# Symbolic computation for electrical circuits

Dejan Djukic
*FIT*
*Alfa University*
Belgrade, Serbia
0000-0001-7581-148X

***Abstract*—In this work, the application of symbolic computation to problems of electrical engineering has been presented. In particular, a method has been designed for deriving a symbolic equivalent impedance of a circuit comprising an arbitrary number of resistors, capacitors, and inductors, connected in series or in parallel. The supporting environment for performing symbolic computations is Prolog programming language. This is a computing environment capable of effectuating symbolic computation and logical inference, and as such, it is a member of a wide domain of Artificial intelligence (AI). In this work, the main development ideas, the illustrative parts of the created Prolog code, and the results of its execution on sample networks, have been presented.**

***Keywords—symbolic computation, artificial intelligence, logic programming, Prolog programming language, electrical engineering, electrical circuits, RLC networks, impedance, Laplace transform***

## I. Introduction

Linear dynamic electrical circuits composed from resistors, inductors, and capacitors (RLC networks) are ubiquitous in all domains of electrical an electronic engineering, from power supply systems to wireless communication systems [1], [2]. There are good software solutions for simulating such networks, and for optimising the values of their components [3], [4], [5], [6], [7], [8], [9]. Nevertheless, in searching for new solutions, it may be important to try novel circuit topologies. Such task will be greatly facilitated if symbolic analytical mathematical expressions describing such circuits are known. Symbolic computation in cases of RLC networks entails application of a sequence of transformations of symbolic expressions.

Performing only numerical simulation of such circuits may be done readily and very efficiently. The results may be presented in the form of tables, or graphically. However, for a true insight in the behaviour of a circuit, an analytical form of a circuit's characteristics is needed.

Symbolic computation in cases of RLC networks entails application of a sequence of transformations of symbolic expressions. All the steps in these transformations are of light or moderate complexity, and are easily performed by humans. However, in case of complex circuits, operating on a large number of symbols presents an increased cognitive load, which may be a source of errors [10]. For this reason, there is a need for utilisation of computer aided symbolic computation when working with RLC networks. Computer, i.e. software systems enabling symbolic computation will be a great aid to engineers and designers.

Nowadays, pre-trained transformer and generative neuronal networks trained on large data sets, commonly known as the LLM, are being used in all and sundry cases pertaining to text and image applications [11]. Such systems are developed by using extremely large data sets with information in a correct form [12]. This is, so called, the *example based artificial intelligence* (AI). Whilst applications of the LLM for symbolic computation seem to be appearing recently [13], [14], it is doubtable that there exist sufficiently large sets of examples of symbolic computations with electrical circuits, necessary for their training.

Thus, for these kinds of applications, the author proposes to recourse to an older kind of artificial intelligence, now largely abandoned, which is the *knowledge based artificial intelligence* (AI) [15], [16], [17]. In this kind of artificial intelligence, the machine performs reasoning by applying transformation rules set by knowledgeable humans [18], [19].

## II. Symbolic computation with prolog

Programming language Prolog has been originally devised as an aid to proving mathematical theorems [20], [21]. This is performed by repeatedly applying transformations to a symbolic expression representing the theorem to be proved, until the conformance of this expression with previously established true symbolic sets, i.e. with axioms, has been achieved [22], [23], [24]. But the usefulness of Prolog goes beyond this, as the ability of Prolog for manipulation of symbolic expressions may be profitably used also in deriving mathematical descriptions of electrical circuits.

Programming paradigm of Prolog is the declarative programming. Declarative programming has been rather neglected in the recent years, and it has conceded place to object oriented programming, imperative programming, and, to some extent, to functional programming. Declarative programs are essentially composed from lists of transformational rules, called the clauses in Prolog. The rules are being applied exhaustively to symbolic expressions by the Prolog inference engine, until an expression of the correct form, i.e. the programme goal, has been reached.

In order to be able to deal with all manner of symbolic expressions, there is only one data structure in Prolog : the list. List data structure is truly universal, though its use has to be carefully deliberated. Its use in the case of electrical circuits is explained in the next section.

It has to be mentioned that Prolog is not the only programming system able to perform arbitrary symbolic computations. Some commercial software systems for

numerical or general mathematical computations, such as Matlab, and Mathematica, are being used in engineering practice for these very purposes. In fact, any of the programming languages of the broader Lisp family and their decendants, such as Common Lisp, Scheme, Clojure, Haskel, and even Javascript, can be also be used. Which software system is more efficient will depend on the implementation, and the convenience will depend on the skills and habits of the engineer using it. In the present work, Prolog has been used in order to demonstrate the validity of the approach using symbolic computation. Once the principles have been confirmed, they may easily be transferred to another, more efficient or more convenient platform.

## III. Electrical circuits and Prolog

### A. Textual representation of electrical circuits

Electrical circuits treated in this work are composed from resistances (R), inductances (L), and capacitances (C) connected in series or in parallel. At the beginning of the processing, the circuit elements are represented conventionally, by a letter indicating the type of the element, R, L, or C, followed by the element determinant ( a letter, a number, or a combination of these ).

In order to facilitate entering the circuit description into the Prolog symbolic computation system, a textual circuit description has been used. This textual circuit description has been devised as a slightly modified electricians' notation. Together with the symbols representing circuit elements, it uses two symbols indicating a composition operator : "`||`" for a parallel connection, and "`--`" for a series connection. In addition, the symbols of round brackets are used for element grouping. The examples of textual representation of RLC networks are given for circuits in Figures 1, 2 and 3. The textual representation proposed there for the circuit in Figure 1 is "`La--Rb`", for that in Figure 2 is "`Ca||Rb`", and for the slightly more complex circuit presented schematically in Figure 3 is "`(L1--R2--C3)||C4`".

### B. Representation of electrical circuits in Prolog

Firstly, the circuit elements are represented in Prolog as lists, as it has been announced in the previous section. For each element, the list contains two elements, a textual symbol of the type of the element, and a textual symbol of the determinant part of the element symbol. Examples of the list representation for the elements used in this work are shown in Tables 1, 2 and 3.

Then, also in Tables 1, 2, and 3, Laplace domain impedances and their Prolog representations are shown. The Prolog representations are formed as nested lists, which will be explained below.

It is important to notice that impedances in Laplace domain are rational functions. It may be shown that, in general, the form of an impedance in Laplace domain is

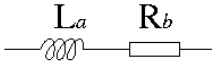



Fig. 1. Series L R network

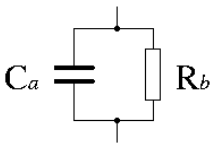



Fig. 2. Parallel R C network

$$Z(s)=\frac{Q(s)}{P(s)}=\frac{\sum_{j=0}^{m} s^j q_j}{\sum_{i=0}^{n} s^i p_i}=\frac{q_0+s\,q_1+s^2 q_2+\dots}{p_0+s\,p_1+s^2 p_2+\dots} \quad (1)$$

In Prolog, a rational function will be represented as a list with two members : "`Z=[Q,P]`", where "`Q`" and "`P`" are polynomials. The representation of a polynomial is also a list. E.G. polynomial $Q(s)=q_0+s\,q_1+s^2 q_2$ is represented as "`Q=[[0,q0],[1,q1],[2,q2]]`". Further on, coefficients of the polynomials, which are in the form of a sum of products are also represented in Prolog as lists. For example, for coefficient $x=ab+cde+fg$ its Prolog form is "`X=[[a,b],[c,d,e],[f,g]]`". One exception is the case where the polynomial coefficient is a numerical constant. Then, the coefficient is represented by the number itself. E.G. if the polynomial in question is $Q(s)=1+s(ab+c)+s^2(de+fg)$, its representation in in Prolog is

```
Q=[[0,1],[1,[[a,b],[c]]],
   [2,[[d,e],[f,g]]]]
```

### C. Symbolic computation of impedances in Prolog

Let two impedances connected in series be $Z_1(s)=\frac{Q_1(s)}{P_1(s)}$ and $Z_2=\frac{Q_2(s)}{P_2(s)}$. The resulting impedance is the result of the series transformation, denoted functionally as $f_s$, is the sum of the impedances :

$$Z=f_s(Z_1,Z_2)=Z_1+Z_2=\frac{Q_1}{P_1}+\frac{Q_2}{P_2}=\frac{Q_1P_2+Q_2P_1}{P_1P_2} \quad (2)$$

Resulting impedance of a parallel connection of the same two impedances is computed by a transformation denoted functionally as $f_p$ :

$$Z=f_p(Z_1,Z_2)=\frac{1}{\frac{1}{Z_1}+\frac{1}{Z_s}}=f_r(f_s(f_r(Z_1),f_r(Z_2))) \quad (3)$$

TABLE I. REPRESENTATION OF RESISTANCES

| Circuit element symbol | R1 |
|---|---|
| Prolog representation of the element | `['R','1']` |
| Impedance in Laplace domain | $R_1$ |
| Prolog representation of the impedance | `[[[0,[['R','1']]]],[[0,1]]]` |

where $f_r$ is a function transforming an impedance into its reciprocal value.

$$Z=\frac{P_1}{Q_1}=f_r(Z_1)=f_r(\frac{Q_1}{P_1}) \qquad (4)$$

Clearly, after any such transformation, the form of the impedance remains the rational function. The transformation itself requires just operations of addition and multiplication of polynomials. Prolog clause producing the reciprocal value is extremely simple:

```
z_rec([Q,P],[P,Q]) .
```

Functor for performing impedance transformation in case of parallel connection, "`z_par`" , is performed through Prolog functor performing impedance transformation in case of series connection, "`z_ser`" , and through the functor for producing the reciprocal value, "`z_rec`".

```
z_par(Z1,Z2,Z):-
    z_rec(Z1,RZ1) ,
    z_rec(Z2,RZ2) ,
    z_ser(RZ1,RZ2,RZ) ,
    z_rec(RZ,Z) .
```

Functor "`z_ser`" is a bit more involved, as it uses further functors performing addition and multiplication of polynomials.

```
z_ser([Q1,P1],[Q2,P2],[Q,P]):-
    p_mult(Q1,P2,AQ1) ,
    p_mult(Q2,P1,AQ2) ,
    p_add(AQ1,AQ2,Q) ,
    p_mult(P1,P2,P) .
```

T :S

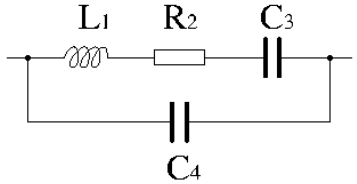
| Circuit e |  |
|---|---|
| Prolog re the | |
| Impeda d | |
| Prolog re the impedance | `[[[0,1]],[[1,[['C','1']]]]]` |



Fig. 3. Mixed R-L-C network

Prolog clauses for addition "`p_add`" and multiplication "`p_mult`" of polynomials are recursive, which requires some care with treating special cases. They are not presented here in full, however, the main direction of their development is briefly sketched. Adding polynomials requires adding their coefficients. Thus we have

$$(ab+cd)+(fg+h)=(ab+cd+fg+h) \qquad (5)$$

In Prolog, this can be achieved rather simply as :

```
c_add(C1,C2,C) :- append(C1,C2,C) .
```

Then, we would have the following sequence of clauses passing as correct, i.e. as true :

```
...
C1 = [[a,b],[c,d]] ,
C2 = [[f,g],[h]] ,
c_add(C1,C2,C) ,
C = [[a,b],[c,d],[f,g],[h]],
...
```

Multiplication of coefficients requires double recursion. Thus, the Prolog code is somewhat longer, though, it is conceptually neither more complex nor more difficult to analyse. For example, if the required computation is as follows :

$$(ab+cd)(fg+h)=(abfg+abh+cdfg+cdh) \qquad (6)$$

then a corresponding Prolog functor is :

```
coef_mult(C1,C2,C) :-
    coef_mult_a(C1,C2,C, [] ) .
coef_mult_a([],_,S,S) .
coef_mult_a([AC1|RC1],C2,S,S1) :-
    coef_mult_b(AC1,B,M,[]) ,
    append(S1,M,S2) ,
    coef_mult_a(RC1,B,S,S2) .
coef_mult_b(_,[],M,M) .
coef_mult_b(AC1,[AC2|RC2],P,M1) :-
    append(M1,[[AC1,AC2]],M2) ,
    coef_mult_b(AC1,RC2,M,M2) .
```

TABLE III. REPRESENTATION OF INDUCTANCES

| Circuit element symbol | L1 |
|---|---|
| Prolog representation of the element | `['L','1']` |
| Impedance in Laplace domain | $sL_1$ |
| Prolog representation of the impedance | `[[[1,[[['L','1']]]]],[[0,1]]]` |

When applied to polynomial coefficients, this functor produces the followint result :

```
...
C1 = [[a,b],[c,d]] ,
C2 = [[f,g],[h]] ,
c_mult(C1,C2,C) ,
C =[[a,b,f,g],[a,b,h],[c,d,f,g],
    [c,d,h]] ,
...
```

Finally, the complex list structures, representing impedances in Laplace domain, are put in a text form that is more easily readable by humans.

With the help of this Prolog code, initial textual descriptions of RLC circuits have been transformed into a form of human readable symbolic expressions of Laplace domain impedances.

## IV. EXAMPLES

Symbolic computation of impedances of RLC networks has been tested, and the results of the tests are presented below. The examples of circuits are rather simple, but they are deemed appropriate for the purpose of illustrating the principle of working of the system.

### A. *Example 1 (Figure 1)*

The circuit in Figure 1 is represented by an initial textual representation of the circuit :

```
L1 -- R2
```

Entry form for transformation by Prolog is produced by the clauses below. In order to facilitate the analysis, the results of the processing are also given.

```
zL(['L','1'],Z1) ,
Z1 = [[[1,[[['L','1']]]]],[[0,1]]] ,
zR(['R','2'],Z2) ,
Z2 = [[[0,[[['R','2']]]]],[[0,1]]] ,
```

After the symbolic transformation has been applied, the resulting expression is :

```
z_ser(Z1,Z2,Z) ,
Z=[[[0,[[['R','2']]]]],
    [1,[[['L','1']]]]],[[0,1]]] ,
```

This expression is further transformed into a human readable form :

```
R2 + s ( L1 )
```

### B. *Example 2 (Figure 2)*

The expressions for the circuit in Figure 2 are as follows :

Textual representation :

```
C1 || R2
```

Initial Prolog representation :

```
zC(['C','1'],Z1) ,
Z1 = [[[0,1]],[[1,[[['C','1']]]]]] ,
zR(['R','2'],Z2) ,
Z2 = [[[0,[[['R','2']]]]],[[0,1]]] ,
```

Transformed impedance : .

```
par( Z1 , Z2 , Z )
Z = [[[0,[[['R','2']]]]],
    [[0,1],[1,[[['R','2'],
    ['C','1']]]]]] ,
```

The human readable form of the obtained impedance is :

```
( R2 ) / ( 1 + s ( R2 C1 ) )
```

### C. *Example 3 (Figure 3)*

Somewhat more complex circuit is given in Figure 3. The expressions for this circuit are as follows :

Textual circuit representation :

```
( L1 -- R2 -- C3 ) || C4
```

Initial Prolog representation :

```
zL(['L','1'],Z1) ,
Z1 = [[[1,[[['L','1']]]]],[[0,1]]] ,
zR(['R','2'],Z2) ,
Z2 = [[[0,[[['R','2']]]]],[[0,1]]] ,
zC(['C','3'],Z3) ,
Z3 = [[[0,1]],[[1,[[['C','3']]]]]] ,
zC(['C','4'],Z4) ,
Z4 = [[[0,1]],[[1,[[['C','4']]]]]] ,
```

Transformed impedance : .

```
ser( Z1 , Z2 , Zx )
ser( Zx, Z3 , Zy )
par( Zy , Z4 , Z )
Z = [[[0,1],
    [1,[[['C','3'],['R','2']]]],
    [2,[[['L','1'],['C','3']]]]],
    [[1,[[['C','3']],[['C','4']]]],
    [2,[[['R','2'],['C','3'],
    ['C','4']]]],[3,[[['L','1'],
    ['C','3'],['C','4']]]]] ,
```

It has to be remarked that it is almost impossible for a human to understand the list representation of the impedance produced by Prolog symbolic computation. This is, however, quite normal and is to be expected, as this representation is not at all intended to be read by humans. This list has to be transformed into a human readable form, and the result of this transformation is : .

```
( 1 + s ( C3 R2 ) + s^2 ( L1 C3 ) ) /
( s ( C3 + C4 ) + s^2 ( R2 C3 C4 ) +
    s^3 ( L1 C3 C4 ) )
```

## V. Remarks and discussion

The presented method for symbolic calculation of RLC circuit impedances using Prolog is, at the moment, of limited applicability. In both power distribution systems, and in sub-microwave radio circuits, an essential circuit element is the transformer. If an RLC be initially formed by series and parallel connections of elements, which allows for an easy analysis, inserting a transformer would certainly require a fully developed system for solving circuit equations. Furthermore, a reasonable scope of use of a method for symbolic circuit analysis would necessarily require an ability to analyse two port circuits, and produce their input, output, and transfer parameters. At some time in the future, a Prolog method for symbolic solution of general circuit equations will be developed. It is for this reason that not much effort has been invested here into an input formulation of a circuit formed by parallel and series connections. A general circuit composition requires stating corresponding sets of nodes and branches. A suitable circuit grammar has already been defined by SPICE system for numerical circuit simulation, and it may be used in future variegations of this work.

## VI. Conclusion

A Prolog system for symbolic computation in electrical engineering has been presented. This system takes a textual description of a circuit composed from resistances, inductances, and capacitances. These elements are combined by parallel and series interconnections. The initial textual representation of the circuit is transformed into a symbolic, analytical representation of its impedance in Laplace domain. This impedance is then further transformed into its textual representation that is more easily readable by humans. Such a system may be of great use for the engineers, in circuit analysis and design, and, of course, for the students of engineering.

Another remark emanating through this work is that the methods of knowledge based artificial intelligence, and, in particular, Prolog programming language, have been wrongfully neglected in recent years. Admittedly, these do go contrary to the present hype of LLM and other systems based on example based artificial intelligence. On the other hand, these older methods are still of considerable use in engineering practice. Especially, this work also shows that the scope of use of Prolog programming language is not limited to the theoretical computer science and mathematical logic, and that it can be very useful in other domains, too, such as electrical engineering.

The presented system is, of course, not without its shortcomings. Firstly, not all circuits may be composed only by series and parallel connections. It is necessary to implement a more general method of circuit solving. An important circuit element both in radio engineering and in power systems is the transformer. The system has to be augmented by an appropriate representation of the transformer and a model of its effect in a circuit. Finally, perhaps also a better human readable form should be conceived.

## References


[1] Surutka, Jovan V. Osnovi elektrotehnike. Naučna knjiga, 1986.

[2] Ђорђевић, Антоније Р., Основи електротехнике, Академска мисао, 2016.

[3] Labovitz, Stuart Lewis, and AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING. Formal Verification of Digital Logic. Diss. Air Force Institute of Technology, 1991.

[4] Cois, A., et al. "Qualitative analysis in simulating analog circuits." Proceedings. IEEE International Symposium on Intelligent Control 1989. IEEE, 1989.

[5] Tanaka, Takushi. "Parsing Electronic Circuits in a Logic Grammar." IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING 5.2 .1993)

[6] Tanaka, Takushi. "Circuit grammar: knowledge representation for structure and function of electronic circuits." International Journal of Reasoning-based Intelligent Systems 1.1-2 ,2009, pp. 56-67.

[7] Eicher, J. W., and F. M. Brown. "Metaprogramming in digital simulation." Proceedings of the IEEE 1995 National Aerospace and Electronics Conference. NAECON 1995. Vol. 2. IEEE, 1995.

[8] Tanaka, Takushi. "KNOWLEDGE REPRESENTATION FOR ELECTRONIC CIRCUITS IN LOGIC PROGRAMMING." The Handbook on Reasoning-Based Intelligent Systems. 2013: 495-523.

[9] Clockskin, WF. "Logic programming and digital circuit analysis." The Journal of Logic Programming 4.1 , 1987, pp. 59-82.

[10] Sweller, J. and Chandler, P., 1991. Evidence for cognitive load theory. Cognition and instruction, 8(4), pp.351-362.

[11] Gou, F., Liu, J., Xiao, C., & Wu, J. Research on artificial-intelligence-assisted medicine: A survey on medical artificial intelligence. Diagnostics, 14(14), 2024, pp. 1472.

[12] Jindong Li, Yali Fu, Fengxiang Cheng, Yang Yang, Jiahong Liu, Hongce Zhang, Yutao Yue, Menglin Yang, Haoxuan Li, Liangming Pan, Zhouchen Lin. "A Survey on LLM Symbolic Reasoning". preprint, 2025, available at https://www.techrxiv.org/doi/full/10.36227/techrxiv.176538331.19733376

[13] Yang, Xiao-Wen, Jie-Jing Shao, Lan-Zhe Guo, Bo-Wen Zhang, Zhi Zhou, Lin-Han Jia, Wang-Zhou Dai, and Yu-Feng Li. "Neuro-symbolic artificial intelligence: Towards improving the reasoning abilities of large language models." arXiv preprint arXiv:2508.13678 (2025)

[14] Li, Y., Meng, R. and Duck, G.J., 2025. Large language model powered symbolic execution. Proceedings of the ACM on Programming Languages, 9(OOPSLA2), pp.3148-3176.

[15] Jabri, Marwan A. "BREL—a Prolog knowledge-based system shell for VLSI CAD." Proceedings of the 27th ACM/IEEE Design Automation Conference. 1991.

[16] Hrycej, Tomas. "A temporal extension of PROLOG." Journal of Logic Programming 15.1-2 , 1993, 113-145.

[17] Luger, George, and William Stubblefield. Artificial intelligence: structures and strategies for solving complex problems. Benjamin/Cummins, 2004.

[18] da Silva, António Ferreira, and Adriano A. Santos. "Symbolic Manipulation for Optimization of Boolean Functions for Control of Pneumatic and Electropneumatic Circuits." 2021 16th Iberian Conference on Information Systems and Technologies (CISTI). IEEE, 2021.

[19] Djukic, Dejan and Popovic, Stefan. "Artificial intelligence in electrical engineering". Proceedings of AIIT 2024 conference, 2024, pp. 231-237.

[20] Presic, Slavisa. "PROLOG Relacijski jezik." 1996 .

[21] Treleaven, Philip Co, and Isabel Gouveia Lima. "COMPUTERS and PROGRAMMING LANGUAGES for Al.", CLEI - 1985, Vol 1. pp. 713 - 728

[22] Власенко, А. Ю., Мичуров, М. А., Царёв, В. Д., & Курбатов, М. А. (2024). Построение комплекса автоматизированной отладки фрагментированных программ. Вестник НГУ. Серия: Информационные технологии, 22(1), pp. 5-20.

[23] Da Silva, A. F., Santos, A. A., Pereira, F., Felgueiras, C., & Moreira, A. P. (2024, June). Application of Artificial Intelligence to Improve Skills in Electropneumatic Control Systems to Engineering Students. In International Conference Innovation in Engineering pp. 384-398. Cham: Springer Nature Switzerland.

[24] Schrijvers, Tom, Birthe Van Den Berg, and Fabrizio Riguzzi. "Automatic Differentiation in Prolog." Theory and Practice of Logic Programming 23.4 (2023): pp. 900-917.

[25] Biasizzo, Anton, and Franc Novak. Model-Based Diagnosis of Analog Circuits with CLP (R). Technical report CSD-TR-95-9, Jozef Stefan Institute, Ljubljana, Slovenia, 1995.G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.