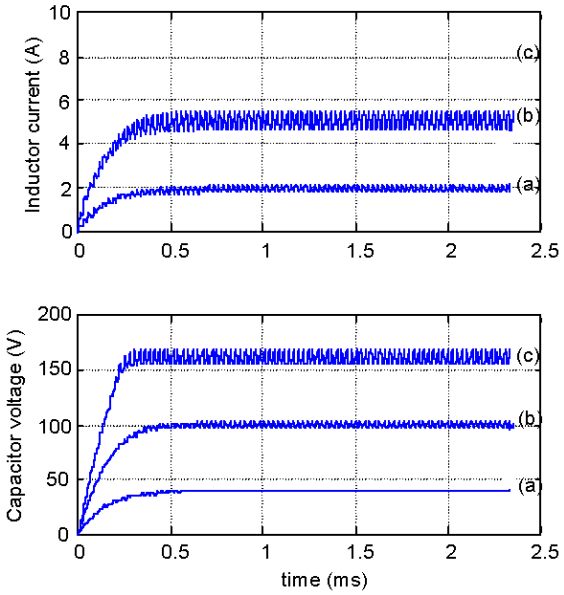


Fig. 40: Buck converter waveforms when L is a saturable inductor and the duty cycle is 0.2 (a), 0.5 (b) and 0.8 (c).

Let's consider now the boost converter (Fig. 41), whose parameter values are E = 5 V, L = 50 μH, C = 100 μF, R =15 Ω, f_s = 50 kHz, duty cycle = 0.5, as used in [6]. Fig. 42 shows the waveforms of the inductor current and capacitor voltage in a particular case, that is for a step change of the input voltage E from 5 V to 6 V about 3 ms after start-up.

We emphasise that this method allows circuit analysis either in completely ideal situations or with real components, and that all these points are helpful for the optimal circuit design according to the developed switching strategy. In fact, it is possible to simulate the circuit with totally ideal components at first. Then, introducing parasitic elements, non-linearities, etc., and using the same symbolic solution, one can model the circuit in a more real way.

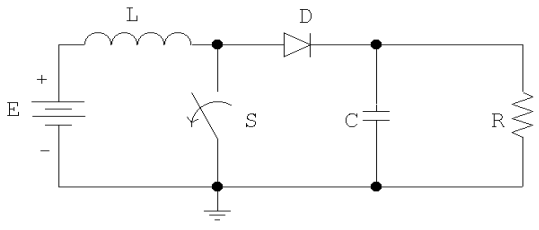


Fig. 41: Ideal boost converter.

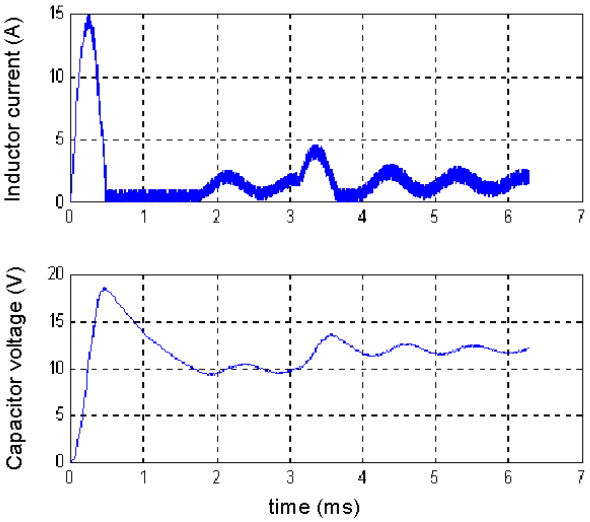


Fig. 42: Boost converter waveforms for a step change of the input voltage.

1.4 Resonant topologies

The great flexibility of the proposed method allows more complex analyses to be performed in an efficient way. For example, we can suppose to add a resonant switch to the base buck converter. In Fig. 43 we report an L-type half-wave resonant DC-DC converter, where the capacitor between L_2 and R has been removed for simplicity reasons. The chosen parameters are: E = 201 V, L_1 = 2.5 µH, L_2 = 3 mH, C = 18 nF, R = 10 Ω, f_s = 500 kHz, duty cycle = 0.4. The values for C, L_1 and for the duty cycle are chosen so that the current through L_1 is zero at the external switching instant, that is, the circuit works in resonant mode. For this circuit, the preliminary analysis shows that four topological configurations are of concern, but, for shortness, we report only the discharge phase configuration, when the load current is sustained by the capacitor C. In the Laplace transform domain this configuration is depicted in Fig. 44.

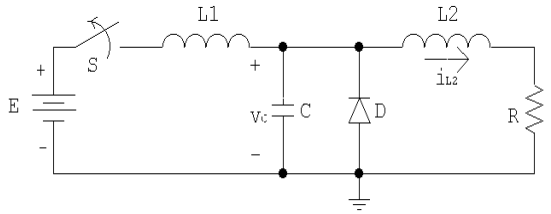


Fig. 43: L-type half-wave resonant DC-DC converter.

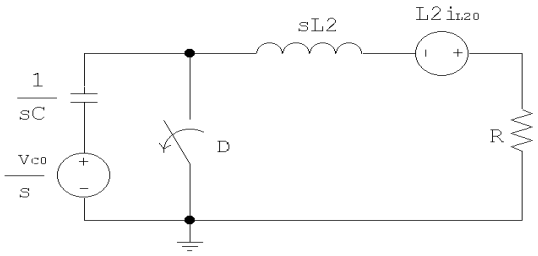


Fig. 44: Discharge phase configuration in the Laplace transform domain.

For this circuit the capacitor voltage in symbolic form is:

$$V_c(s) = -\frac{v_{C0}}{s}\frac{1}{1+CRs+CL_2s^2} - \frac{L_2 i_{L_2 0}}{1+CRs+CL_2s^2}$$

where v_{C0} and i_{L20} are the values of the current through L_2 and the voltage across C at the end of the previous resonant phase. This response allows us to monitor when the capacitor C will be discharged to determine the switching time of the diode. During the decisional phase, the rule that activates this response segment is of the type:

IF *Vc* > 0 & S_state = OFF THEN discharge phase.

The related implementation problem will be discussed in the next subsection. Some results for the considered resonant converter are reported in Fig. 45, Fig. 46 and Fig. 47.

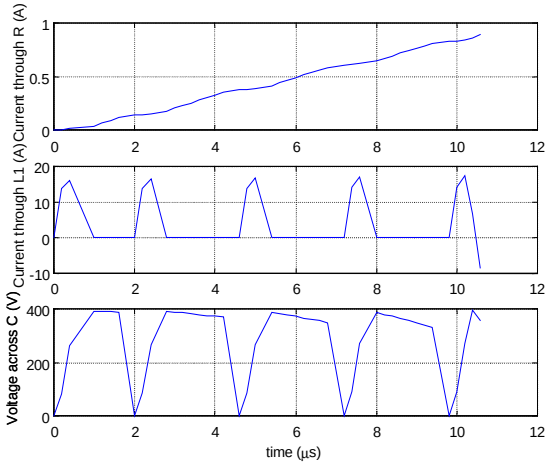


Fig. 45: Beginning of the start-up phase for the resonant buck converter.

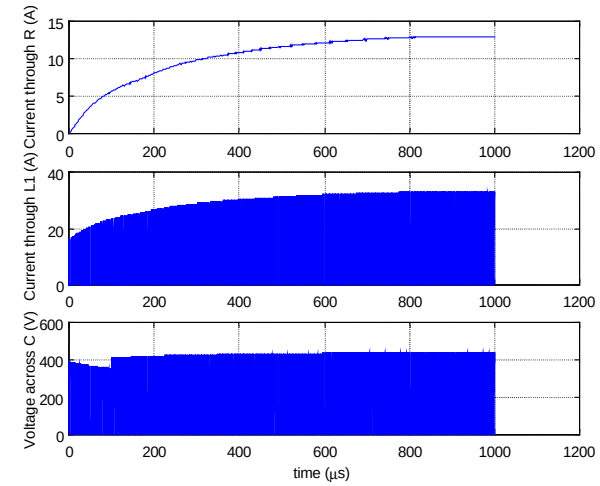


Fig. 46: Waveforms for the resonant buck converter, from power-on until steady state is reached.

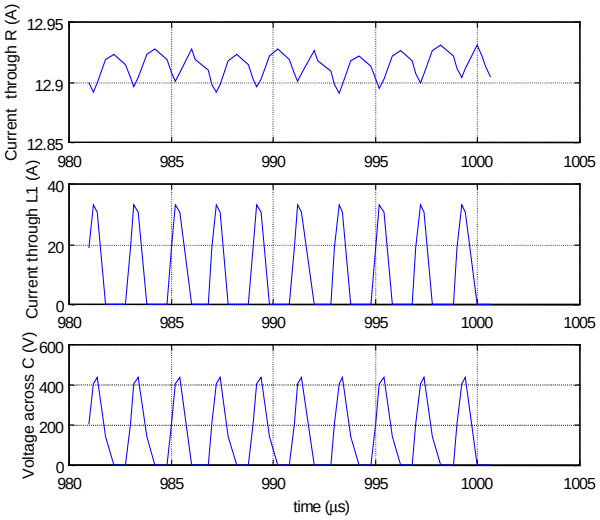


Fig. 47: Steady state waveforms for the resonant buck converter.

During the start-up phase (Fig. 45) it is noticeable that the zero instants of the current through inductor L_1 are not equal spaced because the circuit has not reached the complete resonant condition yet. As Fig. 46 shows, after 800 µs the steady state in complete resonant condition is reached, as depicted in detail in Fig. 47, where periodic waveforms are reported.

1.5 The implementation of the proposed method

The proposed method can be applied to any kind of switching circuit. We applied it to the resonant buck converter of Fig. 43, to test the reliability of the event driven approach, due to the relative "hardness" in treating resonant circuits.
The procedure has been developed in a Mathematica™ environment, thus it was written in a C-like form. The program implementation follows the simple scheme reported in Fig. 36. After we have stored the symbolic solution in the system memory (in mass memory devices as well, if the circuit is really huge) and after the initialisation block, a loop begins. Inside the loop, until the simulation end time is reached, this job is done:

- the time response is calculated by the inverse Laplace transform of the symbolic solution for the actual configuration;
- this time response is scanned at the sample times to check whether a rule becomes satisfied. This is the decisional block: the next configuration is selected depending on the event occurrence and every event, the external ones as well, is checked through rules;
- the new configuration and its initial conditions are set and the output is updated, reporting the valid time response segment between the configuration change instants.

The procedure cannot be considered "a tool" yet, because, from the software point of view, it is not completely automated and integrated as PSPICE is, even if simulations times are comparable with those needed by PSPICE. In perspective, the method implementation will be made more flexible and capable of easy including any kind of rule.

1.6 Conclusions for Part One

In Part One, a flexible methodology has been proposed, trying to alleviate some troubles concerning power converter simulation, as well as convergence problems. The procedure has been tested on several DC-DC converter topologies, including resonant ones. The results seem to be satisfactory because the method always guarantees a solution with simulation times comparable to PSPICE simulations, and due to the fact that it is possible, using the same symbolic solution, to move from an ideal circuit analysis gradually to a real circuit analysis. At the moment the procedure cannot be considered as a complete circuit simulation environment, but a contribution to develop tools for approaching a class of circuits, traditionally hard to treat, referring in particular to resonant circuits.

1.7 References to Part One

[1] A. Liberatore, A. Reatti, ABOUT RECENT SWITCHING TECHNIQUES FOR DC-DC CONVERTERS, *L'Energia Elettrica*, No. 12, 1989, pp. 601-620, (in Italian).

[2] M. K. Kazimierczuck, D. Czarkowski, RESONANT POWER CONVERTERS, *John Wiley & Sons, Inc*, New York, 1995.
[3] Opal, J. Wlach: CONSISTENT INITIAL CONDITIONS OF LINEAR SWITCHED NETWORKS *IEEE Trans. Circuits Syst. I*, vol. 37, pp. 364-372, Mar. 1990.
[4] Liberatore, S. Manetti, M. C. Piccirilli, A. Reatti, SIMULATION OF SWITCHING POWER CONVERTERS USING SYMBOLIC TECHNIQUES, *Alta Frequenza*, No. 6, 1993, pp. 16/304-23/311.
[5] R. J. Dirkman, THE SIMULATION OF GENERAL CIRCUITS CONTAINING IDEAL SWITCHES, *IEEE Power Electron. Special. Conf.*, 1987, pp. 185-194.
[6] D. Bedrosian, J. Vlack, TIME-DOMAIN ANALYSIS OF NETWORKS WITH INTERNALLY CONTROLLED SWITCHES, *IEEE Trans. Circuits Syst. I*, vol. 39, pp. 199-212, Mar. 1992.
[7] A. Massarini, U. Reggiani, COMPUTER-AIDED TIME- DOMAIN LARGE-SIGNAL ANALYSIS OF NETWORKS WITH SWITCHES, *IEEE Int. Symp. on Industrial Electron*., ISIE'96, 1996, vol.2, pp. 567-572.
[8] A. Massarini, U. Reggiani,, M. K. Kazimierczuck, ANALYSIS OF NETWORKS WITH IDEAL SWITCHES BY STATE EQUATIONS, *IEEE Trans. Circuits Syst. I*,. vol. 44, pp. 692-697, Aug. 1997.
[9] R. M. Lamaison, A. Esquivel, J. Bordonau,, J. Peracaula, A GENERALIZED APPROACH TO OBTAIN THE STEADY-STATE CHARACTERISTICS OF DC/DC AND DC/AC RESONANT CONVERTERS, *Eur. Conf. on Power Electron. and Appl.*, EPE'99, 1999.
[10] A. Liberatore, S. Manetti, SAPEC A PERSONAL COMPUTER PROGRAM FOR THE SYMBOLIC ANALYSIS OF ELECTRICAL CIRCUITS, *IEEE Int. Symp. on Circuits and Systems*, ISCAS'88, 1988.
[11] M. Artioli, F. Filippetti,, M. Martelli, IMPROVEMENTS TO THE FAULT DIAGNOSIS OF LARGE LINEAR ELECTRIC CIRCUITS VIA IME METHOD, *Int. Symp. on Theoretical Electrical Engineering*, ISTET'97, 1997, pp. 156-159.
[12] M. Artioli, F. Filippetti, LINEAR ANALOG CIRCUIT DIAGNOSIS BASED ON SYMBOLIC ANALYSIS AND REDUCED OBSERVABLE POINT SET, *Int. Symp. on Theoretical Electrical Engineering*, ISTET'99, 1999, pp. 485-489.

2 Part Two - A procedure for piece-wise linear circuits

Part Two is intended to show an approach to directly translate linear methods to PWL circuit problems. The basic idea is to hide the non-linearity to the linear method, so that the user can handle the PWL problem as if it was a linear one. So doing writing equations for a PWL circuit can be extremely simple and fast, since the PWL formulation becomes a linear formulation. Moreover, since it is possible to use usual linear solvers, the solution is improved as well. This approach is based first on symbolic formulation and calculations, then on rules application, which deliver to the symbolic processor the task to deal with the PWL aspects of the problem. A Mathematica procedure dedicated to this aim will be suggested, together with a practical example where a four-diode rectifier is handled as a linear circuit.

2.1 Introduction

PWL techniques are often based on numerical computations or need to implement a kind of linear programming technique in order to reduce this amount of computations, even if the PWL circuit problem is, in some circumstances, ideally quite simple [1,2]. A symbolic approach to this problem could be interesting [3,4].
We assume to work with piecewise linearized characteristics, defined on the entire real axis and strictly monotone in each interval of the real axis obtained dividing it at each breakpoint of the linearized characteristic. Thus they are represented by continuous functions on the real axis, generally not derivable at the breakpoints but derivable and monotone between two consecutive breakpoints and between infinity and the nearest breakpoint.
In each interval every characteristic is linear, therefore the solution for each region of the solution space can be found (if it exists) combining linear equations. Furthermore, excluding pathological cases, the symbolic solution of a linear system always exists, so the basic idea is to deliver a part of the computation workload to a procedure which can find a general symbolic solution using linear tools and then to particularize it numerically or semi-symbolically as needed.
As a simple example we'll consider an ideal rectifier with four identical diodes, described by a seven-segment line, but we remember that symbolic solutions for relatively wide circuits can be calculated via IME method [5,6,7, Chapter II].

2.2 Symbolic approach

Mathematica is a commercial package whose built-in mathematical kernel is oriented to lists, rules applications and symbolic manipulations. This fact allows to directly handle problems in a more abstract way [5,6,7,8].

Since PWL circuits are basically linear circuits when considered on a single region of their dominion, the idea is to transfer a linear solving method to PWL circuits, hiding to the method this particular kind of non-linearity.
With this kind of symbolic "macro" solutions, calculated in a linear way, and with a symbolic processor capable of applying mathematical and logical rules to any kind of expression, is possible to take in account the non-linearity as well. That is, instead of finding all particular solutions (one per region) the suggested technique allows to determine a general symbolic solutions which can be easily and quickly particularized for each region.

2.3 Symbolic technique

To realize a procedure, using Mathematica features, the fundamental configuration steps are:

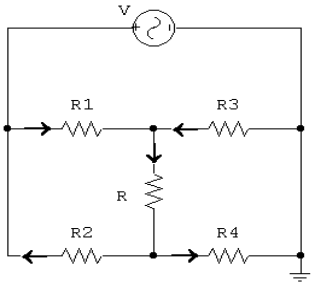
1. choice of the shape of the characteristic of each PWL component, that is describing each region on the characteristic plane with a couple of numbers: threshold and slope;
2. description of the complete set of characteristics by means of a list of Mathematica rules;
3. writing symbolic equations system of the circuit as if all components were linear. Each PWL component is denoted by a variable name, which will never contain the actual value of the parameter for that component, but acts as a simple placeholder in solving the system.

At this point the automatic Mathematica procedure starts and follows these steps:

1. expansion of each placeholder in couple of unique symbols which represent threshold and slope in a generic region for that component;
2. solution of the linear system, finding a unique general symbolic solution, if it exists;
3. generation of the regions of the PWL solutions space, obtained by crossing the regions of the characteristics. This results in sets of inequalities in several variables;
4. expressing these variables in terms of a single variable (i.e: current or voltage supply): this can be obtained by symbolical manipulation of the starting linear equations;
5. solution of the inequalities system with respect to this single variable: the result is that all regions are expressed in terms of the source variable;
6. elimination of all non-admissible regions and particularization of the general solution for every admissible region;
7. merging the particularized solutions with interval functions in a all-in-one Mathematica expression which represents the general symbolic solutions of the circuit, disregarding validity regions (except for the starting hypothesis, of course).

2.4 A practical example

Let's consider the very simple resistive network of Fig. 48 (parameter values are not reported because we are only interested in pure symbolic calculations).




```
v==-i2 R2+i4 R4
v== i1 R1 -i3 R3
v== i1 R1+(i1+i3)R+i4 R4
i1+i3==i2+i4
```

Fig. 48: Linear resistive circuit and related equations.

The network can be solved, for example, using branch currents (numbered like the resistors), by the system of equations in Fig. 48. This system can be written directly, due to the fact that all components are linear. If non-linear components are present, other electrical considerations are needed before writing down the system. For example: let's consider the four-diode rectifier shown in Fig. 49.
It has the same topology of the previous network, but there are four non-linear components (that we will represent with PWL devices) which have several combinations of on-off states. These configurations should have to be analysed before getting a symbolic solution (one or more expressions) in a traditional way.

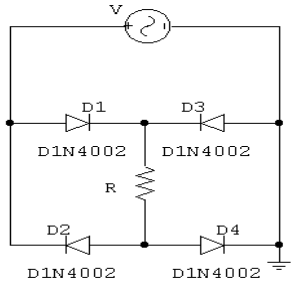
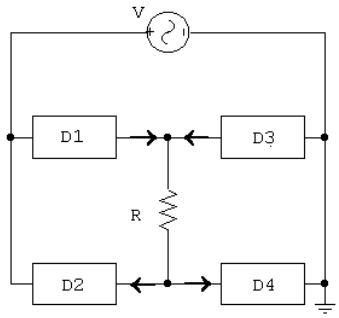


Fig. 49: Sample PWL circuit.

Through the proposed method it is possible to approach this PWL circuit as if it were a linear circuit. The non-linearity (PWL type, in our case) is collected and hidden in mathematical boxes, so that the circuits appears like in Fig. 50.




```
v==-i2 D2+i4 D4
v== i1 D1 -i3 D3
v== i1 D1+(i1+i3)R+i4 D4
i1+i3==i2+i4
```

Fig. 50: Circuit with hidden non-linearity and related pseudo-linear equations.

At this point it is possible to write down linear equations like in Fig. 50, where D1, D2, D3 and D4 are considered generic linear resistors. These equations are formally identical to those of Fig. 48.
The four diodes are of the same type and we will assume for them the same characteristic, which is represented by a list of points on the *i-v* plane. They are obtained from a PSPICE component characteristic, and they will be written in Mathematica as:

```
breakpoints={{-12*10^-3,-5},{0,0},{10^-3,691*10^-3},{20.5*10^-
3,728*10^-3},{250*10^-3,867*10^-3},{1,961*10^-3}}
```

These points are the breakpoints between segments which approximate the characteristic (step 1) shown in Fig. 51. The list of points is first automatically converted into a data structure for each diode, containing much more information, like: slope of the segments, name and variables of the characteristic. For diode 1, this will be:

```
diodeline=PWLLine[breakpoints];
diodecharacteristic=PWLCharacteristic[D1,{i1,v1},diodeline]
```

Then these data structures will be manipulated to generate a list of rules concerning slope and threshold in each region for every diode (still step 2). In the example below, *mD1* and *qD1* indicate slope and threshold for diode 1 in a couple of regions:

```
{..{mD1 -> 0.125333, qD1 -> 0.835667, 0.25 <= i1},{mD1 -> 1.89744,
qD1 -> 0.689103, 0.001 <= i1 < 0.0205}..}
```

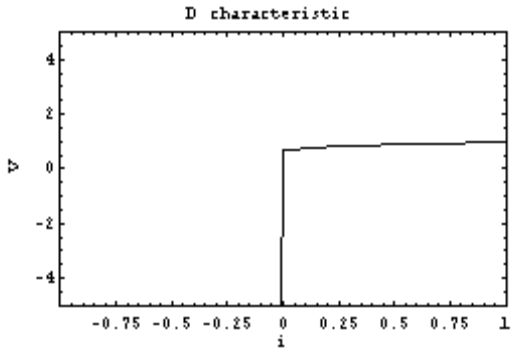


Fig. 51: Mathematica plot of the diode characteristic.

At step 3 *D1, D2, D3* and *D4* would be linear components in a circuit under DC current and a linear system can be described. We emphasize again that the system could be obtained through whatever linear method and that it will be solved as linear, because the non-linearity is embedded in the rules structure and thus transparent to the linear solver. Referring to our circuit the system will be:

```
LinearSys={    v==-i2 D2+i4 D4,
               v== i1 D1 -i3 D3,
               v== i1 D1+(i1+i3) R+i4 D4,
               i1+i3==i2+i4}
```

It stems that, if a library of non-linear components is available, and it is formalized in agreement with the conventions shown above, it is possible to analyse any circuit using that components, by simply writing linear equations, because the non-linearity is automatically handled by the Mathematica program.
Then, the program itself expands each *D* symbol in two correlated symbols: *mD* (slope) and *qD* (threshold); then it solves this system with the normal built-in Mathematica *Solve* function to obtain a general and generic solution containing the same parameters *mD* and *qD* that are subject to the previous rules:

```
{i1 -> -((2 + mD2 + qD1 + mD2*qD1 + qD2 + mD2*vi)/
      (mD1 + mD2 + mD1*mD2)),
  i2 -> -((2 + mD1 + qD1 + qD2 + mD1*qD2 - mD1*vi)/
      (mD1 + mD2 + mD1*mD2)),
  i -> -((mD1 - mD2 - mD2*qD1 + mD1*qD2 - mD1*vi - mD2*vi)/
      (mD1 + mD2 + mD1*mD2)),
  vu -> -((-mD1 + mD2 + mD2*qD1 - mD1*qD2 - mD1*mD2*vi)/
      (mD1 + mD2 + mD1*mD2))}
```

Through the symbolic manipulation of the inequalities that describe the regions for each diode, we obtain the admissible regions for the circuit as reported at step 6,7,8 and as here (partially) shown:

```
admregions={
            -19                  -19
  -2.13151 10   < v < 2.13151 10   ,
                          -19
  -1.38134 < v < 1.60437 10   ,
             -16                    -17
  -1.52564 10    <= v < -5.52705 10   ,
. . .
  v >= 1.98133}
```

These admissible regions become part of that data structure which is used to indicate rules for variables.
Applying these rules to the generic solution of the linear system we will obtain a single semi-symbolic solution particularized for our circuit. That is, a single expression that "knows when to switch" to a particular value depending on the region of the solution space (step 9,10). For example the current *i1* is partially reported:

```
i1->
         -6
-4.60 10 (290598. -174080. v) If[v >= 1.98,1.,0]+
         -6
-2.59 10 (-298.59 -669.99 v) If[-1.98 < v <= -1.47,1.,0] -
     . . .
```

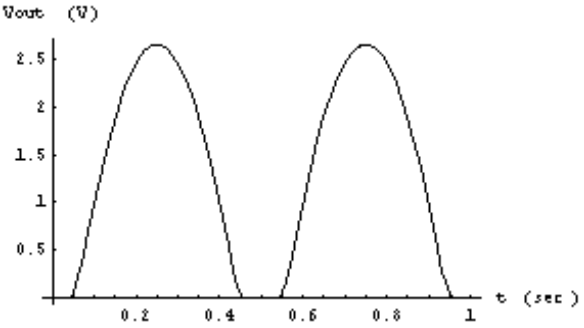


Fig. 52: Mathematica plot of the output voltage.

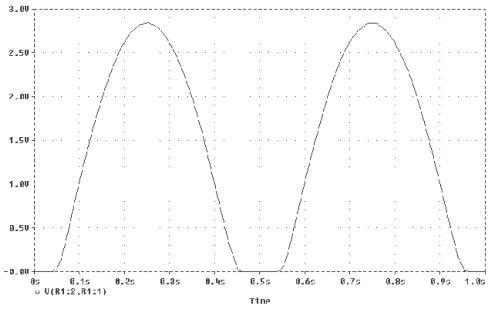


Fig. 53: PSPICE plot of the output voltage.

This solution can be directly handled by Mathematica to plot the rectified voltage on the resistor (Fig. 52). The same plot obtained by a PSPICE simulation is shown in Fig. 53 for comparison.

2.5 Conclusions for Part Two

In Part Two we showed that the analysis of resistive PWL circuits is easy to implement and relatively fast to be executed, allowing a linear approach to piecewise linear circuits, because the non-linearity is hidden to the solver and, as a matter of fact, the equations are linear, avoiding convergence problem so common in many simulators. The non-linearity workload is transferred to the process of screening between admissible and non-admissible regions and to the particularization of a generic solution.
Particularization of a solution is still not an heavy duty because, at symbolical level, Mathematica rules applications are similar to strings substitution.
The final solution (but the particularized solutions too) are still in a symbolic form, of course, which is ready for further normal manipulations under Mathematica session like plotting, storing, and calculating.
We proposed, not a complete environment for PWL circuit simulation, but a possible approach to this kind of problem. The procedure can be embedded in a more complex Mathematica program, provided with a reliable user interface to build a new tool to handle PWL circuits or it can be simply used as intermediate procedure between the user and an existing tool.

2.6 References to Part Two

[1] S. Pastore, A .Premoli, CAPTURING ALL BRANCHES OF ANY ONE-PORT CHARACTERISTIC IN PIECEWISE-LINEAR RESISTIVE CIRCUITS, *IEEE Trans. On Circuits And System*, vol.43, n.1, Jan 1996.

[2] L.O.Chua, L.A.Desoer, E.S.Kuh, LINEAR AND NON LINEAR CIRCUITS, *McGraw-Hill*.

[3] A. Liberatore, S .Manetti, SAPEC A PERSONAL COMPUTER PROGRAM FOR THE SYMBOLIC ANALYSIS OF ELECTRICAL CIRCUITS, *IEEE I nt. Symp. on Circuits and Systems*, ISCAS'88, 1988.

[4] A .Liberatore, S.Manetti, NETWORK SENSITIVITY ANALYSIS VIA SYMBOLIC FORMULATION, *IEEE ISCAS*, Portland Jun 1989.

[5] F. Filippetti, M. Martelli, SYMBOLIC TECHNIQUES ADDRESSED TO THE FAULT DIAGNOSIS OF LARGE LINEAR ELECTRIC CIRCUITS, IEEE Trans. On Circuits And System, 1996.

[6] F. Filippetti, M. Martelli, A METHODOLOGY FOR THE SYMBOLIC ANALYSIS OF LINEAR ELECTRIC CIRCUITS, Journadas Hispano-Lusas de Ingegneria Electrica, Vigo Jul 1990.

[7] M. Artioli, F. Filippetti, M. Martelli, IMPROVEMENTS TO THE FAULT DIAGNOSIS OF LARGE LINEAR ELECTRIC CIRCUITS VIA IME METHOD, *Int. Symp. on Theoretical Electrical Engineering*, ISTET'97, 1997, pp. 156-159.

[8] M. Artioli, F. Filippetti, LINEAR ANALOG CIRCUIT DIAGNOSIS BASED ON SYMBOLIC ANALYSIS AND REDUCED OBSERVABLE POINT SET, *Int. Symp. on Theoretical Electrical Engineering*, ISTET'99, 1999, pp. 485-489.

Chapter V

Virtual instruments

1 Introduction

This chapter represents a kind of short briefing about some activities of the Author in a field of interest which is not directly based on the Symbolic Calculus, but which could be merged great with it.
The intention was to investigate a feasible solution to integrate analysis and diagnosis symbolic algorithms in a diagnostic tool. A pre-existent version of a LabVIEW-based diagnostic system (at the Department of Electrical Engineering, University of Bologna) seemed to be a good starting point, due to its hierarchical open structure and due to the advanced interface functions of the LabVIEW environment.
Here are presented new solutions implemented in this diagnostic system, realized as global instrument for on-line electromechanical system integrity assessment.

2 Why a global intrument

Electromechanical system condition monitoring meets increasing attention, since the costs for unexpected outages are growing with the rate of applied technology. It seems that a definite need exists for all motor failure prediction devices, matched to detect and predict different kinds of onsetting faults. These devices should meet the characteristics of a broad variety of machines and operate with nameplate data and non-invasive sensors [1].
Spectrum analysis of input currents can show different kind of electrical and mechanical failure [2,3,4]. Spectra are computed by ordinary Fast Fourier Transform algorithms, and failure spectrum lines are identified by means of C-programs implemented as suitable virtual LabVIEW instruments. The resulting

data can be stored in order to outline time-trends of system status and, if the existence of anomalous conditions is suspected, processed by diagnostic algorithms, both conventional and employing AI techniques.
These methodologies have a general validity, but in the following sections we refer to induction motors, because specific techniques for this kind of power devices were already developed.
An “optimum” maintenance scheduling leads obviously to additional cost of the instrumentation required, along with the additional operator time. In order to decrease the cost/benefit ratio, one strategy is to pursue a reduction of training and monitoring times of the diagnostic tool operator. The diagnostic equipment should, therefore, be designed to ease the interaction between man and instrumentation. The use of graphical interfaces between software and hardware allows a more intuitive approach to diagnostic operation.
The main features of the system are: multi-diagnosis facility on different systems simultaneously as well, automatic or man-driven diagnostic process and trend analysis. All the procedure is based upon the identification of anomalous spectrum lines in the input currents according to the CSA method (Current Signature Analysis). The results of FFT computation are then stored in a database and can be easily retrieved for visualization. Some instrument front panels are reported, in order to show the features of the diagnostic system.

3 Basic scheme for a multi-diagnosis instrument

The basic virtual instrument designed for machine diagnostics has the architecture schematised in Fig. 54.

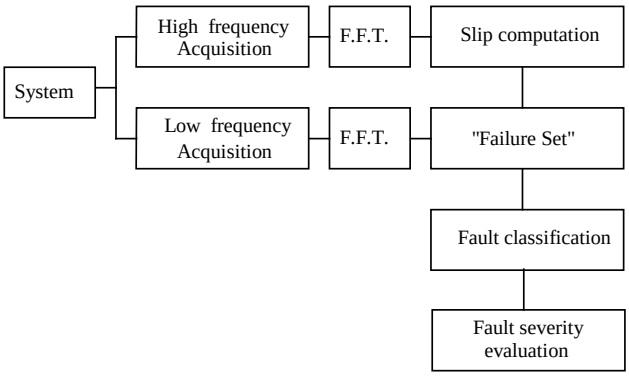


Fig. 54: Sub-instrument global scheme.

Two acquisitions of the instantaneous currents over different time intervals and with different sampling rates are performed. The former is devoted to slip

computation, while the latter provides the data that are stored for further diagnostic functions.

3.1 A. Slip computation

Two methods are selectable to perform slip computation, starting from DFT analysis of the stator current signal: one based on slot spectral components [1] and one based on rotor electrical dissymmetry components [5]. There are, of course, two different virtual instruments devoted to the computation procedure. Their front panel (Fig. 55 and Fig. 56) are referred to a 1700kW three-phase machine with 3 pole pairs and 74 rotor slots. Both techniques give almost the same slip value: 0.0059.

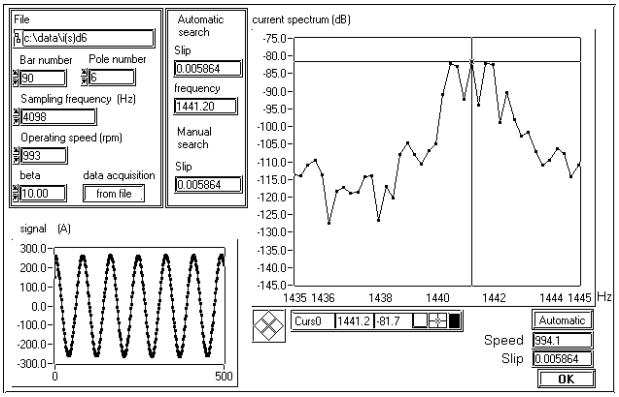


Fig. 55: Slip measurement by the slot harmonics technique (front panel).

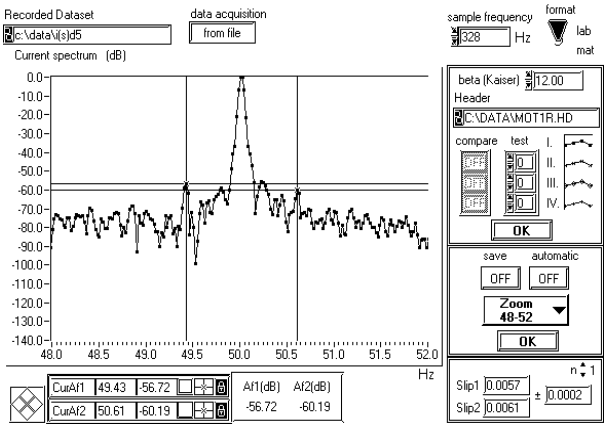


Fig. 56: Slip measurement by the fault frequencies technique (front panel).

3.2 B. Failure set

The failure set formulation can be done in different ways, all reflecting the Minimum Configuration Artificial Intelligence (MCAI) [6].
The task is to filter spectral components, to eliminate those components that provide no useful failure information and to perform a components selection, searching the spectrum components which characterize the fault under test. The function is performed by a "C" program implemented in a virtual instrument and it ends building a vector of considered components, named “Failure Set” (FS) [7,8,9] for the motor currently under test.
This diagnostic system is FC-FSM compliant (Fault Classification – Fault Specification Method) [6,10]. The diagnostic session, in fact, has two steps: first, classification of the fault and second, evaluation of the fault severity.

3.3 C. Fault classification

This identification process is done by an expert system inferential engine, implemented as a "C"-program in a suitable virtual instrument. The expert system, having at disposal the FS, databases containing machine operating conditions and machine history in term of trend of well stated failure components, is able to classify a fault occurrence [4,11].
Selectable procedures are: threshold analysis or neural network (NN) technique (if NN previously trained). These techniques are embedded in C-program under virtual instruments.
According to the selected procedure the expert engine activates, for example, an unsupervised NN based classification method.
If a FS containing current spectrum lines related to the machine situation is presented, the unsupervised NN clusters the FS into the characteristic region of the output node lattice, realizing a fault topographic map that can be also utilized immediately by visual inspection. As an example, Fig. 57 reports the fault map trained by rotor dissymetries (1), stator dissymetries (2) and bearing damages (3) together with healthy machine cases (H) [4,6,11] for a three-phase 0.45 kW test motor.

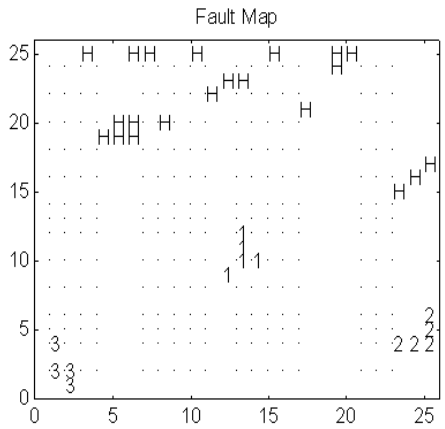


Fig. 57: Fault map example to distinguish stator, rotor and bearing damages.

If threshold analysis is selected, the expert engine starts this method to verify whether the diagnostic indicator for that FS is over the threshold. The attention threshold can be derived by analysing the diagnostic indicator trend. Fig. 58 reports a typical example related to a 900kW machine under test for a long time. The trend of the diagnostic indicator (that is, the sum of the two fault components at the frequencies *(1±2s)f* in percentage with the fault component at the supply frequency [6,10]). It shows that a bar get effectively broken at the 50th monitoring test of the figure. It is also easy to state that in this case can be assumed an attention threshold of 0.5%.
If the FS is ambiguous, these methods can not decide on the fault classification.

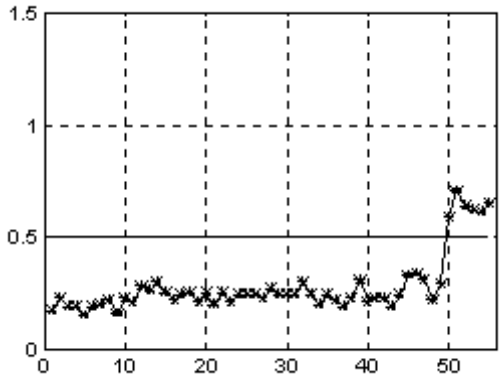


Fig. 58: Trend of the input current sideband sum of a 900kW 6kV 8 pole induction machine.

3.4 D. Fault severity evaluation

The last step is evaluating the fault severity. This evaluation is done starting from motor operating conditions, motor characteristics, trend, and threshold value for faults, by using conventional methods or AI techniques [10,14,15,16,17,18,19]. The selection of different specific methods is programmable in the diagnostic session editor.

Each diagnostic session can involve three type of faults (on stators, rotors or bearings). Depending on the result of the classification step, the system assumes that a particular failure is probably occurred and invokes the specific diagnostic method for that kind of fault (or, if none occurred, states the correct operating condition to the operator). If the FS is ambiguous, instead, it asks for an operator action, which can decide on the fault severity by visual inspection of the data shown by various available front panels related to the justification features.

This kernel is used by several virtual instruments integrated on a programmable diagnostic system, that is, the operator is allowed to define a diagnostic session through some configuration panels. As an example we show in Fig. 59 a panel related to a rotor faults diagnostic session. The machine under test is the usual three-phase 0.45 kW test motor with one pole pair in faulty condition (two broken bars). The front panel has (software) led indicator to highlight the diagnosis result and a label for the selected specific method. In the case reported below “fuzzy” is the indication of the used method. By clicking the “Details” key, an other front panel is activated, which explains more detailed information about the diagnosis result: each one of the possible situations, that are NO FAULT, INCIPIENT FAULT, ONE BROKEN BAR, TWO BROKEN BARS, TWO OR MORE BROKEN BARS, is characterized by a degree of membership, which is evidenced by bar indicators [1].

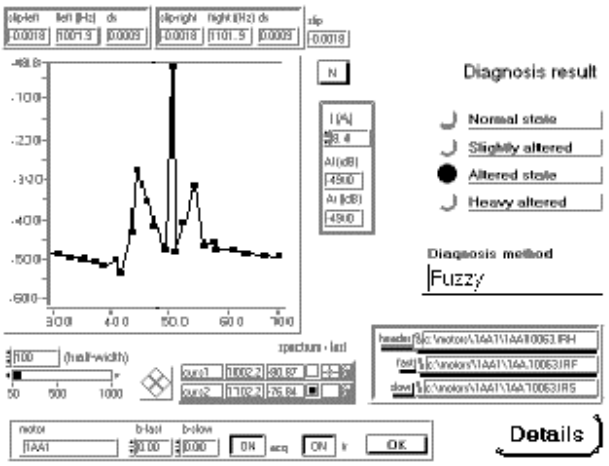


Fig. 59: Rotor fault diagnosis result front panel.

4 Multi-device monitoring features

Limited to the acquiring board capabilities, it is also possible to use up to eight channels for data acquisition, specifying acquiring process and test device for each of them.
The independence of the data channels allows to program the diagnostic session in two basic way:

- the same diagnostic on up to eight different motors (if the diagnostic needs only a data stream, of course);
- different diagnostics on a single motor (in the heaviest case six data streams are needed: three currents and three voltages), that is, the system reads all needed data and through the previously outlined process, it individuates the fault and activates the appropriate diagnostic.

It is also possible:

- to perform a manual diagnostic session;
- or to automate it to minimize operator action: it is possible to repeat the programmed set of diagnostics (on several and different devices as well) a fixed number of times during a definite period, without man operations;
- to record the diagnostic session data on a database (that can be visualised at any time) to add new information for further trend analysis.

Several front panel get the operator able to define the various features of the diagnostic system. As an example we report in Fig. 60 and Fig. 61 the front panels for the channel function definition and for motor parameter definition.

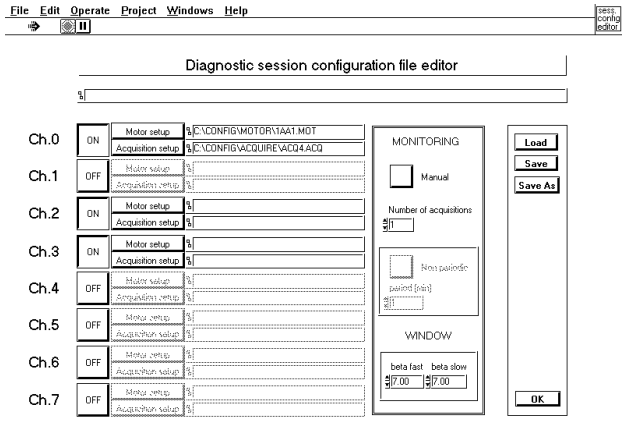


Fig. 60: Configuration panel for diagnostic session.

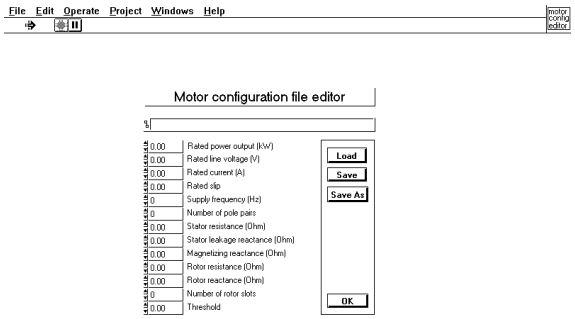


Fig. 61: Configuration panel for motor description.

5 Conclusions

In this chapter, a diagnostic tool for induction machine monitoring, realised by means of the software package LabVIEW, has been presented. The acquisition hardware is controlled by virtual instruments, VIs, that is, images displayed on a personal computer monitor, closely resembling real-world instruments. The front panels of the VIs are designed in order to provide trained operators with an immediate insight of machine conditions, while threshold-controlled alarm system are employed to alert un-expertised personnel.
The tendency of the machine toward fault conditions can be pointed out by means of an other sub-panel of the VI, designed to display data recorded in a data-base. Operators are, therefore, enabled to track trends and plan maintenance operations to be performed ahead of machine failure.
The proposed diagnostic system is programmable and devoted to the monitoring of different devices and it is realized as hierarchical and modular structure. For these reasons it seems very suitable to be completed with diagnostic modules based on Symbolic Calculus [Chapter III] to enhance its flexibility.
Besides, graphic programming tools can greatly ease man/machine interaction in induction machine condition monitoring, allowing un-expertised personnel to perform routine checks of machine status. Hence, the cost/benefit ratio of the diagnostic procedure can be, on the whole, significantly reduced.

6 References

[1] A. Cavallini, F. Filippetti, G. Franceschini, S. Pirani, C. Tassoni. A GLOBAL INSTRUMENT FOR ON-LINE INDUCTION MOTOR INTEGRITY ASSESSMENT. *Stockholm Power Tech*, Stockholm Sweden June 1995.

[2] J.R. Cameron, W.T. Thomson, A.B. Dow: ON-LINE CURRENT MONITORING OF INDUCTION MOTORS, *IEE-EMD Conf.*, N.282, 1987.

[3] J.B. Kliman, J. Stain: METHODS OF MOTOR CURRENT SIGNATURE ANALYSIS. *Electric Machines and Power Systems*, N.20, 1992.

[4] R.R. Schoen, T.G. Habetler, B.K. Lin, J.H. Schlag, S. Farag: AN UNSUPERVISED ON-LINE SYSTEM FOR INDUCTION MOTOR FAULT DETECTION USING STATOR CURRENT MONITORING. *IEEE-IAS'94*, October 94, Denver.

[5] L. Collamati, F. Filippetti, G. Franceschini, C. Tassoni. A TECHNIQUE FOR INDUCTION MACHINE SPEED DETECTION BASED ON ROTOR ELECTRICAL ASYMMETRY FREQUENCIES IDENTIFICATION. *IEEE Conf.on Harmonics and quality of power ICHQP'96*, Las Vegas October 1996.

[6] F. Filippetti, G. Franceschini, C. Tassoni, P. Vas: INTEGRATED CONDITION MONITORING AND DIAGNOSIS OF ELECTRICAL MACHINES USING MINIMUM CONFIGURATION. *Int. Eur. Conf. EPE'97*, Trondhaim, Norway, September 1997.

[7] F. Filippetti, G. Franceschini, M. Martelli, C. Tassoni: AN APPROACH TO KNOWLEDGE REPRESENTATION ABOUT INDUCTION MACHINE DIAGNOSTICS IN EXPERT SYSTEMS. *Proc. of ICEM'88 Conference*, September 1988, Pisa, Italy.

[8] D. Leith, N.D. Deans, L.I.D. Stewart: CONDITION MONITORING OF ELECTRICAL MACHINE USING REAL TIME EXPERT SYSTEMS. *Proc. of ICEM'88 Conference*, September 1988 Pisa Italy.

[9] P. Calonnec, T. Derrey, E. Destobbeler, L. Protin: INDUCTION MOTOR RELIABILITY USE OF FAULT TREE. *EPE'95*, September 1995, Sevilla, Spain.

[10] F. Filippetti, G. Franceschini, C. Tassoni, P. Vas: A.I. TECHNIQUES IN INDUCTION MACHINES DIAGNOSIS CONSIDERING SPEED RIPPLE EFFECT. *IEEE- Trans. on I.A.*, Jan. 1998

[11] J. Penman, C.M. Yin: FEASIBILITY OF USING UNSUPERVISED LEARNING NN FOR THE CONDITION MONITORING OF ELECTRICAL MACHINES. *IEE Proc. Electr. Power Appl*., Vol. 141 (6) 1994.

[12] J.H. Schlang, T.G. Habetler, B.K. Lin. AN UNSUPERVISED NEURAL NETWORK FAULT DISCRIMINATING SYSTEM IMPLEMENTATION FOR ON LINE CONDITION MONITORING OF INDUCTION MACHINE USING STATOR CURRENT. *Int. Symp. SDEMPED'97*, Carry le Rouet France September 1997.

[13] F. Filippetti, G. Franceschini, C. Tassoni, P. Vas: TRANSIENT MODELLING ORIENTED TO DIAGNOSTICS OF INDUCTION MACHINE WITH ROTOR ASYMMETRY. *Proc. of ICEM'94*, Paris, Sept. 1994.

[14] G. Gentile, N. Rotondale, F. Filippetti, G. Franceschini, M.Martelli, C. Tassoni: ANALYSIS APPROACH OF INDUCTION MOTOR STATOR FAULTS TO ON-LINE DIAGNOSTICS. *ICEM'90*, Cambridge, Massachusetts, Ag.90.

[15] R. R. Schoen, T.G. Habetler, F. Kamran, R. G. Bartheld: MOTOR BEARING DAMAGE DETECTION USING STATOR CURRENT MONITORING. *IEEE-IAS'94*, October 94, Denver, Colorado.

[16] E. Ritchie, Xiaolan Deng, T. Jokinen: DIAGNOSIS OF ROTOR FAULTS IN SQUIRREL CAGE INDUCTION MOTORS USING A FUZZY LOGIC APPROACH. *Proc. of Int. Conf. ICEM'94*, September 1994, Paris

[17] F. Filippetti, G. Franceschini, C. Tassoni, P. Vas: A FUZZY LOGIC APPROACH TO ON-LINE INDUCTION MOTOR DIAGNOSTICS BASED ON STATOR CURRENT MONITORING. *Stockholm Power Tech*, June 1995, Stockholm, Sweden.

[18] K. R. Cho, J. H. Lang, S. D. Umans: DETECTION OF BROKEN ROTOR BARS IN INDUCTION MOTORS USING STATE AND PARAMETER ESTIMATION. *IEEE-IAS'89*, San Diego, California, Oct. 1989.

[19] F. Filippetti, G. Franceschini, C. Tassoni, NEURAL NETWORKS AIDED ON-LINE DIAGNOSTICS OF INDUCTION MOTOR ROTOR FAULTS. *IEEE Transactions on Industry Applications Society*, IEEE Publisher, July/August 1995

Index