U.P.B. Sci. Bull., Series C, Vol. 69, No. 3, 2007 ISSN 1454-234x

A BASH SCRIPT FOR CONVERTING SPICE LIKE, SCHEMA DESCRIPTION − TEXT FILES, INTO MODIFIED NODAL ECUATION MATRIX WITH SYMBOLIC ELEMENTS

C. ZORIO[1], M. BODEA[2], I. RUSU[3]

Această lucrare prezintă o implementare a algoritmului de generare a matricei sistemului modificat de ecuații al potențialelor la noduri, aceasta fiind generată ca matrice cu elemente de forma unor formule simbolice. Datele de ieşire ale acestui program pot fi stocate intr-un fişier de tip text care poate fi importat într-un program CAD cu ajutorul căruia se pot face calcule simbolice complexe pentru analiza de semnal mic a unui circuit, inclusiv extracție de parametrii. Avantajul unei metode simbolice pentru extracția de parametrii de semnal mic, este faptul că utilizează un algoritm de calcul direct si nu mai este necesara determinarea unor valori inițiale de "start" pentru parametrii de extras. Se arată că, în scopul generării formulelor simbolice ca elemente ale matricei nodale, se impune modificarea sintaxei formatului de intrare de tip SPICE si este propusa o sintaxă extinsă reprezentând un format modificat de fişier text tip SPICE. Programul face posibilă generarea matricei nodale pentru un circuit oarecare, oricât de complex, funcționalitatea sa fiind ilustrată utilizând ca exemplu circuitul "Giacoletto". Sunt date sursele in "bash", direct utilizabile in orice mediu "UNIX".

This paper presents an implementation of the modified nodal algorithm which can generate the description of the modified nodal matrix with symbolic formulae as elements. The output data can be stored in a text file which can be imported in a mathematical CAD environment for further mathematical symbolic complex calculations for small signal analysis of a circuit, including parameter extraction. The advantage of a symbolic method for small signal parameter extraction is that it uses a direct algorithm and initial "start values" for the parameters to be extracted are not needed anymore. In order to generate symbolic formulae as the nodal matrix elements, the syntax of the SPICE input format has to be modified. An extended syntax was proposed resulting in a modified SPICE-like text file format. The program can generate the nodal matrix of any circuit, no matter its complexity, its functionality being illustrated using as example the "Giacoletto" circuit. "Bash" sources, directly usable in any "UNIX" environment, are provided.

Keywords – symbolic analysis, modified nodal analysis, small signal analysis, parameter extraction, SPICE input format.

[1] Eng., Dept. of IT & Computers, Romanian National Television Society, Romania, cristian.zorio@gmail.com
[2] Prof., Dept. of Devices, Circuits and Electronic Apparatus, University POLITEHNICA of Bucharest, Romania
[3] Prof., Dept. of Electronic Technology and Reliability, University POLITEHNICA of Bucharest, Romania

1. Introduction

The state of the art in symbolic computer aided mathematical calculus, opens the opportunity of taking advantage of the important benefit of automatically generating and manipulating large symbolic formulae [1], [2], which can now be used for the analysis of real (bigger) circuits. In the case of parameter extraction for the small signal circuit's schema, the use of symbolic computation can be much more effective than old numerical methods. This is the consequence of the fact that direct symbolic extraction algorithms eliminates the problem of finding an initial starting point for an iteration scheme [6]. The solution for this "initial" problem depends entirely on the circuit designer's experience and intuition based on some simplified circuit model.

The purpose of this paper is to evidence the new problems that can arise in case of using symbolic formulae manipulation with mathematical CAD programs, for parameter extraction and for the small signal analysis of a given circuit.

The paper presents the implementation of a program which makes a link between two different representations of a circuit model, using the algorithm which generates the modified nodal system of equations, to convert the circuit description into a description of the modified nodal system: a matrix description having numbers and/or symbolic formulae as elements. The output data can then be stored in a text file which could be imported in a mathematical CAD environment [2], for further both mathematical symbolic and/or numeric computations.

2. Input and Output formats

In order to achieve the goal of generating the system of equations of a circuit, for use in a mathematical CAD environment, the relevant information contained in the schema of a circuit (meaning circuit description: topology and values of the elements) has to be "expressed" and stored (as data structure) as the content of a file which could be then imported by the mathematical CAD program [2], or by some other formulae interpreter software [1].

For this purpose, an input format has to be defined, and a SPICE–like circuit description is proposed in section *§1.2.2*. This description will be the content of an input data text file, which is then used as input by the transformation program described in section *§3*.

The output format of the data transformation is imposed by the chosen program which will manipulate the symbolic formulae (in our case Maple [2]) in order to perform the small signal analysis. This format is defined in section *§1.2.1* and section *§4* also provides the output file for the example presented in *§1.2.2*.

2.1 Output Format: format of the Modified Nodal Matrix text file

A text file, importable in Maple, which can generate a matrix with elements which can be numbers, symbols or symbolic formulae, has the format described in Fig. 1 [2]:

```
1        1        <character string>
1        2        <character string>
                 ……..
1      <n+1>      <character string>
2        1        <character string>
2        2        <character string>
                 ……..
2      <n+1>      <character string>
                 ……..
n        1        <character string>
n        2        <character string>
                 ……..
n      <n+1>      <character string>
```

Fig. 1 - Output format of the modified nodal matrix description

The meaning of <character string> is given next in (1) :

<chatacter string> := <symbolic formula> | <symbol> | <number> (1)

The nodal matrix with the format presented in Fig. 1, has n lines and n+1 columns. These dimensions correspond to a circuit having n+1 nodes (numbered form node *0* - the current notation for the ground node – to node n). The $(n+1)^{th}$ column of the nodal matrix corresponds to the free term constants of the equation system.

A typical output file content is given in Fig. 4, in the “Results section" *§3.*

2.2 Input Data Format: SPICE-like circuit description and modified SPICE syntax

It is well known that, for most CAD circuit simulation environments, before any computations are performed, the graphical interfaces of the CAD system converts the graphic schema representation of the analyzed circuit into a text file containing the same schema description written in SPICE format [3]. In the SPICE text description model, the input information coming from the schema of a circuit does not represent a mathematical formulation, although, like mathematical formulation models, this type of data also represent an exhaustive description of the topology of the circuit and also of all the numeric values for the elements of the schema. The content of the SPICE file always represents the "starting point" for the basic "core" calculations of the engine of any CAD program used for circuit analysis or design.

The use of the main rules of the SPICE input format also for symbolic calculation must come as a natural consequence of the fact that this kind of representation has imposed itself as an industry standard. Hence, for allowing the use of symbols inside the conversion algorithm, the common SPICE syntax also has to be modified and an extension of the SPICE input syntax rules is needed.

For this purpose it will be formally assumed that if there are unstated (unknown) numeric values for some elements in a particular circuit schema, the very symbols of these elements must be used inside the algorithm, instead of the (missing) numerical values. As a consequence, the new SPICE-like format syntax must allow lines which do not contain the numerical values of the elements. For these lines only, the reading program must use the text strings representing the names of the schema elements (considered as, symbols representing the simplest symbolic formulae), as the numeric values of these schema elements (needed by the old syntax rules) are not available.

The features of the SPICE-like format are presented using the following example.

▶ ***Example:***

Suppose that a symbolic analysis of the simple circuit of Fig. 2 has to be performed using symbolic formulae. A SPICE-like description of the circuit of Fig. 2 is given in Fig. 3.

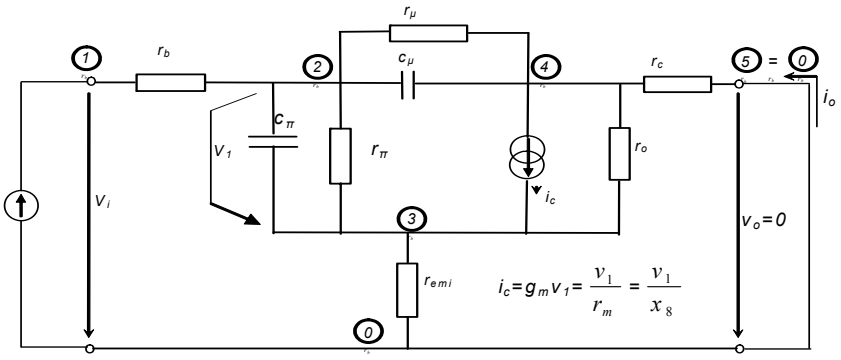

Fig. 2 - Circuit Example – The Giacoletto equivalent circuit of a bipolar transistor

Using the example of Fig. 3 the following, most important, general syntax features of the SPICE-like format can be evidenced:

- comment lines beginning with the "#" character are to be omitted
- all non-comment lines must begin with the name of an element of the schema. According to the usual SPICE syntax [3], the first letter of the symbolic name of the element indicates the element type.
- the symbolic name has to be followed by at least two node labels representing the numbers that indicates the nodes to which it is connected.
- if the next two node labels that may come after the first two nodes exist, they determine the nodes which define a signal that determines the value of the signal corresponding to the first two nodes. (this is the case of the g_m element which is connected between nodes 4 and 3 and depends of the difference of the electric potentials of nodes 2 and 3).
- in the usual SPICE syntax, every non-commented line normally ends with the value of the symbolic element the line was written for. In the (modified) SPICE-like format, the value of the element can be omitted, the meaning for the "reading program" being as follows: the name of the circuit element (the first element in the current line), as "symbolic value", has to be taken into account instead of the numeric value.

In the example shown in Fig. 3, the values of all the elements are omitted (starting with line two, all non-commented lines do not end with a corresponding numerical value), and the result will be a matrix with all its elements being symbolic formulae, each formula using circuit element symbols inside.

```
# <Element>      Nodes          Values
  Rb           1   2
  #Rb          1   2                  300
  Cpi          2   3
  #Cpi         2   3                      0.00000000002
  Rpi          2   3
  #Rpi         2   3                  5000
  Ru           2   4
  #Ru          2   4              200000000
  Cu           2   4
  # Cu         2   4                      0.0000000000002
  gm           4   3   2   3
  #gm          4   3   2   3              0.038
  Ro           4   3
  #Ro          4   3              250000
  Rc           4   0
  #Rc          4   0                  50
  Remi         3   0
  #Remi        3   0                      2
  I1           0   1
  #I1          0   1                      0.1
```

Fig. 3 - SPICE-like input file content for the circuit in Fig. 2

For the purpose of illustrating the possibility of also using numerical values (in order to show that the new SPICE-like syntax maintains the rules of the old SPICE syntax too) each non-comment line representing an element is followed by a possible replacing line which is commented. These are the same lines as the above uncommented ones with the difference that a numerical value was also specified. For this example, all lines containing numerical values for schema elements are commented but this not necessarily the general case.

The conclusion is that changing the type (numeric with symbolic) of the algorithms which use information contained in the schema of a circuit in such a way that, in the conversion process, (at least) a part of the input data, originally numbers, is replaced with symbols, has raised the following question: do the data structure for numerical computations (the classic SPICE format) remain appropriate for describing the schema of some circuit when performing symbolic computations? Even if the answer is that the original data format had to be changed, the data structure can be "fixed" in a natural way by extending the syntax rules of the classic SPICE format for the reading program to also accept symbols for the values of the elements of the circuit (and not only numbers).

A closer comparison between the classic SPICE format and the extended one reveals that the new syntax rules did not change in any way the "coding features" which make possible the description of the topology of the circuit. From this point of view the data structure remains the same.

3. Conversion program

Based on the modified nodal method [4], a conversion program was implemented. This program generates the matrix of the linear system having as variables the potentials in the circuit nodes and the currents in the voltages sources.

3.1. Algorithm implementation

The algorithm was implemented in bash [5]. The resulting scripts are: a main script, the content of the file named "mkmatrix" and a subroutine script, the content of the file named "elemt_to_y" [7].

The flowchart for the main program is given in section "Appendix 1", in Fig. 5 and Fig. 6. The routine "elemt_to_y", presented in Fig. 7, is invoked for each schema element for which admittance can be defined.

3.2. Description of the conversion program

In order to provide a better understanding of the flowchart of the main program (Fig. 5 and Fig. 6), it was formally divided into more sections delimited by comment lines of the form: #SECTION <number> begins , # SECTION <number>ends . The functionality of each section is as follows:

- SECTION 01 - reads the input (SPICE-like) file line by line, determines the number of (non-commented) lines, **nr_lines** and declares a vector of **nr_lines** elements, **memline**[**nr_lines**], which will store the lines of the input file.
- SECTION 02 - stores each (non-commented) line of the input file in each component of previously declared **memline** vector. This section uses the same control loop to determine variable **labelmax**. This variable must store the biggest of the numbers allocated to the set of node labels. If assuming that the nodes in the SPICE-like input file are numbered in increasing order starting with 0 (which corresponds to the node representing the ground) and adding 1 to get the next label value, then the number of nodes is **labelmax** + 1 [4]
- SECTION 03 - determines the number of variables/equations by increasing the node number with the number of independent voltage independent

[4] According to Kirchoff I law, the number of independent current equations to be used in the nodal method will be equal with **labelmax** = **nodenr**-1, where **nodenr** is the number of nodes. In order to obtain a determined linear system, one of the **nodenr** current equations K I, has to be eliminated: the equation corresponding to node 0 (ground).

sources. The vector **outline** that will contain the elements of the matrix of the linear system of equations generated by the modified nodal method is then defined. This matrix has **labelmax** + 1 lines (starting from line 0 to line **labelmax**) and **labelmax** + 2 columns (starting from column 0 to column **labelmax**+1).

- SECTION 04 - implements the main rules of the modified nodal algorithm, [4], as follows :
  - SECTION 04-1 : - implements the rule for the contribution to the nodal matrix of the voltage controlled current sources (VCCS) : a VCCS of trans-admittance **y** connected between nodes **node1_2** and **node2_3** and commanded by the difference of potentials of nodes **word_4** and **node4_5** alters the linear system's matrix by adding **–y** and respectively +**y** to the system's matrix elements of both lines **node1_2** and **node2_3** at columns **word_4** and respectively **node4_5**.
  - SECTION 04-2 : - implements the rules for the contribution to the nodal matrix of the admittances of the resistors, capacitors and inductors (R,L,C elements). An admittance **y** connected between nodes **node1_2** and **node2_3** alters the linear system's matrix by adding **–y** and respectively +**y** to the system's matrix elements of both lines **node1_2** and **node2_3** at columns **node1_2** and respectively **node2_3**.
  - SECTION 04-3 : - implements the rule for the contribution to the nodal matrix of the voltage independent sources E. An independent voltage source of value E connected between nodes **node1_2** and **node2_3** alters the equation system by adding a new variable (the current through E) to the system. This variable will be present in equations with index number **node1_2** and **node2_3** which means that the coefficient of the corresponding column (**varnr)** of the equation's system matrix (**outline**) will be -1 respectively +1 for lines **node1_2** and **node2_3** (and zero in the same column for the rest of the lines). A new equation which defines the difference between potentials of **node2_3** and **node1_2** to be E is also introduced. This goal is achieved by setting to -1 and respectively to +1 the elements of columns **node1_2** and **node2_3** in the corresponding line of the system's matrix (line **ecuatnr)** and also the last column in the same line to the value of E.
  - SECTION 04-4 : - implements the rule for the contribution to the nodal matrix of the current independent sources (I). An independent current source of value I connected between nodes **node1_2** and **node2_3** alters the equation system by adding to the last (free) term of equation with

index number **node1_2** a value of –I respectively and to equation with index number **node2_3** a value of +I

- SECTION 05 - prints out each element of the matrix of the modified nodal system of equations, each preceded by a pair of two numbers representing the position of this element in the matrix. Note that the equation obtained from the corresponding Kirchoff I law, corresponding to the ground node (labeled 0), was eliminated by starting iteration loops from I = 1 and J = 1 (and not from I=0 and J=0). Also note that the rule for node labeling in the input file, imposed that the vector **outline** had to be defined in C-like style meaning that it's elements are numbered starting with index 0 (meaning **outline**[0,0] is the first element).

The routine of Fig. 7 could be formally divided in sections corresponding each to the right branches of the decision statements included.

Each affirmative decision branch leads the execution of the program to the appropriate formulae generators in order to compute the admittance of a schema element. The type of each element is identified, according to SPICE syntax [3], using the first letter of the corresponding element symbol.

Each admittance is calculated for the current schema element, either as numerical value or as symbolic formula. For this choice the routine checks in the current line of the input file, for the existence (or absence) for a numerical value of the current schema element.

4. Results

In order to use the program and the subroutine presented in section "Appendix 1"and explained in section §3, the two script files named **mkmatrix** and respectively **elemt_to_y** have to be used as commands, in any UNIX environment. The main file must receive the execute attribute:

```
#chmod +x  mkmatrix                                    (2)
```

The main script has the file containing the SPICE-like description as argument:

```
#./mkmatrix <input file>                               (3)
```

▶ ***Example:***

Using the content of Fig. 3 as the content of the file "**giacoletto**" and running:

```
#./mkmatrix giacoletto                                 (4)
```

will print out all the elements of the modified nodal matrix as in Fig. 4.

```
#I      J       <Element>
 1      1       -1/Rb
 1      2       +1/Rb
 1      3       0
 1      4       0
 1      5       +J1
 2      1       +1/Rb
 2      2       -1/Rb-I*Omega*Cpi-1/Rpi-1/Ru-I*Omega*Cu
 2      3       +I*Omega*Cpi+1/Rpi
 2      4       +1/Ru+I*Omega*Cu
 2      5       0
 3      1       0
 3      2       +I*Omega*Cpi+1/Rpi+gm
 3      3       -I*Omega*Cpi-1/Rpi-gm-1/Ro-1/Remi
 3      4       +1/Ro
 3      5       0
 4      1       0
 4      2       +1/Ru+I*Omega*Cu-gm
 4      3       +gm+1/Ro
 4      4       -1/Ru-I*Omega*Cu-1/Ro-1/Rc
 4      5       0
```

Fig. 4 – Modified nodal matrix output file content, for the circuit of Fig. 2 (for the SPICE-like input file content described in Fig. 3)

Saving the output in a text file (here named "mat_giacoletto.txt") is done using the redirection operator ">":

```
#./mkmatrix_11 giacoletto > mat_giacoletto.txt                    (5)
```

This type of text file describing a matrix can then be easily imported in the Maple environment [2] for further symbolic manipulation.

5. Conclusions

The paper presents a program which is an implementation of the algorithm generating the matrix of the associated modified nodal linear system of equations of a circuit, with symbolic formulae and/or numbers as elements. Two aspects were discussed: first the input/output format, second effective implementation using "bash".

The output file format of the program was imposed by the mathematical CAD environment which has to import the file containing the symbolic element matrix description. The input format had to be chosen in such a way that it will make the description of the circuit as simple as possible.

Knowing that the SPICE input format, is the industry recognized standard (as data structure to be "understand" by any computer program) for describing the schema of electronic circuits, a SPICE-like syntax was proposed by extending the rules of the usual SPICE input format syntax in order to allow the use of symbolic values inside the algorithm. It has been shown that the proposed extended SPICE format which allows the use of symbols as "element values" could be easily interpreted by the transforming program.

The symbolic formulae (as the elements of the "modified nodal matrix") could be generated, basically using the same main algorithm as in numeric computation, but with symbolic features. The new input data structure (with extended syntax rules) proved to be a very good candidate for a more general circuit schematic description. The input text file can be easy written even in the case of large circuit schemas.

The sources of the program are given and the functionality was illustrated by presenting a process of generating the nodal matrix associated in some particular case, the "Giacoletto" circuit. The program can to generate this matrix for any circuit, no matter its complexity. The original ideas of using "bash" scripting as the implementing language makes possible a direct use on any "Linux"/"UNIX" machine, by copying and pasting the sources in executable text files. Thus, anyone which could imagine possible further developments of the program needs no special compiler or programming environment.

REFERENCES

[1] *D. Cox, J. Little, D. O'Shea*, Computer Algebra Systems *in*: "Ideals, Varieties and Algorithms-An Introduction to Computational Algebraic Geometry and Commutative Algebra", APPENDIX C, pp. 505,-517, *–Springer-Verlag New York Berlin–1992– ISBN 0-387-94680-2, SPIN 10675946*.

[2] MAPLE-*A Mathematical CAD Program*, http://www.maplesoft.com/ .

[3] *A. Vladimirescu*, The SPICE Book, John Wiley & Sons, Inc., New York, 1994.

[4] *Benedykt S. Rodanski and Marwan Hassoun,* Symbolic Analysis Methods*, chap. 47.2 in*: "The Circuits and Filters Handbook"*, Second Edition ISBN 0-8493-0912-3*

[5] BASH-*Reference Manual*, http://www.gnu.org/software/bash/manual/bashref.htm .

[6] *C. Zorio*, Extragerea parametrilor de semnal mic ai unui circuit liniar utilizând calculul simbolic automat, - *Referat de doctorat,* - Universitatea Politehnica Bucureşti, Facultatea de Electronica Telecomunicaţii şi Tehnologia Informaţiei - Catedra de Dispozitive Circuite şi Aparate Electronice

[7] Sources of the conversion program: *mkmatrix* and *elemnt_to_y* BASH script files - available from the authors, via E-Mail request (*)

(*) In order to get the sources of the conversion program please send a blank E-Mail with the subject MATRIX CONVERSION PROGRAM to cristian.zorio@gmail.com

Appendix 1: Flowcharts for the main conversion program “mkmatrix” and for its subroutine “elemt_to_y”

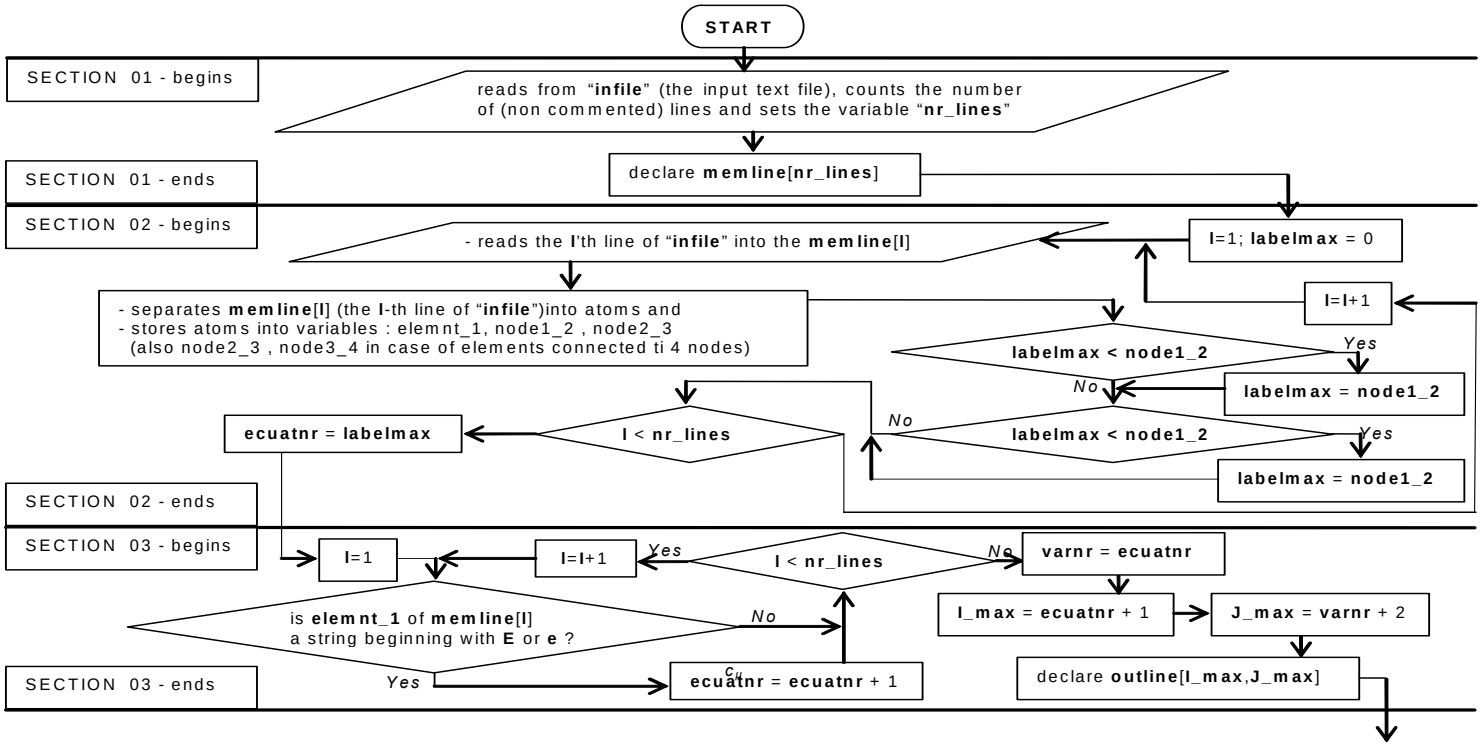

Fig. 5 - First part of “mkmatrix”: the main conversion program

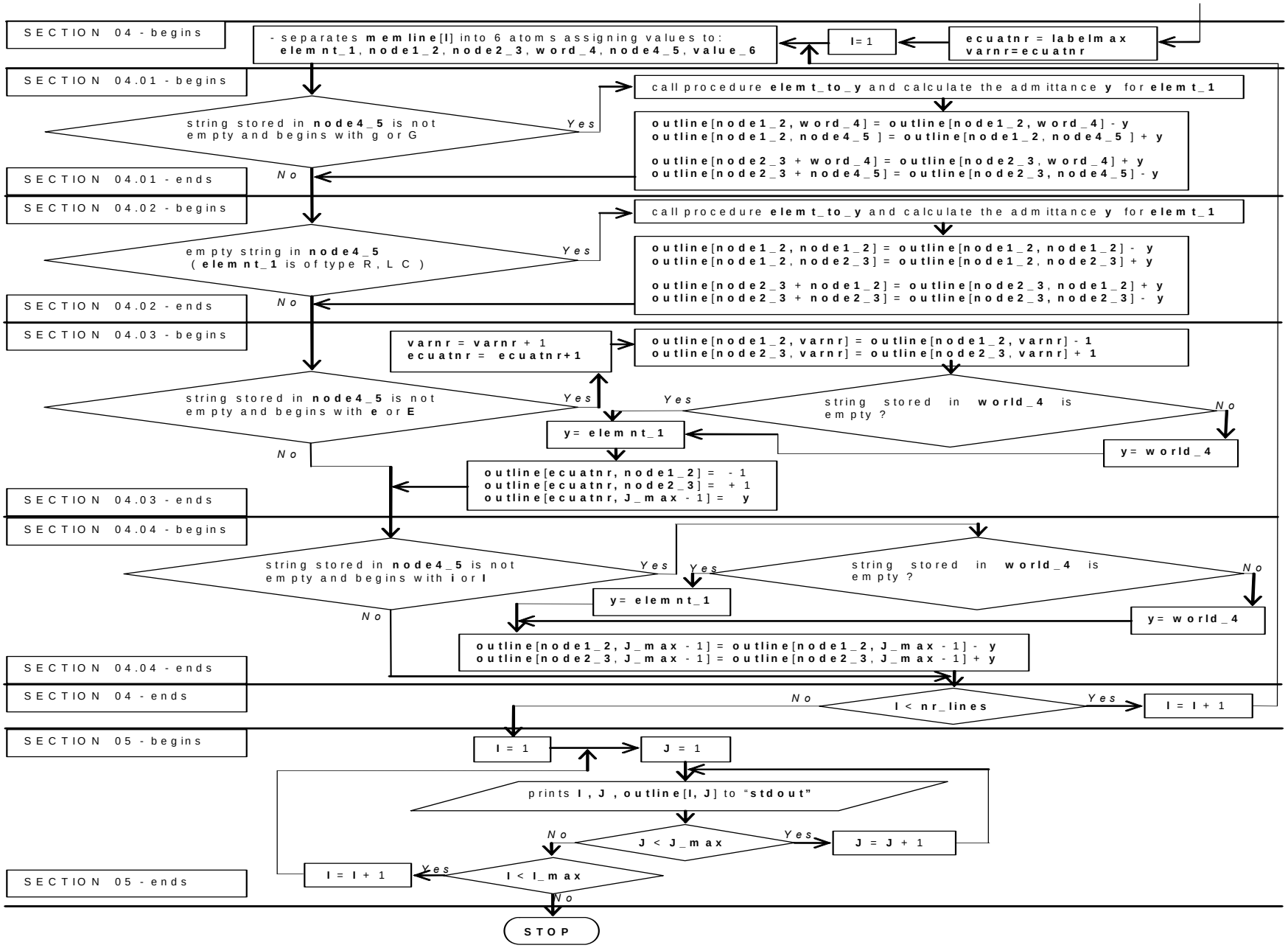

Fig. 6 - Last part of "mkmatrix": the main conversion program

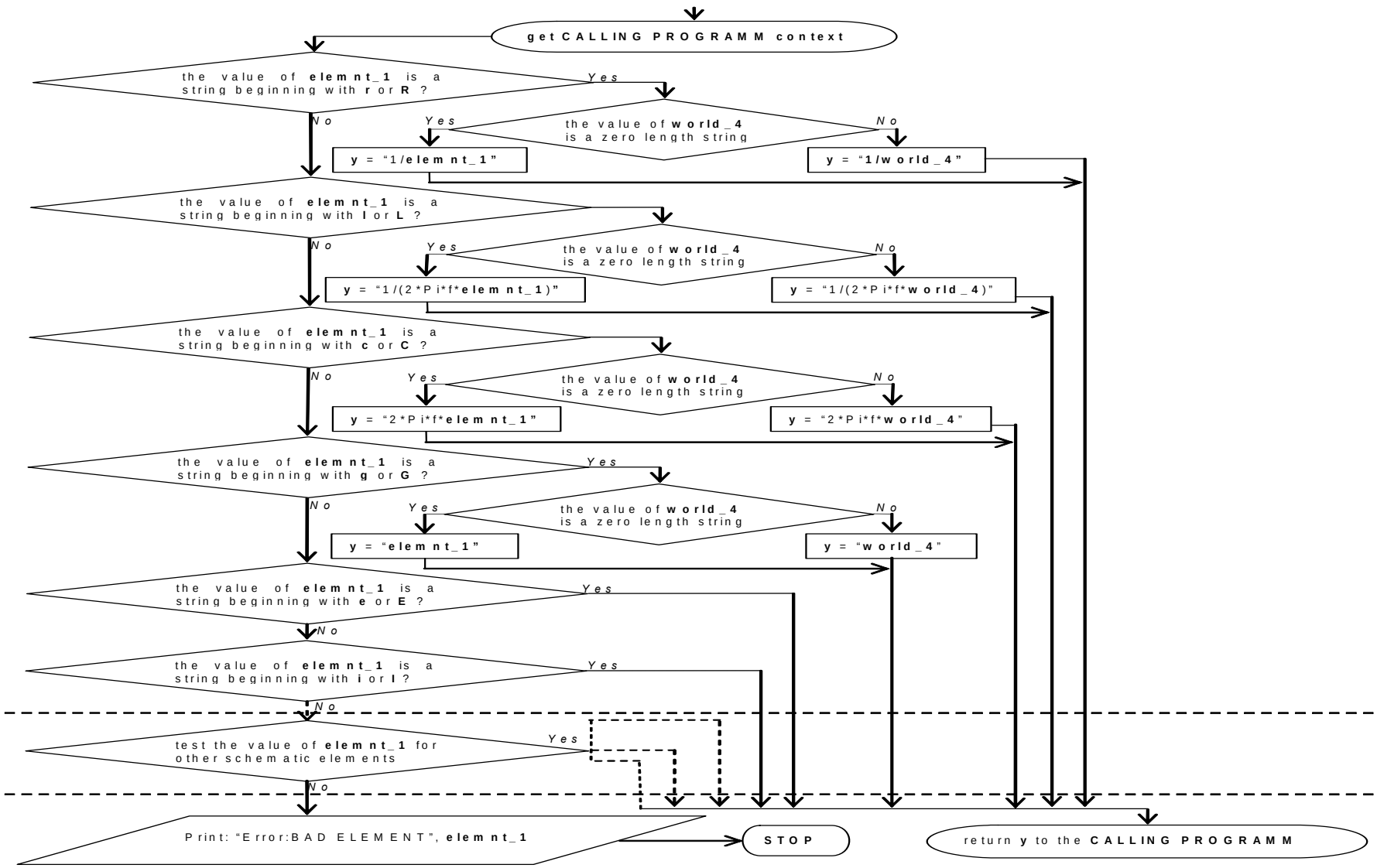

Fig. 7 - Subroutine "elemt_to_y": calculates the admittances of the schema elements