Mathematica pattern matching for identification of Foster functions

Dejan V. Tošić and Miroslav D. Lutovac, *Senior Member IEEE*

***Abstract* — Present-day computer algebra systems offer new programming paradigms that can solve many scientific and engineering problems more efficiently than the classic programming styles, such as procedural or object-oriented programming. This contribution focuses on symbolic pattern matching and rule-based programming paradigm, which is exemplified by a real-life electric network synthesis problem. The benefits of the new approach are highlighted and the entire code in *Mathematica* is presented and explained. The symbolic programming language is suggested for rapid software development and algorithm prototyping in the field of electric filter design.**

***Keywords* — Pattern matching, Foster functions, Network synthesis, *Mathematica*.**

I. Introduction

NETWORK synthesis is a process in which we have a given electric network function, and we try to find an electric network whose properties will be described by that function. In most cases, we try to do so for a voltage transfer function. The first question that comes into mind is whether a network function can be realized as a network composed of passive electrical elements: capacitors, inductors, resistors, and transformers. In many cases, we need to realize LC impedances or admittances, that is, lossless one-port networks. [1]–[5]

In this paper, we present a use of computer algebra systems (CAS) for identifying whether a network function can be realized by a passive one-port LC network. In this case, the use of an automated symbolic technique is a natural choice because manual computation is practically impossible, error-prone, and a fatiguing labor. We present our original symbolic algorithm in *Mathematica* [6]–[8], as well.

II. Foster functions

Network characteristics can be studied in terms of voltage and current parameters defined at the network ports. A one-port network can be characterized, in frequency domain ($s = \sigma + j\omega$), by its input impedance $Z(s)$ or input admittance $Y(s)$ defined in terms of port voltage $V(s)$ and port current $I(s)$:

$$Z(s) = \frac{1}{Y(s)} = \frac{V(s)}{I(s)} \tag{1}$$

Analytical properties of input immittances of passive, lossless one-ports are of fundamental importance in the synthesis of electrical networks, such as electrical filters. Some basic properties of these immitance functions are summarized below.

Driving point impedance $Z(s)$ (and $Y(s)$) of a linear, time invariant, passive and lumped one-port is a positive real function:

1. $Z(s)$ is real for s real,
2. $\mathrm{Re}(Z(s)) > 0$ for $\mathrm{Re}(s) > 0$.

Let

$$Z(s) = \frac{N(s)}{D(s)}.$$

Then:

1. $N(s)$ and $D(s)$ are Hurwitz polynomials (i.e. they have no roots in the right half of the $s = \sigma + j\omega$ plane).
2. The roots on the $s = j\omega$ axis, including those at $s = 0$ and $s = \infty$, if they exist, must have unit multiplicity with positive residues.
3. The degrees of $N(s)$ and $D(s)$ may differ at most by unity.

Foster's Reactance Theorem further details some properties of lossless networks (networks consisting of pure reactances).

For a positive real rational function

$$Z(s) = \frac{1}{Y(s)}$$

to be realizable as the driving point impedance of a lossless one-port, the necessary and sufficient condition is that it should be expressible in the form

$$Z(s) \text{ or } Y(s) = \frac{a_n(s^2+\omega_1^2)(s^2+\omega_3^2)\cdots}{b_m(s^2+\omega_2^2)(s^2+\omega_4^2)\cdots} \tag{2}$$

where a_n and b_m are constants and

$$0 < \omega_1 < \omega_2 < \omega_3 \cdots$$

(Interlacing poles and zeros, all on $j\omega$ axis).

Foster's Theorem restricts the degrees of the numerator, n, and denominator, m, by requiring that they must differ

D. V. Tošić, University of Belgrade, School of Electrical Engineering, Bulevar kralja Aleksandra 73, 11120 Belgrade, Serbia (tel.: 381.11.3218352; fax: 381.11.3248681; e-mail: tosic@etf.bg.ac.yu).

M. D. Lutovac, University of Belgrade, School of Electrical Engineering, Bulevar kralja Aleksandra 73, 11120 Belgrade, Serbia (tel.: 381.11.137885; fax: 381.11.3248681; e mail: lutovac@etf.bg.ac.yu).

by unity. In other words, if the numerator is an even degree, the denominator is odd, and vice versa.

The following properties of lossless networks can be deduced:

1. Input immittance must have either a single pole or a single zero at both $s = 0$ and $s = \infty$.

2. $Z(\mathrm{j}\omega)$ is purely reactive, $Z(\mathrm{j}\omega) = \mathrm{j}X(\omega)$, and $X(\omega)$ is always an increasing function of frequency. The rational functions satisfying these requirements are called *Foster functions*.

3. Since all poles of input immittances are on the imaginary axis, they can always be expanded as

$$Z(s) \text{ or } Y(s) = \frac{k_0}{s} + k_\infty s + \sum \frac{2k_i}{s^2 + \omega_i^2} \tag{3}$$

where the coefficients k are real and positive. Physically, they correspond to simple network elements, inductors and capacitors.

III. Symbolic Pattern Matching Programming Paradigm

A common approach taken in modern software engineering is to combine various programming paradigms, such as procedural programming or object-oriented programming, to achieve the desired goals. Programming has become a task of knowledge accumulation that tells the computer what to know, when to use information, and how to apply the knowledge in solving problems.

Contemporary computer algebra systems (CAS) introduced a qualitatively new suite of software tools and programming concepts. The leader in implementing CAS, *Mathematica* by Wolfram Research, launched an original programming language, which gives a different perspective to software realization of scientific and engineering algorithms [9].

At the core of *Mathematica* are its highly developed symbolic language and the foundational idea that everything – data, programs, mathematical formulas, lists, graphics, and documents – can be represented as symbolic expressions. The advanced notion of expressions is a crucial unifying principle and it is the fact that every object has the same underlying structure.

Expressions are used to specify operations and to maintain a structure, which can then be acted on by the operations. A prototypical example of an expression is

```
f[x,y,z,…]
```

where the symbol **f** is the head of the expression, the symbols **x,y,z,**… are the arguments, and the square brackets are delimiters; the head and the arguments itself can be expressions, as well.

The parentheses are used exclusively for grouping following standard mathematical notation to specify the precedence of operators.

The use of distinct delimiters for arguments is a unique concept important for symbolic programming and it adds a new level of flexibility to the very concept of programming: the pattern matching and transformation rules can be applied to both expression heads and expression arguments.

Patterns are used to represent classes of expressions with a given structure. The main power of patterns comes from the fact that many operations can be done not only with single expressions, but also with patterns that represent whole classes of expressions.

The fact that patterns specify the structure of expressions is crucial in making it possible to set up transformation rules which change the structure of expressions, while leaving them mathematically equal.

IV. Identification of Foster functions

The pattern matching and rule-based programming paradigm is best illustrated by an example which demonstrates the uniqueness and benefits of the symbolic language concept.

Consider an essential problem of electric network synthesis and practical filter design [10,11]: given a transfer function, e.g.

$$H(s) = \frac{s^4 + 3s^2 + 3}{s^5 + 5s^3 + 6s} \tag{4}$$

determine whether the function can be realized as a driving-point impedance of an electrical network of interconnected capacitors and inductors.

According to the Foster's Reactance Theorem, an algebraic rational function to be realizable as the driving-point impedance of a lossless one-port electrical network can always be expanded as

$$H(s) = \frac{a_{-1}}{s} + a_0 s + \sum_i \frac{a_i s}{b_i s^2 + c_i} \tag{5}$$

where all coefficients are positive. *Mathematica* code that performs the required test, based on the Foster's theorem, is given in Fig. 1.

The code is concise, elegant, readable, easy-to-maintain, and self-explanatory.

The function **Apart** expands the transfer function (**H**) into partial fractions and the function **List** converts the expansion to a list of terms (**partialFractions**).

The function **MatchQ** performs the desired pattern matching, term by term, under the conditional rule that the coefficients should be positive numbers; it returns a list of logical constants (**patternMatchTest**).

The function **And** returns true if all terms pass the pattern match, otherwise it returns false.

The intermediate results are shown in Fig. 2. The second term does not match the pattern so the transfer function is not realizable as the driving-point impedance of a lossless one-port electrical network.

Symbolic pattern matching and rule-based programming paradigm is an important issue and a choice of preference for rapid software development and algorithm prototyping. It is the key programming paradigm involved in the development and implementation of *SchematicSolver* [12].

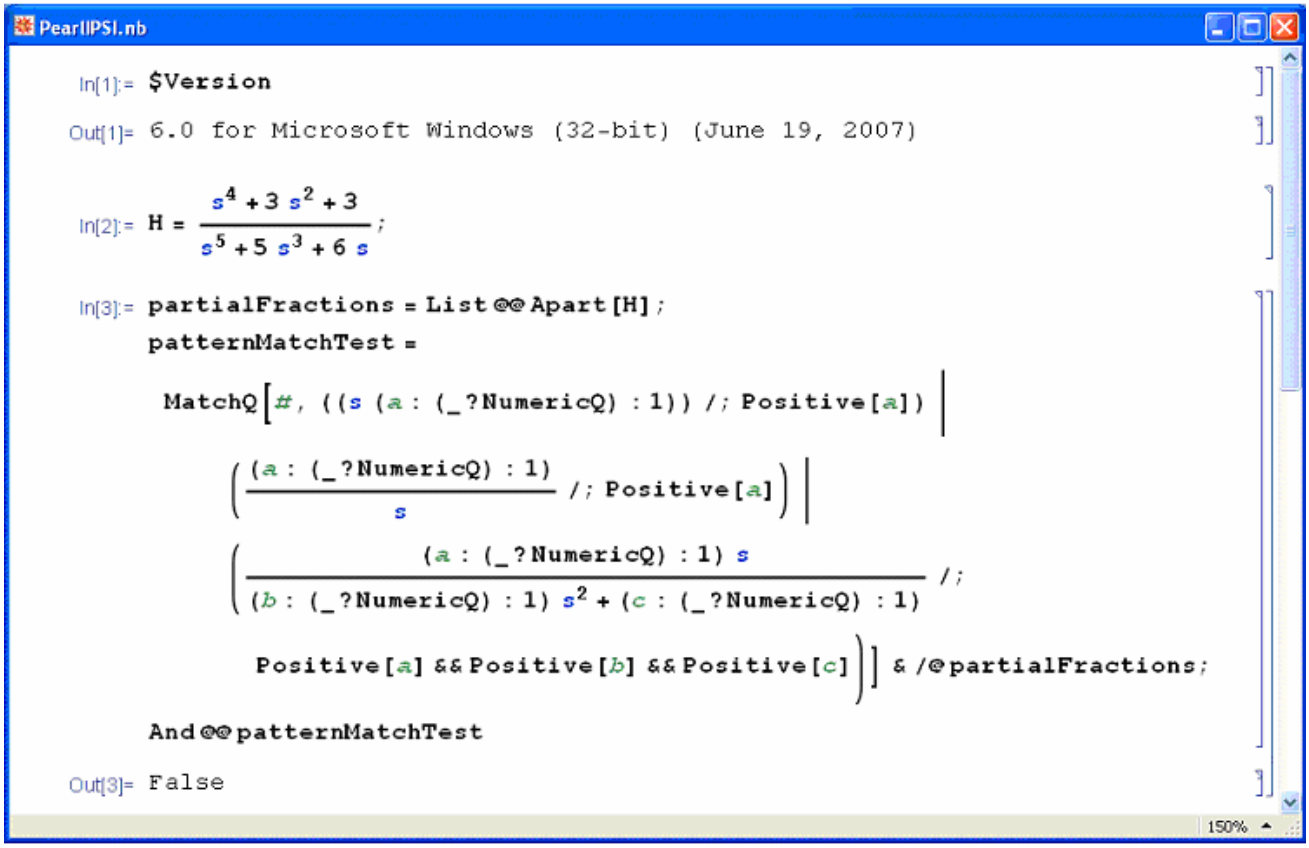

Fig. 1. *Mathematica* code for identification of Foster functions.

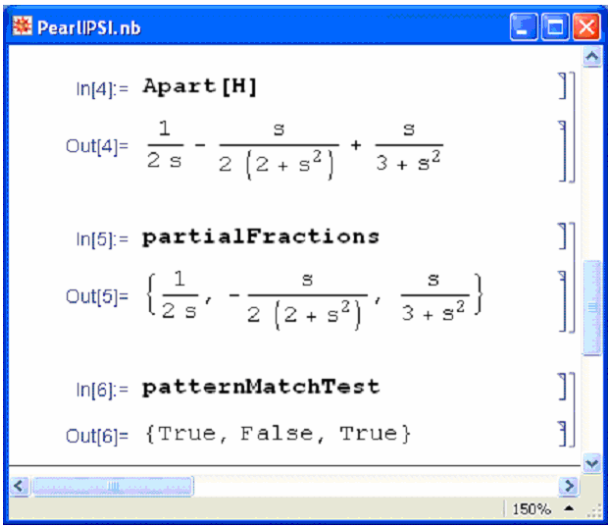

Fig. 2. Intermediate *Mathematica* results. Obviously, the transfer function is not realizable as the driving-point impedance of a lossless one-port electrical network.

I. Conclusion

Computer algebra system has been used for identifying whether a network function can be realized by a passive one-port LC network. The use of an automated symbolic technique has been identified as a natural choice and it has been suggested for rapid prototyping of algorithms in the field of electric network synthesis and filter design. An original *Mathematica* code has been given for identifying Forster functions. The code has been based on the pattern matching and rule-based programming paradigm. The future research directives might include a complete one-port network synthesis based on symbolic computation.

Acknowledgement

We thank Ministry of Science and Environmental Protection of the Republic of Serbia for partial support of our research on this topic (Project No. TR-6154).

Literatura

[1] N. Balabanian, *Network synthesis*. Englewood Cliffs, NJ: Prentice Hall, 1958.
[2] E. A. Guillemin, *Synthesis of passive networks*. New York, NY: John Wiley & Sons, 1957.
[3] M. E. Van Valkenberg, *Introduction to modern network synthesis*. New York, NY: John Wiley & Sons, 1960.
[4] L. Weinberg, *Network analysis and synthesis*. New York, NY: McGraw-Hill, 1962.
[5] A. I. Zverev, *Handbook of filter synthesis*. New York, NY: John Wiley & Sons, 1976.
[6] S. Wolfram, *The Mathematica Book*. Cambridge, MA: Cambridge University Press, Wolfram Media, 2003.
[7] *Mathematica*, http://www.wolfram.com/, Version 6.0.1 released June 19, 2007.

[8] R. E. Maeder, *Computer Science with Mathematica: Theory and Practice for Science, Mathematics, and Engineering*. Cambridge, UK: Cambridge University Press, 2000.
[9] D. V. Tošić, M. D. Lutovac, "Advances in symbolic simulation of systems," *IPSI Transactions on Advanced Research*, vol. 3, no. 1, pp. 9–14, 2007.
[10] Wai-Kai Chen (Editor), *The Electrical Engineering Handbook*. Burlington, MA: Elsevier Academic Press, 2004.
[11] M. D. Lutovac, D. V. Tošić, B. L. Evans, *Filter Design for Signal Processing using MATLAB and Mathematica*. Upper Saddle River, NJ: Prentice Hall, 2001.
[12] M. D. Lutovac, D. V. Tošić, *SchematicSolver*, Version 2.1, http://www.wolfram.com/products/applications/schematicsolver/, for *Mathematica* 6 released July 12, 2007.