An Accurate Sparse-Matrix Semisymbolic Algorithm for Analyzing Distributed Microwave Circuits

Josef Dobeš, Dalibor Biolek*, and Jan Míchal
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
Email: {dobes, michal}@feld.cvut.cz, biolek@feec.vutbr.cz

***Abstract*— An optimal pivoting strategy for the reduction algorithm transforming 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 microwave circuits. The accuracy is furthermore increased using longer numerical data. First, the long double precision is utilized. Further, the 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 by estimating the frequency of a distributed microwave oscillator, and by estimating the bandwidth of a distributed microwave amplifier.**

I. Introduction

The poles-zeros analysis is indisputably among 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. Consequently, many of the theoretically exact methods fail, especially for the large-scale microwave circuits.

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

- the first one consists in a meticulous algorithm design regarding the choice of pivots,
- the second 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 third one suggests the longer numerical data to be used for the QR algorithm to eliminate expected imprecision in the case of multiple eigenvalues.

II. Principle of the Reduction Algorithm

A system of linear equations (or equations that are linearized at an operating point) of a circuit 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 transforms of circuit variables, and $\boldsymbol{Y}$ is the source vector.

This paper has been supported by the Grant Agency of the Czech Republic, grant N° 102/05/0277, and by the Czech Technical University research project MSM6840770014.

The poles of all the transfer functions and the zeros of a transfer function 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 one by clearing j^{th} column, and the second one by replacing j^{th} column by $\boldsymbol{Y}$ with all its elements cleared 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 [1]. 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 a certain modification 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.

A final step for determining the poles or zeros of the transfer function is a natural computing 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)

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

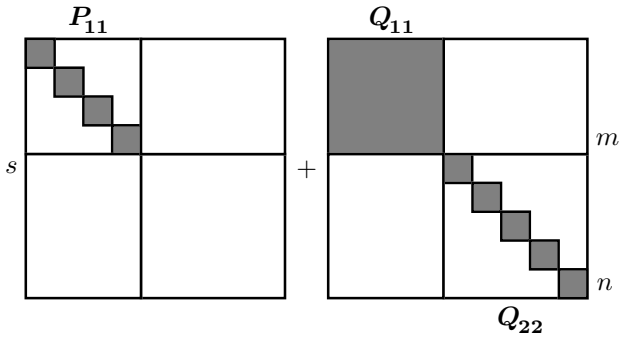

Fig. 1. Position of the nonzero elements after the reduction.

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 k^{th} 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 greatest element of the k^{th} column:

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

ϵ_{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, in fact) 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:

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

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

ϵ_{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 the matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$. These matrices are sparse enough for not very complicated tasks. However, the application of the full pivoting is then very difficult from the programming point of view. Therefore, only *partial pivoting* must be used here – the k^{th} key element is chosen from the rest of the k^{th} column of a reduced matrix:

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

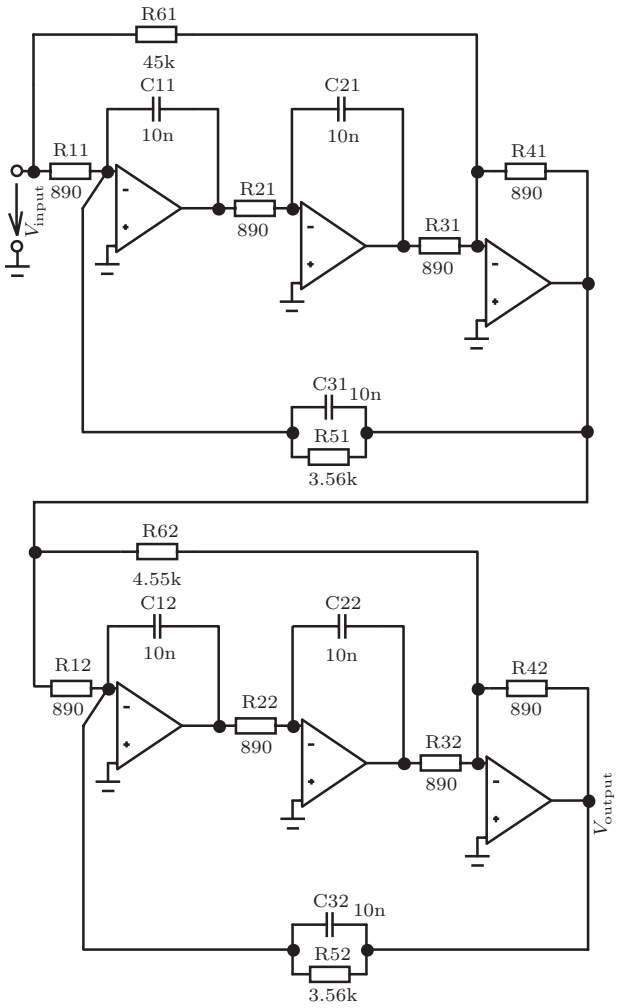

Fig. 2. Cascade of two 2nd-order building blocks analyzed by root polishing.

however, the key element determined in the above stated way is regarded to be zero if it is too small in comparison with the greatest element of the k^{th} row:

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

where ϵ is a parallel to ϵ_{eigen}.

B. Using the Primary and Secondary Root Polishing

Nevertheless, in some cases of the analysis, 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 [2], which are methods of an iterative root improvement. They sometimes enables the error to be decreased to the level which is given by the processor roundoff error. These methods are called *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

TABLE I

COMPARISON OF THE POLES-ZEROS ANALYSES OF THE CASCADE FILTER

Without Secondary Root Polishing (rads/sec)	
Zeros	$\pm 7.98953027764\times 10^5 j,\ \pm 2.54050871091\times 10^5 j$
Poles	$-5.61813401118\times 10^4 - 2.94208471184\times 10^{-25} j$
	$-5.61797752547\times 10^4 + 1.62558164422 j$
	$-5.61797752525\times 10^4 - 1.56365534119 j$
	$-5.61780942729\times 10^4$
With Secondary Root Polishing (rads/sec)	
Zeros	$\pm 7.98953027764\times 10^5 j,\ \pm 2.54050871091\times 10^5 j$
Poles	Quadruple $-5.61797462230\times 10^4$

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*.

C. Implementation of the Variable-Length Arithmetic

An additional improvement can be achieved using the `long double` precision (usually 10 bytes) instead of the standard `double` (usually 8 bytes). However, only using the variable-length arithmetic can be considered a solution of the problem.

The variable-length natural numbers representing the mantissa parts are implemented by means of dynamically allocated linked lists rather than arrays in our software. This eliminates the danger of undesirable heap fragmentation, which could otherwise cause the allocation failure before all available memory has been used. Only the classical general algorithms described in [3] have been employed for the four basic arithmetic operations.

Since the results of the floating point operations are by principle approximate (no matter how long has the mantissa been chosen), an optional mechanism has been added maintaining upper estimates of the cumulated roundoff errors for each variable. This can be useful whenever the information about the *guaranteed* accuracy of obtained results is needed – similar packages do not provide the accuracy estimation (GNU, e.g.).

IV. ENHANCING THE ACCURACY OF THE QR ALGORITHM

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

$$\boldsymbol{Q'} = \begin{pmatrix} 3 & -3 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}. \tag{10}$$

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

$$\lambda_1 = 1.004,\ \lambda_2 = 0.999988,\ \lambda_3 = 0.996012. \tag{11}$$

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

$$\lambda'_1 = 0.999993,\ \lambda'_{2,3} = 1.000004 \pm 0.000006 j. \tag{12}$$

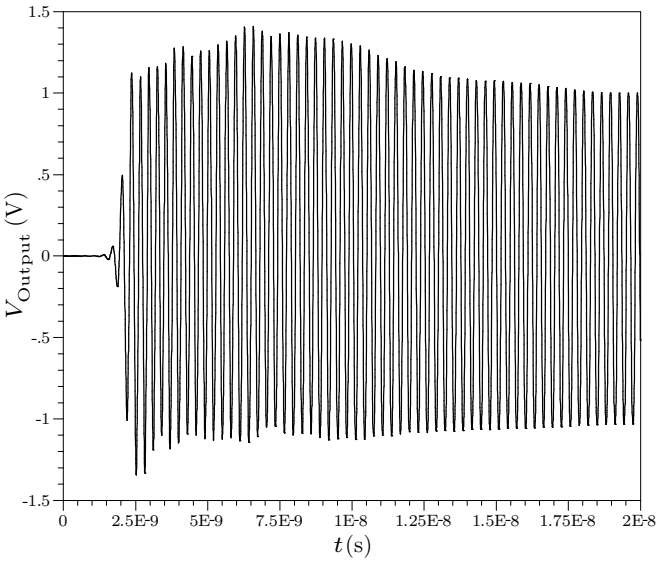

Fig. 4. Transient of the tunable distributed microwave oscillator (this one was given for $V_{\mathrm{gs}} = -0.3$ V, $V'_{\mathrm{gs}} = V''_{\mathrm{gs}} = V'''_{\mathrm{gs}} = -3$ V, and $V_{\mathrm{ds}} = 2.5$ V).

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` one:

$$\lambda''_1 = 1.00000032,\ \lambda''_{2,3} = 0.99999984 \pm 0.00000027 j. \tag{13}$$

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

$$\begin{aligned} \lambda'''_1 &= 0.99999999999999999999999999518544, \\ \lambda'''_{2,3} &= 1.00000000000000000000000024073 \\ &\quad \pm 4.16953 \times 10^{-26} j, \end{aligned} \tag{14}$$

which clearly demonstrates the absolute precedence of the variable-length arithmetic.

V. SOLVING APPLICATION EXAMPLES

A. Demonstrating the Secondary Root Polishing

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 good group delay. Consider ideal operational amplifiers and the equalities

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

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

$$K_i(s) = a_i \frac{s^2 + \omega_{zi}^2}{(s+\omega_0)^2},\ i = 1, 2, \tag{15}$$

where

$$a_i = \frac{R_{4i}}{R_{6i}},\ \omega_{zi} = \frac{1}{RC}\sqrt{\frac{R_{6i}}{R}} = 2\omega_0\sqrt{\frac{R_{6i}}{R}},\ i = 1, 2. \tag{16}$$

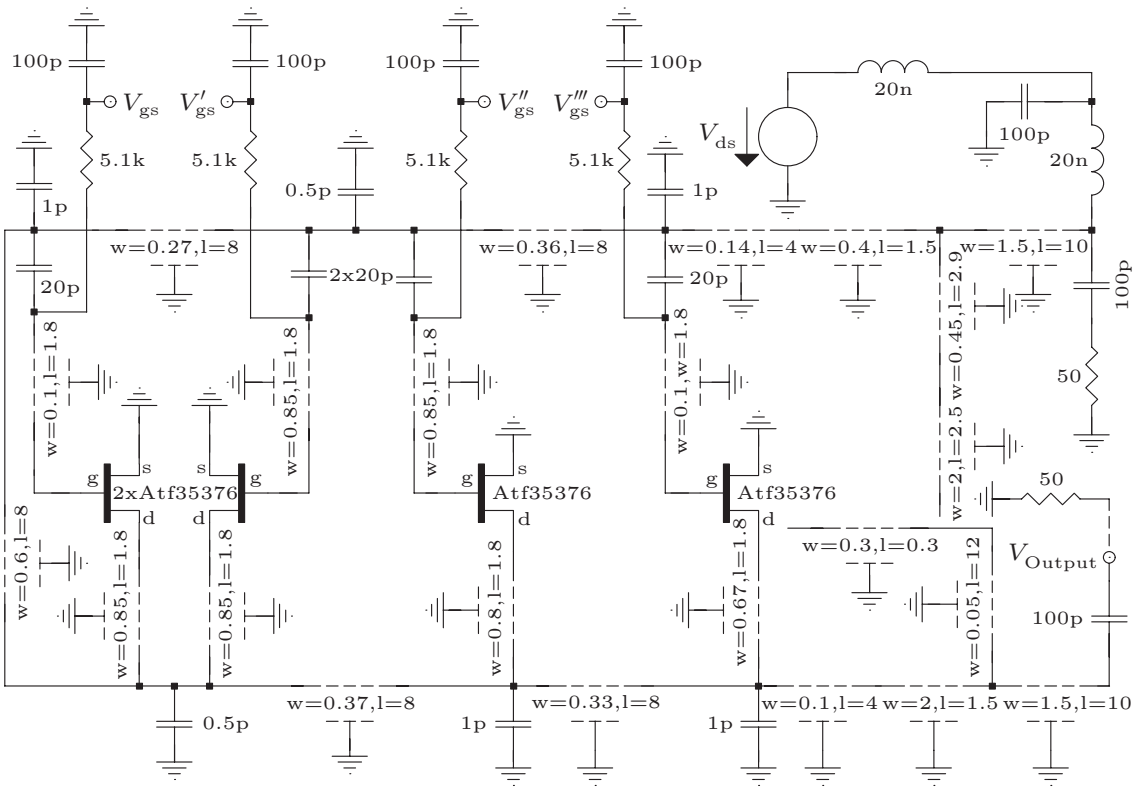

Fig. 3. Tunable distributed microwave oscillator.

TABLE II

COMPARISON OF THE OSCILLATION FREQUENCIES OBTAINED WITH THE POLES-ZEROS (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

That is why the transfer function $V_{\rm output}/V_{\rm input}$ of the filter in Fig. 2 has two double zeros

$$\begin{aligned} \pm j\omega_{z1} &= \pm j7.98953027764 \times 10^5 \text{ rads/sec}, \\ \pm j\omega_{z2} &= \pm j2.54050871091 \times 10^5 \text{ rads/sec}, \end{aligned} \tag{17}$$

and one quadruple pole

$$-\omega_0 = \pm j5.61797752809 \times 10^4 \text{ rads/sec}. \tag{18}$$

Results of the analysis are shown in Table I. Without secondary root polishing, only zeros are found correctly. Instead of the real quadruple pole, four different poles found. After enabling the root polishing, the algorithm found correct quadruple pole.

B. Estimating the Frequency of a Distributed Oscillator

Consider a distributed microwave oscillator in Fig. 3 [4]. Emphasize that the oscillator is tunable – therefore a number of analyses had to be carried out. As shown in Fig. 4, the transient of the oscillator is very complicated. Therefore, determining the steady-state had to be accelerated by the extrapolation algorithm. However, even the accelerated method needs more than 15 minutes (on PC with 3 GHz Pentium 4) to determine a one 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 II (for $V_{\rm ds} = 2.5$ V) – the PZ estimate is based on the imaginary part of the smallest pole with a positive real part, the accurate steady-state value is based on the ϵ-algorithm results. The results show that the error of the PZ estimation is mostly of the percentage order.

C. Estimating the Bandwidth of a Distributed Amplifier

Consider a distributed microwave amplifier in Fig. 5 [5]. The frequency response of the amplifier is shown in Fig. 6.

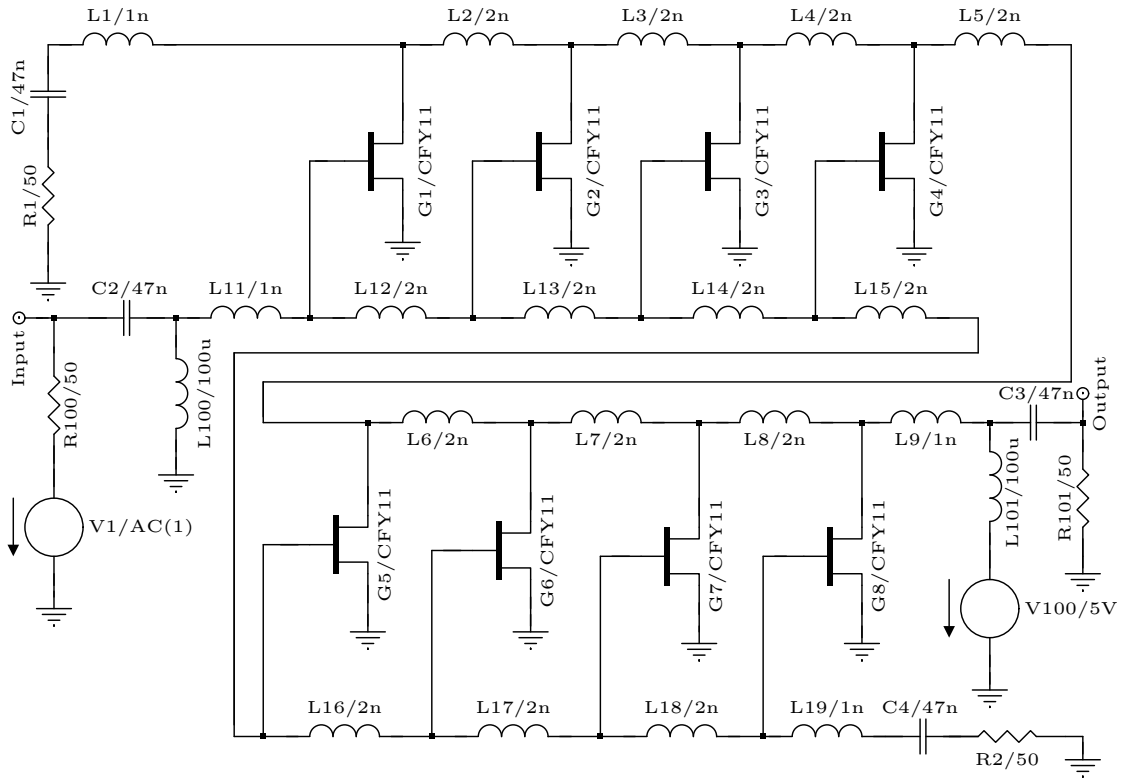

Fig. 5. Distributed microwave amplifier.

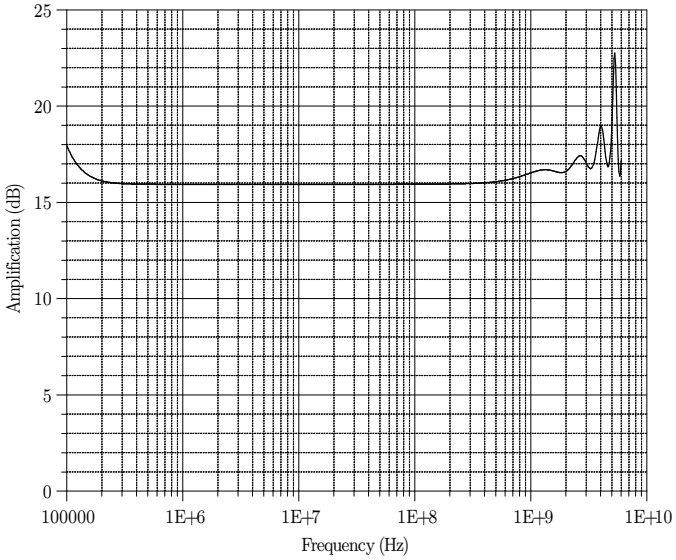

Fig. 6. Frequency response of the distributed microwave amplifier (the result of the AC analysis for the linearized circuit at the amplifier operating point).

The 3 dB-band begins approximately at 100 kHz, and ends approximately at 5 GHz, i.e., the amplification computed as $20\log|V_{\rm Output}/V_{\rm Input}|$ is constant over several decades.

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

$$\begin{aligned} z_1 &= -2\pi\times 1.161386\times 10^5 \text{ rads/sec},\\ z_2 &= -2\pi\times 9.577553\times 10^8 \text{ rads/sec}. \end{aligned} \tag{19}$$

Remaining poles and zeros are located outside this interval. As the two zeros (19) are very near the border of the interval, the amplification must be approximately constant inside it.

VI. Conclusion

Several new methods enhancing the accuracy of the sparse-matrix reduction are suggested. As expected, the best results are enabled with the implementation of the variable-length arithmetic to both reduction and QR algorithms. The procedure is demonstrated in an unusual way by estimating a distributed oscillator frequency and a distributed amplifier bandwidth.

References

[1] 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.

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

[3] D. Knuth, *Seminumerical algorithms*. Reading (MA): Addison-Wesley, 1997, vol. 2.

[4] 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.

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