SYMBOLIC ANALYSIS OF LINEAR NETWORKS

by

Jacob Katzenelson and Shalom Tsur

Memorandum No. ERL-M322

December 28, 1971

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

SYMBOLIC ANALYSIS OF LINEAR NETWORKS

Jacob Katzenelson†

Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa, Israel

Shalom Tsur
Department of Computer Science
Hebrew University
Jerusalem

†Currently on sabbatical leave at

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

Work done at the Technion, Israel Institute of Technology and completed at the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, where it was sponsored by the National Science Foundation Grant GK-10656X1.

Consider a linear time invariant network consisting of RLC and of dependent and independent sources. The value of one of the independent sources is the input, e_{in}, and the value of a network variable, e_0, is the output. Let Z be the impedance of one of the networks branches or the gain of one of the independent sources. We are interested in the effect that changes in Z have on the transfer function, H, $H = \frac{e_0(j\omega)}{e_i(j\omega)}$.

One way to summarize such information is to consider the transfer function as a function of both the frequency ω and the parameter Z and to plot $|H|$ and $\measuredangle H$ as a function of ω for $Z = Z_1, Z_2, \ldots$, etc. (Fig. 1).

Such plots are usually generated by a digital computer and are used as a sensitivity type study which aids the circuit designer to pick a suitable value for Z.

It is clear that such plots can be obtained for more than one parameter. However, usually the number of parameters used is small, one, two, or at most three, since it is very difficult to get any insight when many parameters vary simultaneously.

There are several algorithms which can be used to calculate such plots. The most common one writes the network equations and for each value of ω and Z solves the simultaneous linear equations by means of Gauss elimination [1]. The basic idea in the method presented here is to compute a symbolic expression for the transfer function as a function of s and Z. Once a symbolic expression is available, the value of H(s,Z) for any range of ω and Z can be easily obtained.

For networks of the type considered here (excluding degenerated cases) H(s,Z) has the form

$$H(s,z) = \frac{P_1(s)Z + Q_1(s)}{P_2(s)Z + Q_2(s)}$$

where P_1, P_2, Q_1, Q_2 are polynomials in s. Thus, a symbolic analysis amounts to a method of calculating the coefficients of these polynomials and a computer program that produces the required plots for any range of ω and Z.

The method of getting the symbolic expression is an extension of a method by Pottle [2]. In his work Pottle indicated that the method has numerical troubles. One of our motivations in studying the method was to find out exactly where and why numerical problems appear and, if possible, overcome them. We wrote a program [3] (Elliot 503, 39 bit words) and studied the problem experimentally by calculating critical parts of the algorithm on a computer with a larger word size (CDC 6400, 60 bit word, double and triple precision). The results suggested several improvements which have been introduced and are discussed here.

In the following the method is first introduced and the discussion of numerical problems comes second.

The Transfer Function

In the following the transfer function is calculated as a function of s and one parameter, Z. Similar derivation can be made for more than one parameter.

Let us first assume that Z is the impedance of one of the network branches. Let e_2 and $-i_2$ denote the voltage and current of the branch. Consider the network η obtained by replacing the element Z with a voltage source e_2. Let $\underset{\sim}{u} \triangleq \begin{pmatrix} e_{in} \\ e_2 \end{pmatrix}$ be the input vector for η and $\underset{\sim}{x} \triangleq \begin{pmatrix} e_0 \\ i_2 \end{pmatrix}$ be the output vector. Except in degenerated cases [2,5] one can write state equations which describe the network behavior. Let these equations be

$$\dot{\underset{\sim}{q}}(t) = A\underset{\sim}{q}(t) + B\underset{\sim}{u}(t) \tag{1}$$

$$\underset{\sim}{x}(t) = C\underset{\sim}{q}(t) + D\underset{\sim}{u}(t) \tag{2}$$

We have a third equation at our disposal;

$$Z(s) = -\frac{e_2(s)}{i_2(s)} \tag{3}$$

When Z is a branch impedance, we can chose either voltage or current as input variable at ② ②' of Fig. 2. The other variable becomes the output variable. A small change is needed when we consider a controlled source. The change is illustrated in Fig. 3. In this example r_m is the branch which is extracted from the network. Note that in this example the input admittance at ② ②' can be zero and in order to avoid problems the voltage should be chosen as an input variable and the current as output variable.

Taking the Laplace transform of the state equation, assuming initial conditions are zero, we get

$$\underset{\sim}{x}(s) = \{C(Is - A)^{-1} B + D\}\underset{\sim}{u}(s).$$

Denote:

$$A^* = (Is - A)^{-1};$$

$B = (b_1, b_2)$, b_1 and b_2 are column vectors;

$$C = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \quad c_1 \text{ and } c_2 \text{ are row vectors;}$$

$$D = \begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix}, \quad d_{ij} \text{ are scalars.}$$

$$y(s) = \frac{1}{Z(s)} \quad .$$

Using the above notations we get

$$\begin{pmatrix} e_0(s) \\ i_2(s) \end{pmatrix} = \begin{pmatrix} c_1A^*b_1+d_{11} & c_1A^*b_2+d_{12} \\ c_2A^*b_1+d_{21} & c_2A^*b_2+d_{22} \end{pmatrix} \begin{pmatrix} e_{in}(s) \\ e_2(s) \end{pmatrix}$$

and using (3)

$$\frac{e_0(s)}{e_{in}(s)} = \frac{(c_1A^*b_1+d_{11})y(s)+(c_1A^*b_1+d_{11})(c_2A^*b_2+d_{22})-(c_1A^*b_2+d_{12})(c_2A^*b_1+d_{21})}{(c_2A^*b_2+d_{22})+y(s)} \tag{4†}$$

Since A^* is a ratio of polynomials in s the equation can be brought to the form

$$\frac{e_0(s)}{e_{in}(s)} = \frac{P_1(s)\ y(s) + Q_1(s)}{P_2(s)\ y(s) + Q_2(s)}$$

---

†It might be of interest to note that (4) has the following interpretation

$$\frac{e_0}{e_{in}} =$$

$$\frac{\left\{\begin{matrix} \frac{e_0}{e_{in}} \text{ when } e_2=0, \\ \text{y-short circuit} \end{matrix}\right\} \cdot y(s) + \left\{\begin{matrix} \frac{e_0}{e_{in}} \text{ when y open circuit,} \\ y = 0 \end{matrix}\right\} \left\{\begin{matrix} \text{input admittance at } (2)\ (2') \\ \text{when } e_{in} = 0 \end{matrix}\right\}}{\left\{\begin{matrix} \text{input admittance at } (2)\ (2') \\ \text{when } e_{in} = 0 \end{matrix}\right\} + y(s)}$$

where $P_1(s)$, $P_2(s)$, $Q_1(s)$ and $Q_2(s)$ are polynomials. To get the coefficient of the polymomials one has to invert (Is - A). Fortunately this can be done rather simply using a formula by Soriau and Framme [4]:

Let A be an (nxn) matrix and let

$$(Is - A)^{-1} = \frac{T(s)}{g(s)}$$

where

$$T(s) = T_0 \; s^{n-1} + T_1 \; s^{n-2} + \ldots + T_{n-1} \; ;$$

T_i, i = 0, ..., n-1 are (nxn) matrices, with constant elements,

$$g(s) = s^n + g_1 \; s^{n-1} + \ldots + g_n$$

The g_i and T_i can be calculated as following:

$T_0 = I$ (I denotes the (nxn) unit matrix);

for k = 1, ..., n

$$g_k = - \frac{1}{k} \operatorname{trace}(T_{k-1} \; A)$$

$$T_k = T_{k-1} \; A + I g_k \; .$$

The last equation calculated for k = n yields zero for T_n,

$$0 = T_{n-1} \; A + g_n I$$

and provides a possibility to check and estimate the amount of error accumulated during the calculation.

Thus, the calculation of a symbolic expression proceeds in three steps:

(1) Deriving state equations for the network;

(2) Calculating inverse of (Is - A);

(3) Using equation (4) for getting the result.

In actuality step (2) and (3) are intermixed. As soon as a T_i is found it is multiplied by appropriate row and column vector and stored. Thus no more than one T_i has to be held in memory at one time.

Numerical Properties

While computing examples using the algorithm we come across several numerical problems or numerical troubles. These problems and ways to overcome them are described below.

(1) Consider the last stage of the algorithm. H(s,Z) is available, and we have to evaluate $H(j\omega,z)$ for $Z = z_1$ and $\omega_0 < \omega < \omega_1$. Once the value of Z is substituted H can be viewed as a ratio of two polynomials, say $H = \frac{p(s)}{q(s)}$. If p and q have common roots which are on or near to the $j\omega$ axis and the domain $\omega_0 < \omega < \omega_1$ (network has unobservable modes) we experienced difficulties in evaluating the value of H.

This phenomenon can be best illustrated with a simple example.

$$\text{Let} \quad H(s) = \frac{(s+1)(s+5)}{(s+1)(s+7)} = \frac{s^2 + 6s + 5}{s^2 + 8s + 7}$$

Evaluate at $-1 + \varepsilon$ where ε is very small.

$$H(-1+\varepsilon) = \frac{1 - 2\varepsilon + \cancel{\varepsilon}^2 - 6 + 6\varepsilon + 5}{1 - 2\varepsilon + \cancel{\varepsilon}^2 - 8 + 8\varepsilon + 7} = \frac{4\varepsilon}{6\varepsilon} = \frac{4}{6}$$

Note that the ε cancelled out and in this example the correct result was obtained. In actuality as a result of round off errors the "upstairs" ε and the "downstairs" ε are not equal. If ε is small then round off errors are responsible for a good part of its value and the ratio of the upstairs errors to downstairs errors is meaningless.

This problem has been overcome by expanding H(s) to a continued fraction expansion.

$$H(s) = a_1 s_1 + b_1 + \cfrac{1}{a_2 s + b_2 + \cfrac{1}{a_3 s + b_3 + \cdots}}$$

This expansion is done symbolically after the value of Z is substituted. Common roots conceal each other and do not appear in the result.

(2) In the Soriau-Framme part of the algorithm we often observed the appearance of very small and very large numbers which often got out of the dynamic range of the computer. Some insight to this phenomena can be gained from the following calculation.

Consider the Soriau Framme formulas: Using the mean-square vector norm we get:

$$|g_K| = |-\tfrac{1}{K}\,\mathrm{trace}(T_{K-1}A)| = |-\tfrac{1}{K}(\sum_i \bar{\lambda}_i)| \leq \max_i |\bar{\lambda}_i| = \|T_{n-1}A\|$$

$$\leq \|T_{n-1}\|\,\|A\|$$

where $\overline{\lambda}_i$ are the eigen values of $T_{K-1}A$.

$$\|T_K\| \leq \|T_{K-1}\|\|A\| + |g_n| \leq 2\|T_{K-1}\|\|A\|$$

$$\|T_K\| \leq 2^K\|A\|^K$$

Since we know that T_n is zero this bound has meaning for $K < n$ only.

There are several ways for estimation of $\|A\|$ [6] one way is the following: for real A, $\|A\| = |\lambda_{max}|$, λ_{max} the largest (in absolute value) eigen value of A

$$\max_{ij}|a_{ij}| \leq |\lambda_{max}| \leq n \max_{ij}|a_{ij}|$$

Thus, large numbers appear because essentially A is multiplied several times by itself. (If $\|A\|$ is smaller than 1 the same phenomena will cause small numbers, but this is rare as the time constants of circuits we are using are usually much smaller than one second). Exceeding the dynamic range can be avoided by scaling, i.e. multiply A by a constant C such that $2^n C^n \|A\|^n$ will be inside the dynamic range. Physically this can be interpreted as measuring time not by seconds but by msecs, for example.

Very small numbers are automatically rounded off to zero and do not stop the program from running. However, obtaining small numbers as differences of large numbers is a serious cause of numerical errors. Accuracy can be improved by using a number representation which uses more bits to represent the number's mantisa. We run examples on both the Elliot (39 bits word) and the CDC (64 bits word, double precision). For a band pass Butterworth filter with 8 reactive elements we got 3% change difference in the value of the transfer function.

It is hard to estimate the effect of round off when the network becomes larger and larger. It seems, however, that these errors can be reduced drastically by calculating the Soriau-Framme formulas using a number representation which occupies several machine words. Although not elegant this is a simple technical solution. Since we don't know a priori the accuracy required, one solution is to write arithmetic subroutines for n-precision and rerun the problem with a n+1-precision if the sum with n-precision fails.

(3) Numerical inaccuracies were reported [2] also in the process of getting the state equations. The critical step should be the reduction of the equations from the form $S\dot{q} = A'q + B'u$ to $\dot{q} = Aq + Bu$, which is done by diagonalizing S.

In our examples we did not experience problems in this step. It is our opinion that accuracy can be easily increased by using multi-precision representation.

Summary

The partial fraction expansion, scaling and the crude but technically sound device of multipresicion makes the method presented into an engineering tool.

An interesting question is to compare this method with other methods as far as the amount of computer time is concerned. Unfortunately we don't have enough data to comment on this point.

REFERENCES

[1] Branin, F. H., Jr., "Computer Methods of Netowrk Analysis," in Kuo and Magnusen, Computer Oriented Circuit Design, Prentice Hall, Inc., 1969.

[2] Pottle, C., "State Space Techniques for General Active Network Analysis," in Kuo and Kaizer, System Analysis by Digital Computers, John Wiley and Sons, Inc., New York, 1969.

[3] Tsur, S., "Computer Aided Analysis of Linear Networks," M.Sc. Thesis, Dept. of Elect. Engr., Technion, Haifa, Israel, 1969.

[4] Framme, J. S., "Matrix Functions and Applications," Part IV, IEEE Spectrum, pp. 123-131, June 1964.

[5] So, H. C., "On the Hybrid Description of a Linear n-Port Resulting from the Extraction of ArbitrarySpecified Elements," Trans. IEEE on Circuit Theory, vol. CT-12, no. 3, pp. 381-387.

[6] Faddeeva, V. N., "Computational Methods in Linear Algebra," Dover Publications, New York, 1959.

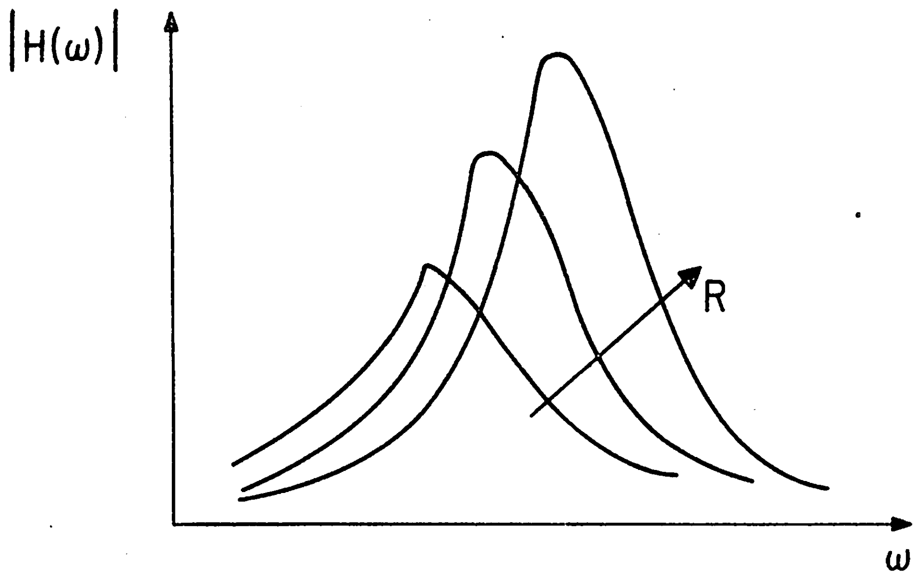

Fig. 1. An example of an absolute value of a transfer function as function ω with R a parameter.

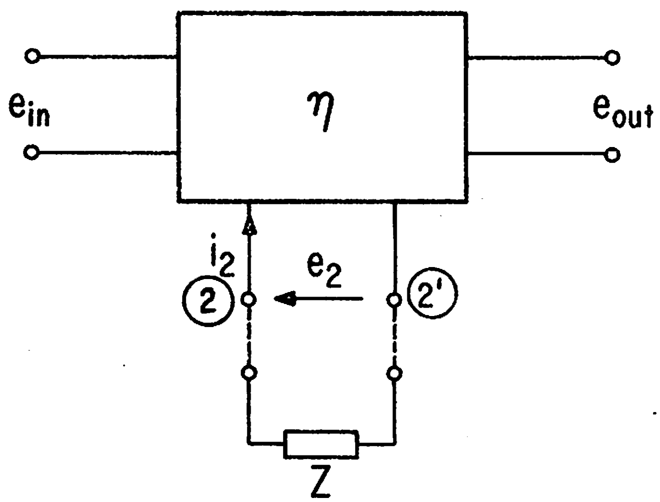

Fig. 2. The extraction of Z from the network.

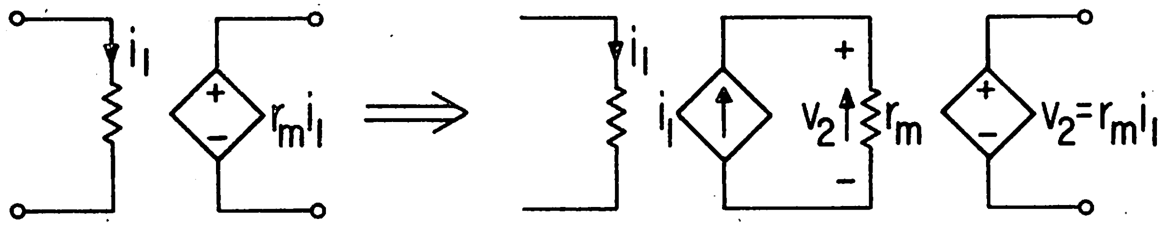

Fig. 3. The transformation of a current controlled voltage source to a configuration suitable for extracting r_m as a parameter.