Precise Algorithms for Pole-Zero Analysis in Electronic Circuit Design

Josef Dobeš*, Dalibor Biolek†, Jan Míchal*, David Černý*, and Libor Sláma*
*Czech Technical University in Prague, Department of Radio Engineering, Technická 2, 16627 Praha 6, Czech Republic
†Brno University of Technology, Department of Microelectronics, Údolní 53, 60200 Brno, Czech Republic

***Abstract*—The pole-zero analysis is generally known to be very sensitive to the numerical precision of the computer arithmetics. In the paper, various methods are suggested for solving that problem. First, an optimal pivoting strategy of the algorithm that reduces the general eigenvalue problem to the standard one is presented for both full- and sparse-matrix procedures. The algorithm increases the precision of the semisymbolic analysis, especially for the large-scale radio-frequency circuits. A novel technique is also incorporated recognizing multiple poles or zeros, which are often computed inaccurately by standard algorithms. A new type of this procedure called secondary root polishing is described in the paper. The accuracy is furthermore increased using longer numerical data. First, the long double precision is utilized. Further, a novel application of a suitable multiple-precision arithmetic library is suggested. Finally, using the longer numerical data to eliminate possible imprecision of the multiple eigenvalues is evaluated. The algorithm is demonstrated in both low- and high-frequency domains. In the low-frequency domain, necessity of using the longer numerical data is demonstrated by a power operational amplifier with poles and zeros located in both hertz and gigahertz ranges, which are often computed inaccurately by the standard algorithms. In the high-frequency domain, the algorithm is demonstrated by estimating the frequency of a distributed microwave oscillator, and by estimating the bandwidth of a distributed microwave amplifier.

***Keywords*—Eigenvalues and eigenfunctions, poles and zeros, resonant or large-scale electronic circuits, sparse matrices, variable-length arithmetic, cascade filters, transmission lines, power amplifiers, distributed microwave amplifiers, distributed microwave oscillators.

I. Introduction

The POLE-ZERO analysis belongs indisputably to the most important parts of the design of electronic circuits. However, the analysis is known to be very sensitive to the numerical precision of algebraic operations during the process [1]–[6]. Consequently, many of the theoretically exact methods fail, especially for the large-scale radio-frequency and microwave circuits.

Four major types of improvements to these methods are proposed here:

- the first one consists in a meticulous algorithm design regarding the choice of pivots during the reduction to the standard eigenvalue problem,
- the second one suggests the primary or secondary root polishing for a substantial improvement of the results of both reduction and QR algorithm,
- the third one is based on using longer numerical data types together with more precise arithmetic for the sparse-matrix reduction – either as fully utilizing the given hardware capabilities or by applying a suitable multiple-precision software arithmetic library,
- and the fourth one suggests the longer numerical data to be used for the QR algorithm to eliminate expected imprecision of the multiple eigenvalues (the imprecision of unequal eigenvalues with the same absolute value is resolved using the shift of algorithm's origin).

This paper has been supported by the Grant Agency of the Czech Republic, grant N° 102/08/0784.

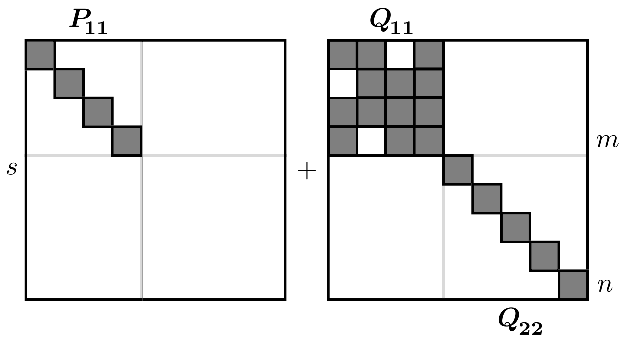

Fig. 1. Final position of the nonzero matrix elements after the reduction. The matrices $\boldsymbol{P_{11}}$ and $\boldsymbol{Q_{22}}$ are diagonal, the matrix $\boldsymbol{Q_{11}}$ is of arbitrary structure.

II. Principle of the Reduction Algorithm

A system of linear circuit equations (or equations that are linearized around an operating point) can be written by means of the Laplace transformation

$$(s\boldsymbol{P} + \boldsymbol{Q})\,\boldsymbol{X} = \boldsymbol{Y}, \tag{1}$$

where s marks the Laplace operator, $\boldsymbol{P}/\boldsymbol{Q}$ are the matrices associated with the dynamic/static parts of the model derivatives, respectively, $\boldsymbol{X}$ is the vector of the Laplace representation of circuit variables, and $\boldsymbol{Y}$ is the vector of external sources.

The poles of all the transfer functions and the zeros of a transfer function "j^{th} circuit variable divided by i^{th} external source" can be computed by solving the equations

$$\begin{aligned} \det\big(s\boldsymbol{P} + \boldsymbol{Q}\big) &= 0 \text{ for poles,} \\ \det\Big(s\boldsymbol{P}^{(0j)} + \boldsymbol{Q}^{(ij)}\Big) &= 0 \text{ for zeros,} \end{aligned} \tag{2}$$

where the matrices $\boldsymbol{P}^{(0j)}$ and $\boldsymbol{Q}^{(ij)}$ arise from the original ones – the first by zeroing j^{th} column, and the second by replacing j^{th} column by $\boldsymbol{Y}$ with all its elements zeroed with the exception of the one corresponding to i^{th} source.

Solving the general eigenvalue problems defined by (2) is more difficult than solving the standard ones. Therefore, a systematic reduction is applied during the transformation of (2) to the standard form, which is shown in Fig. 1 – it is an alternative of the method in [7]. After the transformation, the determinant can be computed in a classical way:

$$\begin{aligned}&\det\big(s\boldsymbol{P}+\boldsymbol{Q}\big) = \\ &(-1)^{n_{\text{exch}}}\prod_{i=m+1}^{n} Q_{22_{ii}} \det\Big(\boldsymbol{P_{11}}\boldsymbol{P_{11}^{-1}}\big(s\boldsymbol{P_{11}}+\boldsymbol{Q_{11}}\big)\Big) = \\ &(-1)^{n_{\text{exch}}}\prod_{i=1}^{m} P_{11_{ii}} \prod_{i=m+1}^{n} Q_{22_{ii}} \det\big(s\boldsymbol{1}+\boldsymbol{P_{11}^{-1}}\boldsymbol{Q_{11}}\big),\end{aligned} \tag{3}$$

where n_{exch} is the total number of row and column exchanges during the reduction, and $\boldsymbol{1}$ is the unity matrix. The operations that transform the matrix $s\boldsymbol{P}+\boldsymbol{Q}$ to the form shown in Fig. 1 are certain modifications of the Gauss elimination method. The only exception occurs when the matrix $\boldsymbol{P_{11}}$ contains a nondiagonal element that is not reducible by the diagonal elements of this matrix. In such cases, it is necessary to multiply a row from the lower part of the matrix by the s operator, which is equivalent to moving a row of the $\boldsymbol{Q_{22}}$ matrix left. The nondiagonal element of the $\boldsymbol{P_{11}}$ matrix can then be reduced by means of the transferred row. Note that the two products in the equation (3) could be extremely big for the large-scale RF circuits and, therefore, only their signs and logarithms may be stored in the computer memory.

Note that for a general efficiency of the reduction algorithm (3), utilizing matrix sparsities is necessary. Various methods of exploiting the sparsity of matrices are described in [7]–[9].

The final step for determining the poles or zeros of the transfer function is naturally the computation of the eigenvalues of the matrix (i.e., solving the standard eigenvalue problem, for which the QR algorithm with a shift of origin is mostly used)

$$\boldsymbol{Q'} = -\boldsymbol{P_{11}^{-1}}\boldsymbol{Q_{11}}. \tag{4}$$

III. Enhancing the Accuracy of the Reduction

A. Optimal Pivoting for the Reduction Algorithms

The reduction process is very difficult from the point of view of numerical precision, especially in the case of the large-scale systems. The matrices often contain elements of very different magnitudes. Therefore, a *full pivoting* should be used for the choice of the kth key element:

$$P_{kk} := \max_{\substack{k\leqslant i\leqslant n\\ k\leqslant j\leqslant n}} |P_{ij}|,\ k=1,\ldots,n. \tag{5}$$

However, the key element determined in the above stated way is regarded to be zero if it is too small in comparison with the maximum element of the kth column:

$$\text{if } |P_{kk}| \leqslant \varepsilon_{\text{eigen}} \max_{1\leqslant i<k} |P_{ik}|, \text{ then } P_{kk} := 0. \tag{6}$$

$\varepsilon_{\text{eigen}}$ is an important parameter of the algorithm. An inappropriately large value of the parameter causes ignoring some (real, in fact) poles or zeros, inappropriately small value causes computing superfluous (spurious) poles or zeros.

The key elements determined in the upper and lower parts of the matrices are used for the reduction of remaining elements of the matrices if they are not too small:

$$\begin{aligned}&\text{if } |P_{i'j'}| \leqslant \varepsilon_{\text{round}} \max_{\substack{k\leqslant i\leqslant n\\ k\leqslant j\leqslant n}} |P_{ij}|, \text{ then } P_{i'j'} := 0,\\ &\qquad k=1,\ldots,n-1,\ i'\geqslant k \wedge j'\geqslant k,\end{aligned} \tag{7a}$$

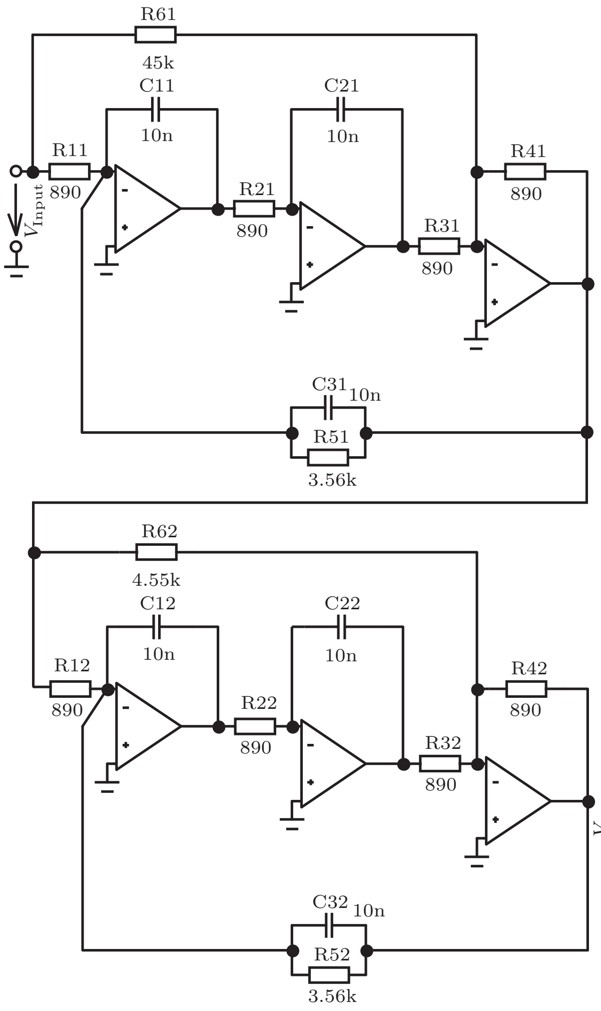

Fig. 2. Cascade of two 2nd-order building blocks. The application of secondary root polishing is necessary for precise analysis.

$$\begin{aligned}&\text{if } |Q_{i'j'}| \leqslant \varepsilon_{\text{round}} \max_{\substack{m<i\leqslant k\\ m<j\leqslant k}} |Q_{ij}|, \text{ then } Q_{i'j'} := 0,\\ &\qquad k=n,\ldots,m+1,\ m<i'\leqslant k \wedge m<j'\leqslant k.\end{aligned} \tag{7b}$$

$\varepsilon_{\text{round}}$ is a second parameter of the algorithm. It prevents the reduction of tiny elements that can arise due to roundoff errors.

The reduction algorithm should be implemented to use the sparsity of matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$. These matrices are sparse enough for not too complicated tasks. However, the application of the full pivoting is then difficult from the programming point of view. Therefore, only *partial pivoting* must be used here – the kth key element is chosen from the rest of the kth column of a reduced matrix:

$$P_{kk} := \max_{k\leqslant i\leqslant n} |P_{ik}|,\ k=1,\ldots,n. \tag{8}$$

However, the key element determined in the above stated way is regarded to be zero if it is too small in comparison with the maximum element of the kth row:

$$\text{if } |P_{kk}| \leqslant \varepsilon \max_{k<j\leqslant n} |P_{kj}|, \text{ then } P_{kk} := 0, \tag{9}$$

where ε is a parallel to $\varepsilon_{\text{eigen}}$.

TABLE I
COMPARISON OF THE POLE-ZERO ANALYSES OF THE CASCADE FILTER

Without Secondary Root Polishing (rads/sec)	
Zeros	$\pm 7.98953027764\times10^5 j,\ \pm 2.54050871091\times10^5 j$
Poles	$-5.61813401118\times10^4 - 2.94208471184\times10^{-25} j$
	$-5.61797752547\times10^4 + 1.62558164422 j$
	$-5.61797752525\times10^4 - 1.56365534119 j$
	-5.61780942729×10^4

With Secondary Root Polishing (rads/sec)	
Zeros	$\pm 7.98953027764\times10^5 j,\ \pm 2.54050871091\times10^5 j$
Poles	Quadruple -5.61797462230×10^4

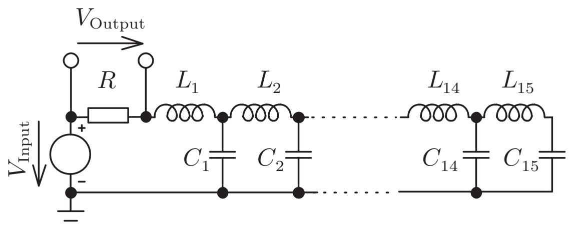

Fig. 3. Modeling a circuit with the transmission line which generates 10 ns impulse after applying voltage V_{input} across the input terminals. $R = 75\ \Omega$, $L_1 = L_2 = \cdots = L_{15} = 25$ nH, and $C_1 = C_2 = \cdots = C_{15} = 4.444$ pF.

B. Using the Primary and Secondary Root Polishing

Nevertheless, in some cases, the procedures (5)–(9) would not be accurate enough and, therefore, they can generate inadmissible errors. To decrease these errors, special techniques are often used [10], [11], which are the methods of an iterative root improvement. They sometimes enable the error to be decreased to the level which is given by the processor roundoff error. These methods are called root polishing [10]. With respect to the following text, let us call them *primary root polishing*.

However, the primary root polishing should be complemented by another step. Let us assume that instead of a correct couple of poles $[-1, -1]$ we get $[-1, -0.999999999999]$ after the primary polishing. This tiny error (10^{-10} %) causes an incorrect identification of two different poles instead of one twofold. For instance, the equation of the step response will be of quite a different structure than that in the correct case. The above example is a typical illustration of an error which can be eliminated by the so-called *secondary root polishing*.

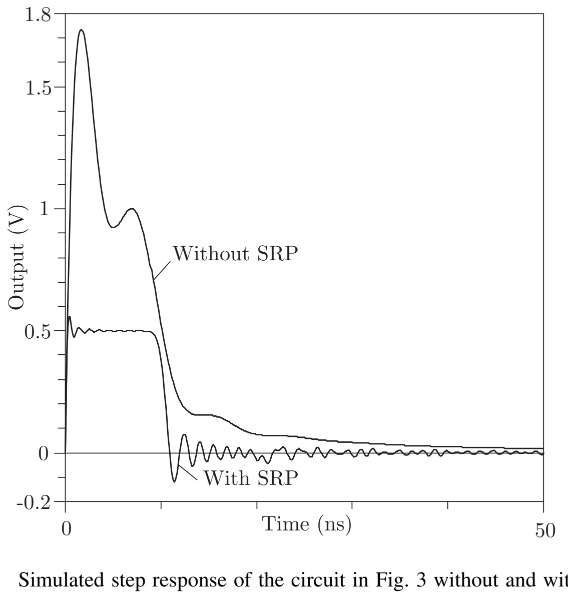

Fig. 4. Simulated step response of the circuit in Fig. 3 without and with the secondary root polishing (SRP).

Systematic procedural steps of the secondary root polishing are defined in a detailed way in [12], [13]. The system consists of the four procedures:

- *Zeroing the negligible components.* This procedure removes the numerical inaccuracy by zeroing poles or zeros with "too small" absolute values. A detailed algorithm for all the possible circumstances is defined in [12].
- *Identification of multiple roots.* This procedure is necessary for the partial fraction expansion of the s-domain circuit function, and for the subsequent inverse Laplace transform for finding equations of the transient and impulse responses. First, all roots are sorted into groups with "almost the same" values. Second, the "coincide test" is performed inside each group. If the roots are identified as identical, their real and imaginary parts are replaced by the mean values. The procedure is sensitive to the control parameter which must be neither too large nor too small. For a large value, two different roots can be identified as multiple. For a small value, multiple roots can be identified as different. This can be dangerous especially for the complex roots, when the algorithm needs not find their conjugate counterparts.
- *Polishing chains of zeros and nines.* This procedure tests the root's mantissa, whether a longer string of zeros is present on the right side of the decimal point. If it is found, all numbers to the right of this string are replaced by zeros. The longer the string of zeros the greater the probability that the subsequent nonzero numbers are due to rounding errors. Similarly, it is tested whether a longer string of nines is present on the right of the decimal point. If it is found, this string is replaced by zeros including all subsequent numbers, and a prospective order transfer is performed to the left-side. The longer the string of nines the greater the probability that it is due to rounding errors.
- *Zeros and poles cancellation.* If this "cancellation" is active, the procedure will try to cancel identical pairs of zeros and poles. The roots are considered to be identical, if both their real and imaginary parts are identical in the frame of the relative error given by the control parameter.

Let us demonstrate the procedures by two typical examples. First, consider a cascade of two 2nd-order building blocks in Fig. 2. This filter is designed with the aim to reach so-called FIR-BL approximation of frequency response with excellent group delay [14]. Considering ideal operational amplifiers and the equalities

$$C_{11} = C_{21} = C_{31} = C_{12} = C_{22} = C_{32} = C,$$
$$R_{11} = R_{21} = R_{31} = R_{41} = R_{21} = R_{22} = R_{32} = R_{42} = R,$$
$$R_{51} = R_{52} = 4R_{21} = 4R_{22}, \tag{10}$$

the quality factor of both blocks is 0.5 and the natural frequency is set to the value $\omega_0 = \frac{1}{2RC}$. The transfer function of each block is given by the equation

$$K_i(s) = a_i \frac{s^2 + \omega_{zi}^2}{(s + \omega_0)^2},\ i = 1, 2, \tag{11}$$

where

$$a_i = \frac{R_{4i}}{R_{6i}},\ \omega_{zi} = \frac{1}{RC}\sqrt{\frac{R_{6i}}{R}} = 2\omega_0\sqrt{\frac{R_{6i}}{R}},\ i = 1, 2. \tag{12}$$

That is why the transfer function $V_{\mathrm{Output}}/V_{\mathrm{Input}}$ of the filter in Fig. 2 has two double zeros

$$\begin{aligned} \pm j\omega_{z1} &= \pm j7.98953027764 \times 10^5 \text{ rads/sec},\\ \pm j\omega_{z2} &= \pm j2.54050871091 \times 10^5 \text{ rads/sec}, \end{aligned} \tag{13}$$

and one quadruple pole

$$-\omega_0 = \pm j5.61797752809 \times 10^4 \text{ rads/sec}. \tag{14}$$

The results of the analysis are shown in Table I. Without the secondary root polishing, only the zeros are found correctly. Instead of the real quadruple pole, four different poles were found. After enabling the root polishing, the algorithm found a correct quadruple pole. This example of filters with identical zeros or poles in a cascade is a typical representative of the circuits with multiple roots.

Further, as a demonstration of the efficiency of the secondary root polishing in the RF and microwave domains, consider a cascade of 15 LC cells in Fig. 3, modeling a transmission line for generating the 10 ns pulse after applying a constant voltage V_{Input} across the input terminals. The solution of this transient process by means of semisymbolic computations represents the following: finding 30 zeros and 30 poles of a $V_{\mathrm{Output}}/V_{\mathrm{Input}}$ transfer function, the partial fraction expansion (PFE) of this transfer function, and assigning the corresponding time-domain formulas. Numerically, the PFE of a ratio of polynomials represents an ill-posed problem [15]. The transfer function poles are the important input data of this procedure. Their small changes, especially in case of multiple roots, can make arbitrarily large changes in the resulting residues or can lead to absolutely different expansion formulas.

The analysis of the above circuit has been performed via the SNAP (Symbolic Network Analysis Program) [16]. The program finds 30 zeros and 30 poles within 15 digits of the mantissa. Without a secondary root polishing, eight complex conjugate couples of zeros and 10 complex conjugate couples of poles were not found with exactly identical real and imaginary parts. The resulting waveform in Fig. 4 is affected by a large error due to the failure of the PFE (see the waveform "Without SRP"). After enabling internal algorithms of the secondary root polishing, the poles are well polished and the PFE operates well (see the waveform "With SRP").

The last example shows that sometimes a small error in the pole-zero analysis can cause fatal errors, and that the root polishing can reliably prevent this failure.

C. Implementation of the Variable-Length Arithmetic

An additional improvement can be achieved using the `long double` precision (usually 10 (on PC) or 16 bytes) instead of the standard `double` (usually 8 bytes). However, only using the variable-length arithmetic can be considered the ultimate solution to the problem. A brief description of fundamental ideas of the variable-length arithmetic has been performed in [17]. In this subsection, a complete set of created procedures is defined, and its efficiency is demonstrated in next subsections.

The arbitrary precision (variable-length) arithmetic routines have been implemented in the Pascal language. Since the design was made with portability in mind, the ISO 7185 standard was strictly obeyed. In addition, only a subset of the language common with Borland Pascal/Delphi was used. Simplicity and clarity of the design encouraged by the chosen programming language are considered to be major virtues. The variable-length natural numbers representing the mantissa parts are implemented by means of dynamically allocated linked lists rather than arrays. This eliminates the danger of undesirable heap fragmentation, which could otherwise cause allocation failure before all available memory has been used. Only the classical general algorithms described in [18], [19] have been employed for the four basic arithmetic operations. Since the results of floating point operations are by principle approximate (no matter how long the mantissa has been chosen), an optional mechanism has been added maintaining upper estimations of the cumulated roundoff errors for each variable. This can be useful whenever the information about the guaranteed accuracy of obtained results is needed (note that alternative packages do not provide such accuracy estimations). No special optimization to enhance the execution speed has yet been performed. This is supposed to be part of the prospective next stage of development, finally resulting in partial or complete conversion of the routines into the assembly language, utilizing all technical capabilities of the given hardware.

The implementation of the variable-length floating point (`VLFloatingPoint`) arithmetic is part of a more extensive library of routines, covering a hierarchy of other variable-length numerical types: natural (or nonnegative integer) `VLNatural`, integer `VLInteger`, rational `VLRational` and complex `VLComplex`. All these types adopt the same philosophy of use: every variable of a particular type `X` (standing for any of `N`, `I`, `R`, or `F`) has first to be allocated by the `VLXNew` procedure. If it is to be used as an input to an arithmetic operation (`VLXAdd`, `VLXSub`, `VLXMul`, or `VLXDiv`), it needs to be initialized by `VLXInit`. By the end of a computation, all used variables should be deallocated by `VLXDispose`. Values can be converted from/into the standard real type by `VLXFromReal` and `VLXToReal`, and read from/written to text files by `VLXRead` and `VLXWrite`, respectively. For the programmers convenience, all arithmetic procedures are designed to allow for variables to be given simultaneously as input as well as output parameters.

Calling Pascal procedures from inside C code has been made possible using the GNU family of compilers, for GNU Pascal has types and calling conventions compatible with those of GNU C.

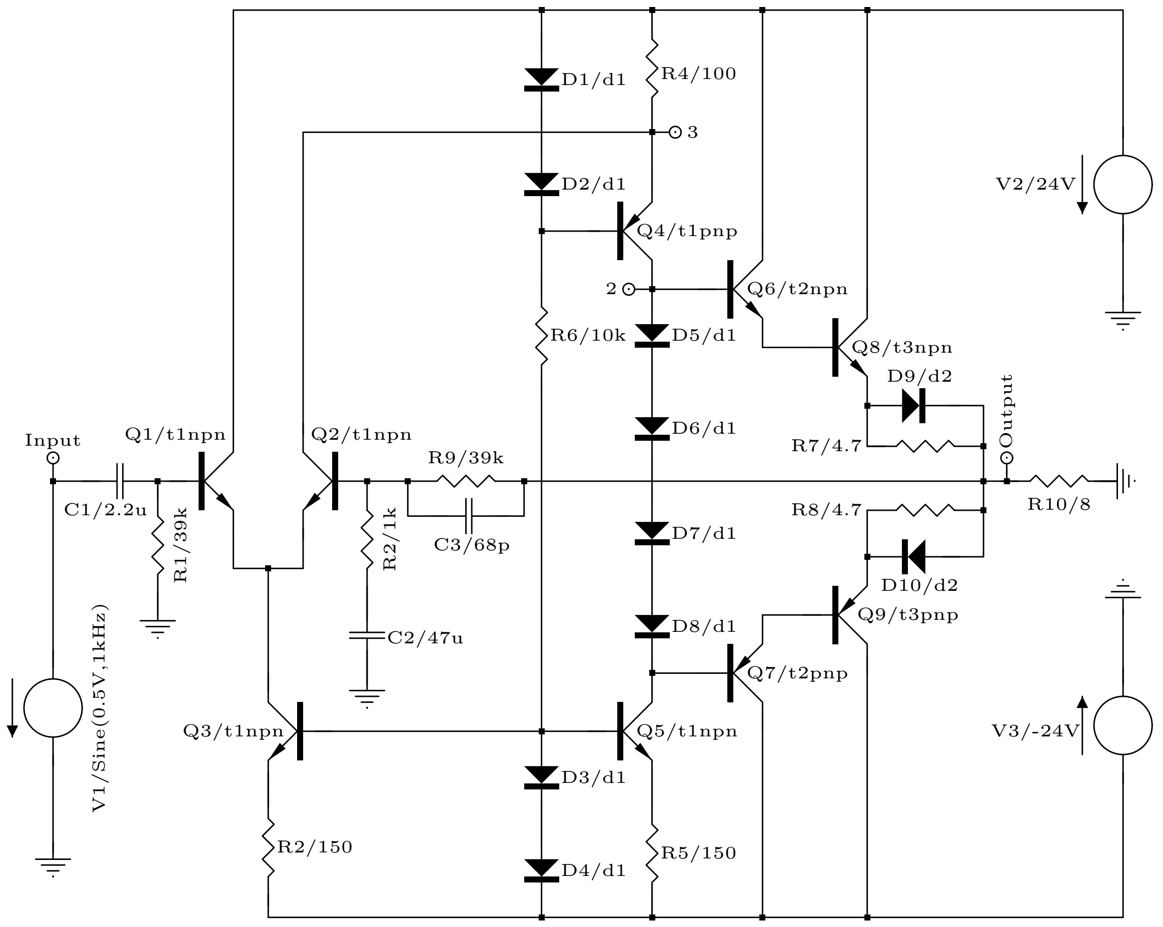

Fig. 5. AB-class power operational amplifier with wide spread poles and zeros (in both hertz and gigahertz bands) on which all the reduction techniques have been compared.

TABLE II
COMPARISON OF THE TRANSFER FUNCTION CRUCIAL ELEMENTS OBTAINED USING FIXED- (UPPER PART) AND VARIABLE-LENGTH (LOWER PART) COMPILER'S ARITHMETIC PROCEDURES

Crucial elements of transfer function regarding precision	**Full**-matrix algorithm, $\varepsilon_{\text{eigen}}=10^{-15}, \varepsilon_{\text{round}}=10^{-20}$	**Sparse**-matrix algorithm, **double**, $\varepsilon=10^{-19}$	**Sparse**-matrix algorithm, **long double**, $\varepsilon=10^{-23}$
Zero which should be 0 Hz	$-0.\underline{986035}\times10^{-6}$ Hz	0 Hz	0 Hz
Minimum pole by magnitude	–1.76518 Hz	–1.76518 Hz	–1.76518 Hz
Maximum pole by magnitude	-7.31331×10^{10} Hz	$-7.3133\underline{0}\times10^{10}$ Hz	-7.31331×10^{10} Hz
2nd minimum zero by magnitude	$-0.08478\underline{16}$ Hz	–0.0847825 Hz	–0.0847825 Hz
Maximum zero by magnitude	-6.99856×10^{10} Hz	$-\underline{7}.\underline{04891}\times10^{10}$ Hz	$-6.99\underline{916}\times10^{10}$ Hz
Constant of transfer function	0.988898	$\underline{1}.\underline{04496}$	$0.988\underline{968}$

The same observed elements	**64**bit mantissa, $\varepsilon=10^{-23}$	**128**bit mantissa, $\varepsilon=10^{-23}$	**256**bit mantissa, $\varepsilon=10^{-23}$
Zero which should be 0 Hz	0 Hz	0 Hz	0 Hz
Minimum pole by magnitude	–1.76518 Hz	–1.76518 Hz	–1.76518 Hz
Maximum pole by magnitude	-7.31331×10^{10} Hz	-7.31331×10^{10} Hz	-7.31331×10^{10} Hz
2nd minimum zero by magnitude	–0.0847825 Hz	–0.0847825 Hz	–0.0847825 Hz
Maximum zero by magnitude	$-6.99\underline{775}\times10^{10}$ Hz	-6.99856×10^{10} Hz	-6.99856×10^{10} Hz
Constant of transfer function	$0.988\underline{965}$	0.988898	0.988898

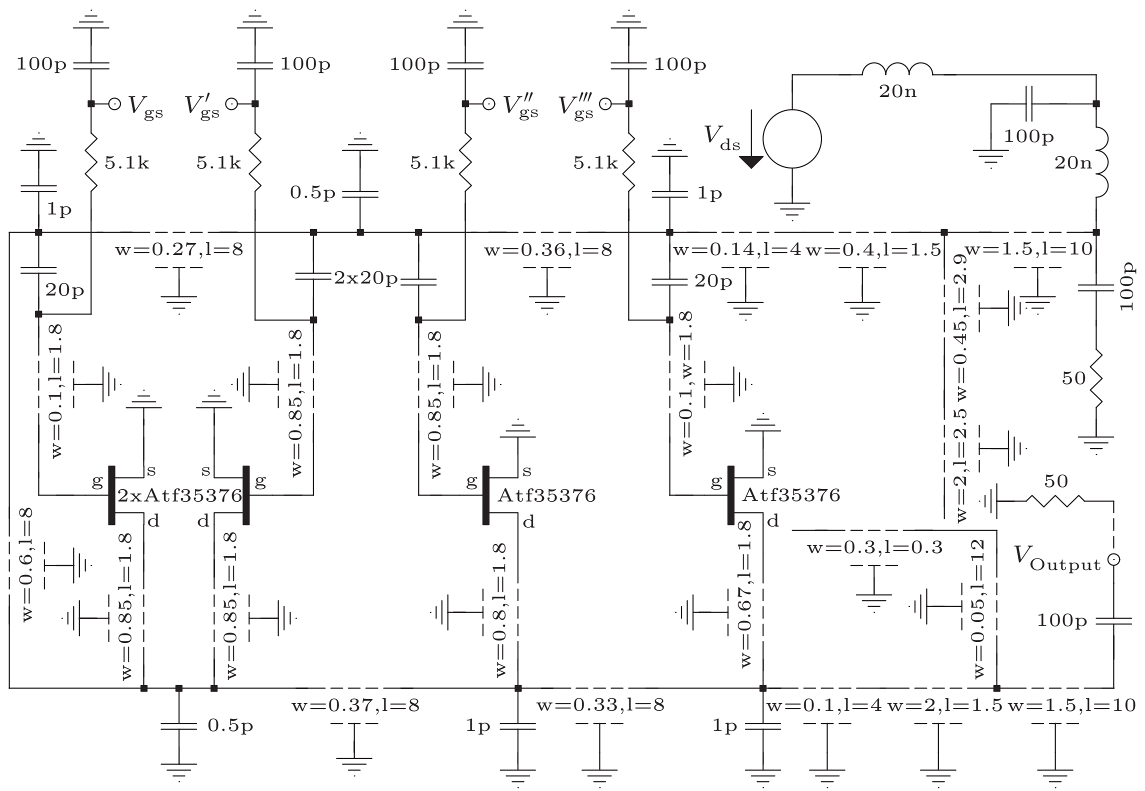

Fig. 6. Tunable distributed microwave oscillator with the frequency controlled by the gate-source voltages V_{gs}, V'_{gs}, V''_{gs}, and V'''_{gs}.

TABLE III
COMPARISON OF THE OSCILLATION FREQUENCIES OBTAINED WITH THE POLE-ZERO (PZ) AND STEADY-STATE (SS) ALGORITHMS

V_{gs} (V)	$V'_{gs}, V''_{gs}, V'''_{gs}$ (V)	$f_{osc}^{(PZ)}$ (GHz)	$f_{osc}^{(SS)}$ (GHz)	$\left(f_{osc}^{(PZ)} - f_{osc}^{(SS)}\right) / f_{osc}^{(SS)}$ (%)
-0.14	-1.4	3.1835	3.2308	-1.46
-0.15	-1.5	3.2582	3.367	-3.23
-0.16	-1.6	3.2922	3.1862	3.33
-0.17	-1.7	3.5040	3.276	6.96
-0.2	-2	3.3398	3.2916	1.46
-0.25	-2.5	3.1916	3.2733	-2.5

D. Enhancing the Accuracy of the QR Algorithm

As known, the QR algorithm may have problem with accuracy in case of multiple eigenvalues. However, implementing the longer numerical data can solve this problem. As a convenient demonstrating example, let us consider the matrix

$$\boldsymbol{Q}' = \begin{pmatrix} 3 & -3 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}. \qquad (15)$$

This matrix has a triplicate eigenvalue, which is (exactly) equal to 1. If we use the simplest classical Givens method in the QR algorithm, we obtain the inaccurate eigenvalues

$$\lambda_1 = 1.004,\ \lambda_2 = 0.999988,\ \lambda_3 = 0.996012. \qquad (16)$$

Utilizing the more complicated, but more efficient Householder method, we obtain the more accurate eigenvalues

$$\lambda'_1 = 0.999993,\ \lambda'_{2,3} = 1.000004 \pm 0.000006j. \qquad (17)$$

According to the above advisements, better results can be obtained using the same (Householder) method, but with the `long double` precision instead of (standard) `double` precision:

$$\lambda''_1 = 1.00000032,\ \lambda''_{2,3} = 0.99999984 \pm 0.00000027j. \qquad (18)$$

The best results can be obtained using the variable-length arithmetic. With the 256-bit mantissa, the results have more than 25 digits correct (Householder method was used again):

$$\begin{aligned} \lambda'''_1 &= 0.99999999999999999999999999518544, \\ \lambda'''_{2,3} &= 1.00000000000000000000000000024073 \\ &\qquad \pm 4.16953 \times 10^{-26} j, \end{aligned} \qquad (19)$$

which clearly demonstrates the absolute precedence of the variable-length arithmetic. Let us emphasize that even for using the 256-bit mantissa, the secondary root polishing (see the chains of zeros and nines in (19)) has also to be used.

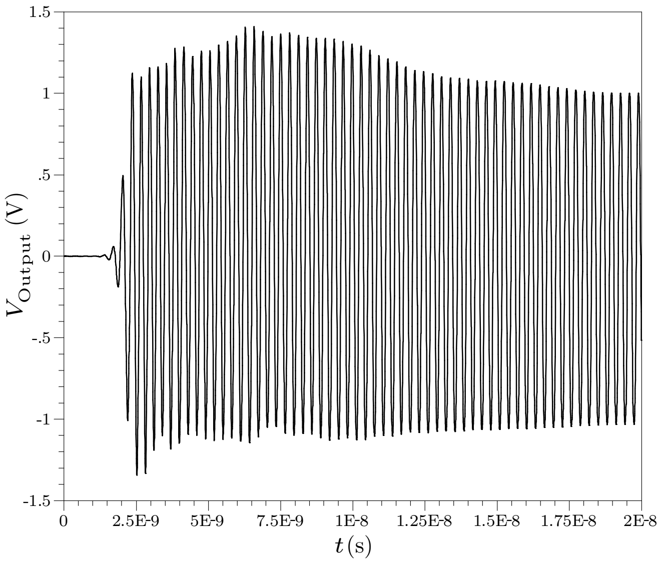

Fig. 7. Transient of the tunable distributed microwave oscillator (this one was given for $V_{\mathrm{gs}} = -0.3$ V, $V'_{\mathrm{gs}} = V''_{\mathrm{gs}} = V'''_{\mathrm{gs}} = -3$ V, and $V_{\mathrm{ds}} = 2.5$ V).

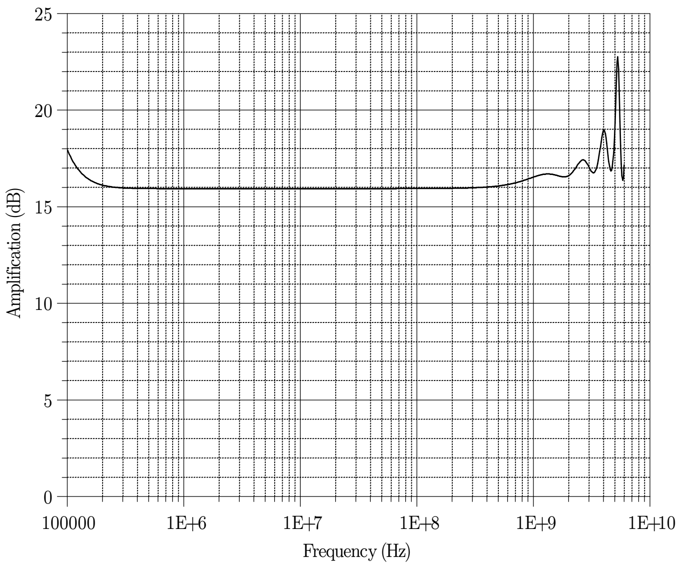

Fig. 8. Frequency response of the distributed microwave amplifier (the result of the AC analysis for the linearized circuit at the amplifier operating point).

IV. Solving More Complex Application Examples

A. Analyzing a Power Operational Amplifier With Poles and Zeros in Both Hertz and Gigahertz Bands

1) Improvement with the `long double` arithmetic: Let us discuss an AB-class power operational amplifier in Fig. 5. Let us also emphasize that although the amplifier itself is an LF device, it has *majority* of the poles and zeros of a $V_{\mathrm{Output}}/V_{\mathrm{Input}}$ transfer function located in the *gigahertz* band due to many parasitic capacitances of the diodes and bipolar junction transistors.

In general, the (inefficient) full-matrix version of the reduction algorithm seems to work sufficiently. However, the numerical precision is still inadequate for a certain class of tasks solved by the efficient sparse-matrix one. In Table II (upper part), a comparison between sparse-matrix `double` and sparse-matrix `long double` (10 bytes) reductions is performed (experiments with the `long double` precision were performed in [20], especially for digital filters analyzed using the $\mathcal{Z}$-transformation). Considering the results for 1024bit-mantissa variable-length arithmetic (see the following subsection) to be correct, the incorrect digits are underlined. As observed, the poles have been computed quite precisely. However, the precision of the computation of zeros is considerably worse – it is caused by the difference between the smallest and largest pole/zero magnitudes, respectively. For the multiplying constant of transfer function, the inaccuracy is similar.

2) Improvement with the variable-length arithmetic: The results obtained with the sparse-matrix reduction algorithm rewritten to call the variable-length arithmetic routines are presented in Tab. II (lower part). The parameter ε was chosen to be 10^{-23}, i.e., the same as in the `long double` computation, to allow a comparison of the corresponding results.

With mantissa length limited to 64 bits, which is the same length as in the `long double` type (`extended precision` of IEEE 754), the achieved precision of results is basically the same as with `long double`. A slight tendency towards the correct values is already visible in the variable-length case due to more correct rounding strategy used. All the poles and zeros obtained with 128-bit mantissa are already correct to 6 decimal digits. The file of results for the 128-bit mantissa length differs from the ones for longer mantissas (256, 512 and 1024 bits were tried) only in the ordering of zeros done by the procedure solving the standard eigenvalue problem. All the poles and zeros for 256 bits and more came out the same to 6 decimal digits and in the same order of listing in the file of results. The duration of the computations by 3 GHz PC did not exceed approximately 1 minute even for the 1024-bit mantissa length.

Changing the ε value was also tried for mantissas of 256 and 1024 bits. For 256 bits, e.g., it turned out to be possible to use its value as tiny as 10^{-150} without any change in the highest six digits and order in the results file. With 1024 bits, this limit even drops below 10^{-320}.

Experiments with the rational "unlimited-precision" arithmetic were also carried out; the computing complexity, however, turned out to be too high for the present example. The poles computation was interrupted after several hours, when only about a half of the approximate total of 200,000 arithmetic operations had been finished. Since the duration of multiplications and divisions increases with about the square of operand length, it is virtually impossible to estimate the time needed for the second hundred thousand arithmetic operations.

B. Estimating the Frequency of a Distributed Oscillator

Consider a distributed microwave oscillator in Fig. 6 [21]. Let us emphasize that the oscillator is tunable – therefore a number of analyses had to be carried out. As shown in Fig. 7, the transient of the oscillator is very complicated. Therefore, achieving the steady state had to be accelerated by the extrapolation algorithm. However, even the accelerated method needs more than 15 minutes (again, on the PC with

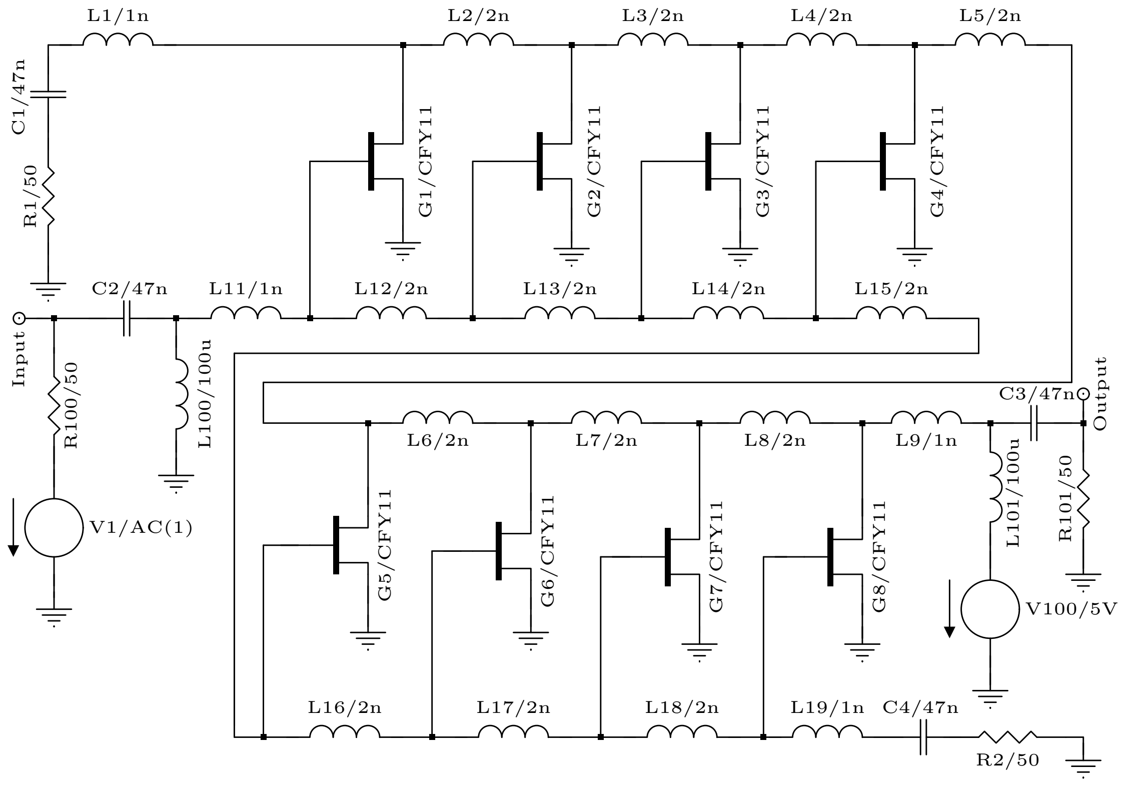

Fig. 9. Distributed microwave amplifier ("active transmission line") with constant amplification $V_{\text{Output}}/V_{\text{Input}}$ over several decades.

3 GHz Pentium 4) to determine a period of the steady state – it is caused by *hundreds* of internal nodes, which are necessary to model the microstrip lines accurately. For this reason, a fast estimation of the frequency of the oscillations using poles should be very useful.

The results are summarized in Table III (for $V_{\text{ds}} = 2.5$ V) – the pole-zero (PZ) estimation is based on the imaginary part of the smallest pole with a positive real part, the accurate steady-state (SS) value is based on the ϵ-algorithm results. The results show that the error of the pole-zero estimation is mostly of the percentage order.

C. Estimating the Bandwidth of a Distributed Amplifier

Consider a distributed microwave amplifier in Fig. 9 [22]. The frequency response of the amplifier is shown in Fig. 8. The 3-dB band begins approximately at 100 kHz, and ends approximately at 5 GHz, i.e., the amplification computed as $20 \log |V_{\text{Output}}/V_{\text{Input}}|$ is constant over several decades.

In this circuit, only eight GaAsFETs have been used. However, to cover higher frequencies, the number of circuit elements could be big and the AC analysis may be too time-consuming. For this reason, a fast estimation of the bandwidth using the poles-zeros analysis is useful. Here, only the two zeros are located in the interval $\langle 2\pi \times 10^5,\ 2\pi \times 10^9 \rangle$:

$$\begin{aligned} z_1 &= -2\pi \times 1.161386 \times 10^5 \text{ rads/sec}, \\ z_2 &= -2\pi \times 9.577553 \times 10^8 \text{ rads/sec}. \end{aligned} \tag{20}$$

Remaining poles and zeros are located outside this interval. As the two zeros (20) are very near the border of the interval, the amplification must be approximately constant inside it.

V. Conclusions

Several methods substantially enhancing the precision of the pole-zero analysis have been suggested including sophisticated pivoting strategy, primary and secondary root polishing, and utilizing the variable-length arithmetic for both sparse-matrix reduction and the QR algorithm. A combination of these procedures seems to be the possible solution of the frequent and serious problems with the accuracy of the pole-zero analysis. The algorithms are demonstrated by several typical circuits, and in an unusual way by estimating a distributed oscillator frequency and a distributed amplifier bandwidth.

References

[1] R. Lehoucq and D. Sorensen, "Deflation techniques for an implicitly re-started Arnoldi iteration," *SIAM J. Matrix Analysis and Applications*, vol. 17, pp. 789–821, 1996.

[2] D. Sorensen, "Implicit application of polynomial filters in a k-step Arnoldi method," *SIAM J. Matrix Analysis and Applications*, vol. 13, pp. 357–385, 1992.

[3] R. Lehoucq, D. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. Philadelphia: SIAM Publications, 1998.

[4] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK User's Guide*, 3rd ed. Philadelphia: SIAM Publications, 1999.

[5] Z. Kolka, M. Horák, D. Biolek, and V. Biolková, "Accurate semisymbolic analysis of circuits with multiple roots," in *13th WSEAS International Conference on Circuits*, Ródos, Greece, 2009, pp. 178–181.

[6] Z. Kolka, D. Biolek, and V. Biolková, "Accurate time-domain semisymbolic analysis," in *Proc. XIth International Workshop on Symbolic and Numerical Methods, Modeling and Applications to Circuit Design (SM2ACD)*, Gammarth, Tunisia, Oct. 2010.

[7] T. Rübner-Petersen, “On sparse matrix reduction for computing the poles and zeros of linear systems,” in *Int. Symp. on Network Theory*, Ljubljana, Slovenia, 1979.

[8] Z. Kolka, V. Biolková, and D. Biolek, “Exploiting matrix sparsity for symbolic analysis,” *WSEAS Transactions on Circuits and Systems*, vol. 3, no. 10, pp. 2278–2280, 2004.

[9] J. Dobeš, “C.I.A.—a comprehensive CAD tool for analog, RF, and microwave IC’s,” in *Proc. 8th IEEE Int. Symp. High Performance Electron Devices for Microwave and Optoelectronic Applications*, Glasgow, Nov. 2000, pp. 212–217.

[10] W. H. Press et al., *Numerical Recipes in Pascal. The Art of Scientific Computing*. Cambridge (UK): Cambridge University Press, 1994.

[11] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flanery, *Numerical Recipes*, 3rd ed. Cambridge University Press, 2007.

[12] D. Biolek and V. Biolková, “Secondary root polishing: Increasing the accuracy of semisymbolic analysis of electronic circuits,” in *Proc. WSEAS and IASME Conference on Applied Mathematics*, Corfu, Greece, Aug. 2004.

[13] ——, “Secondary root polishing: Increasing the accuracy of semisymbolic analysis of electronic circuits,” *WSEAS Transactions on Mathematics*, vol. 3, no. 3, pp. 493–497, 2004.

[14] ——, “FIR-BL analog filters,” in *IEEE Midwest Symposium on Circuits and Systems*, Las Cruses, New Mexico, 1999, pp. 1021–1023.

[15] F. Y. Chin and K. Steiglitz, “An O(N’) algorithm for partial fraction expansion,” *IEEE Transactions on Circuits and Systems*, vol. 24, no. 1, pp. 42–45, January 1977.

[16] D. Biolek, “SNAP – program with symbolic core for educational purposes,” in *Proc. WSEAS Multi-Conference on Circuits, Systems, Communications and Computers CSCC’2000*, Vouliagmeni, Athens, 2000, pp. 1711–1714.

[17] J. Dobeš and J. Míchal, “Accurate sparse-matrix semisymbolic analysis of large-scale RF circuits,” in *European Microwave Conference*, Manchester, UK, Sept. 2006.

[18] D. Knuth, *Seminumerical Algorithms*, 3rd ed. Reading, MA: Addison-Wesley, 1997, vol. 2.

[19] ——, *The Art of Computer Programming (Basic Algorithms)* (Czech translation). Reading, MA: Addison-Wesley, 2008, vol. 1.

[20] J. Dobeš, “An accurate poles-zeros analysis for large-scale analog and digital circuits,” in *IEEE Int. Conf. on Electronics, Circuits, and Systems*, St. Julians, Malta, Sept. 2001, pp. 1027–1030.

[21] L. Divina and Z. Skvor, “The distributed oscillator at 4 GHz,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 46, no. 12, pp. 2240–2243, Dec. 1998.

[22] T. T. Y. Wong, *Fundamentals of Distributed Amplification*. Boston: Artech House, 1993.