An Accurate and Efficient Computation of Zeros and Poles of Transfer Function for Large Scale Circuits

Josef Dobeš, Jan Míchal, František Vejražka, and Viera Biolková

***Abstract*—The zeros and poles of a circuit transfer function are computed solving a general eigenvalue problem, which could be transformed to a standard eigenvalue task to be solved by a suitably modified QR algorithm. Both reduction of the general eigenvalue problem to the standard form and the iterative procedures of the QR algorithm are very sensitive to the numerical precision of all calculations. The numerical accuracy is especially critical for two kinds of circuits: the microwave ones characterized by huge differences among magnitudes of the zeros and poles, and the large scale circuits, where the errors of the zeros and poles are increased by an extreme number of arithmetic operations. In the paper, an illustrative example of the reduction of the general eigenvalue problem and using the QR algorithm is shown first. After that, three circuits of various sizes are analyzed: simpler microwave low noise amplifier, larger power operational amplifier, and the most complex example with a 272 integrated operational amplifier. A meticulous comparison of obtained results shows that a usage of newly implemented 128-bit arithmetics in GNU Fortran or C compilers with partial pivoting can assure both efficient and enough accurate procedures for computing the zeros and poles.**

***Index Terms*—large scale circuits, poles and zeros, general eigenvalue problem, numerical precision, 128-bit arithmetics.**

I. Introduction

COMPUTING the poles and zeros of a circuit transfer function for sparse systems was principally described in [1]; however, the serious accuracy problem for the general eigenvalue problem remains for decades. Microwave circuits represent one typical problematic area [2], [3] due to huge differences in absolute values of their poles or zeros. Another typical problem consists in bad accuracy of computing multiple eigenvalues using the QR [4], [5] and QZ [6] algorithms. For the large scale circuits, besides these accuracy problems, the efficiency of math procedures [7], [8], [2] also has to be taken into account. Although for relatively simple microwave circuits some version of variable-length arithmetic ([3], e.g.) can be used, this way is inappropriate for the large circuits due to computing time. In this paper, we suggest using relatively new `__float128` GCC or `real(16)` GFortran precisions with partial pivoting to solve the accuracy problem for the larger circuits.

II. Description of Math Procedures With an Example

A. Reducing General Eigenvalue Problem to the Standard One

Circuit equations (linear, or linearized at an operating point for nonlinear circuits) are defined by the matrix equation [1]:

$$\big(s\boldsymbol{D}+\boldsymbol{E}\big)\boldsymbol{x}=\boldsymbol{z}, \tag{1}$$

where s represents Laplace operator, the $\boldsymbol{D}$ and $\boldsymbol{E}$ matrices are composed of circuit reactances and resistances, and $\boldsymbol{x}$ and $\boldsymbol{z}$ are Laplace images of circuit variables and/or sources, respectively [7]. Poles and zeros of a transfer function are computed solving

Manuscript received June 19, 2019; revised July 18, 2019. This paper has been supported by the Technology Agency of the Czech Republic under grants TE01020186 and TH02010981, and by the Czech Science Foundation under grant 18-21608S.

J. Dobeš, J. Míchal, and F. Vejražka are with the Department of Radioelectronics, Czech Technical University in Prague, Czech Republic (e-mails: {dobes, michal, vejrazka}@fel.cvut.cz).

V. Biolková is with the Department of Radioelectronics, Brno University of Technology, Czech Republic (e-mail: biolkova@feec.vutbr.cz).

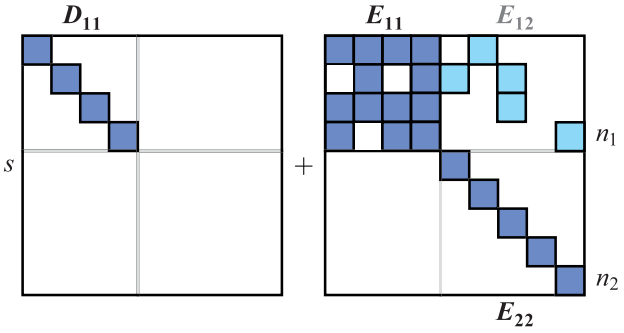

Fig. 1. Final forms of the matrices after the reduction of the general eigenvalue problem to the standard one to be solved by the QR algorithm. The diagonalization of $\boldsymbol{D_{11}}$ and $\boldsymbol{E_{22}}$ is obligatory. (In this case of shapes, there is no need to eliminate $\boldsymbol{E_{12}}$ because its zeroing doesn't change other matrices.)

$$\det\big(s\boldsymbol{D}+\boldsymbol{E}\big)=0 \qquad \text{for poles,} \tag{2a}$$
$$\det\big(s\boldsymbol{D_{(0l)}}+\boldsymbol{E_{(kl)}}\big)=0 \qquad \text{for zeros,} \tag{2b}$$

where the matrix $\boldsymbol{D_{(0l)}}$ arises from $\boldsymbol{D}$ by zeroing its l^{th} column, and the matrix $\boldsymbol{E_{(kl)}}$ arises from $\boldsymbol{E}$ by replacing its l^{th} column by the vector $\boldsymbol{z}$ with all its elements zeroed with the exception of the element corresponding to the k^{th} input source.

A solution of the general eigenvalue problems (2) is more difficult than solving the standard one. Therefore, a systematic reduction (it doesn't change eigenvalues) is applied transforming (2a) to the standard form shown in Fig. 1. (It is an improved alternative to [5]). After this transformation, we can go further:

$$\begin{aligned}\det\big(s\boldsymbol{D}+\boldsymbol{E}\big) &= (-1)^{n_{\updownarrow\leftrightarrow}}\prod_{i=n_1+1}^{n_2}E_{22_{ii}}\det\Big[\underbrace{\boldsymbol{D_{11}}\boldsymbol{D_{11}}^{-1}}_{=\mathbf{1}}\big(s\boldsymbol{D_{11}}+\boldsymbol{E_{11}}\big)\Big]\\ &= (-1)^{n_{\updownarrow\leftrightarrow}}\prod_{i=1}^{n_1}D_{11_{ii}}\prod_{i=n_1+1}^{n_2}E_{22_{ii}}\det\big(s\mathbf{1}+\boldsymbol{D_{11}}^{-1}\boldsymbol{E_{11}}\big)=0,\end{aligned} \tag{3}$$

where $n_{\updownarrow\leftrightarrow}$ is total number of the row and column interchanges during the process, and $\mathbf{1}$ is unity matrix. Finally, determining the poles is finished by computing the eigenvalues of the matrix

$$\boldsymbol{E}'_{\text{poles}}=-\boldsymbol{D_{11}}^{-1}\boldsymbol{E_{11}}. \tag{4}$$

B. Illustrative Example of Calculating the Poles and Zeros

Consider the circuit in Fig. 2, for which (1) gets the form

$$\left[s\begin{pmatrix}C & & -C & \\ & C & -C & \\ -C & -C & 2C & \\ & & & L\end{pmatrix}+\begin{pmatrix}G & & & \\ & G & & \\ & & & 1\\ & & -1 & \end{pmatrix}\right]\begin{pmatrix}V_1\\V_2\\V_3\\I\end{pmatrix}=\begin{pmatrix}VG\\ \\ \\ \end{pmatrix} \tag{5}$$

and the next sequence of the reduction operations is to be done:

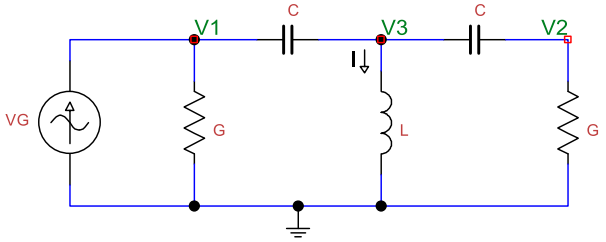

Fig. 2. Simple high pass filter used as an illustrative example of the reduction.

ISBN: 978-988-14048-7-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

- The first and second rows are added to the third row.
- The first and second columns are added to the third column.
- Exchange the third and fourth rows, and then the third and fourth columns.
- $-1/2$ multiple of the fourth column is added to the first, and $-1/(2G)$ multiple of the fourth column is added to the third.

After these operations, the matrix in (5) gets the form of Fig. 1:

$$s\begin{pmatrix} C & & \\ & C & \\ & & L \end{pmatrix}+\begin{pmatrix} G/2 & -G/2 & -1/2 & G \\ -G/2 & G/2 & -1/2 & G \\ 1/2 & 1/2 & 1/(2G) & -1 \\ & & & 2G \end{pmatrix}, \tag{6}$$

therefore, the QR algorithm will be applied to the matrix (4):

$$\boldsymbol{E}'_{\text{poles}} = -\begin{pmatrix} 1/C & & \\ & 1/C & \\ & & 1/L \end{pmatrix}\times\begin{pmatrix} G/2 & -G/2 & -1/2 \\ -G/2 & G/2 & -1/2 \\ 1/2 & 1/2 & 1/(2G) \end{pmatrix}. \tag{7}$$

Evaluating (7) ($G = 1$ mS, $L = 106.1033\,\mu$H, $C = 53.05165$ pF), and applying 25 iterations of the QR algorithm [8], we obtain

$$\boldsymbol{E}'^{(25)}_{\text{poles}} = \begin{pmatrix} -1.88496\times10^{7} & -0.163957 & -89.0719 \\ -0.163957 & -4.33837\times10^{7} & -1.33285\times10^{10} \\ \underbrace{0.000885722}_{\approx 0} & 132537 & 3.86713\times10^{7} \end{pmatrix} \tag{8}$$

with apparent two blocks: 1×1 (the upper left one), that gives the first pole $p_1 = -1.88496\times10^7$, and 2×2 (the lower right one), that gives the second and third poles by the solution of

$$\det\begin{pmatrix} p_{2,3}+4.33837\times10^{7} & 1.33285\times10^{10} \\ -132537 & p_{2,3}-3.86713\times10^{7} \end{pmatrix} = 0 \tag{9}$$

as $p_{2,3} = -2.3562\times10^6 \pm 9.12489\times10^6$ j. (If we divide the first pole by -2π, we obtain the anticipated frequency 3×10^6 Hz.)

The second step is the calculation of the zeros. The matrix in (2b) is obtained by the Cramer's rule for $k = 1$ and $l = 2$:

$$s\boldsymbol{D}_{(02)} + \boldsymbol{E}_{(12)} = s\begin{pmatrix} C & & -C \\ & -C & \\ -C & 2C & \\ & & L \end{pmatrix}+\begin{pmatrix} G & VG & & \\ & & & 1 \\ & & -1 & \end{pmatrix}, \tag{10}$$

and the next sequence of the reducing operations will be done:

- The first column is added to the third column.
- The first row is added to the third row.
- The third row is added to the second row.
- The second and fourth rows are exchanged, and then the second and fourth columns are exchanged.
- Subtract the fourth row from the first and third rows.

After these operations, (10) gets an alternative form of Fig. 1:

$$s\begin{pmatrix} C & & \\ & L & \\ & & C \end{pmatrix}+\begin{pmatrix} & -1 & & \\ & & -1 & \\ G & 1 & G & VG \end{pmatrix}, \tag{11}$$

and the final matrix

$$\boldsymbol{E}'_{\text{zeros}} = \begin{pmatrix} 1/C & \\ & 1/L \end{pmatrix} \tag{12}$$

clearly has a triple zero eigenvalue, i.e., $z_{1,2,3} = 0$ as expected.

C. Extraordinary Movement: a Row from the Right to the Left

The transformation itself seems nothing else than a modification of the Gauss elimination method. However, an exception arises when the matrix $\boldsymbol{D_{11}}$ contains a nondiagonal element, which is irreducible by any diagonal element of this matrix. In such case, a row from the lower part of the matrix is multiplied by the s operator, which is equivalent to moving a row of the $\boldsymbol{E_{22}}$ matrix to the left. That nondiagonal element of $\boldsymbol{D_{11}}$ can then be easily reduced by means of this transferred row.

D. Various Pivoting Methods for the Reduction Algorithms

As the reduction process very often has problems due to the numerical accuracy, a full pivoting is preferred. Therefore, the n^{th} key element is selected as (i.e., from the whole submatrix):

$$D_{nn} \coloneqq \underset{\substack{n\leqq i\leqq n_2\\ n\leqq j\leqq n_2}}{\arg\max} \left|D_{ij}\right|,\ n = 1,\ldots,n_2, \tag{13a}$$

$$\text{if } |D_{nn}| \leqq \epsilon_{\text{full}} \max_{1\leqq i<n} |D_{in}|\text{, then } D_{nn} \coloneqq 0. \tag{13b}$$

For large scale circuits, the reduction algorithm has to be implemented with utilizing sparsity of the matrices. However, the application of the full pivoting is extremely difficult here from a programming point of view. Therefore, a partial pivoting is mostly used in this case, and the n^{th} key element is chosen from the rest of the n^{th} column of the reduced matrix:

$$D_{nn} \coloneqq \underset{n\leqq i\leqq n_2}{\arg\max} |D_{in}|\,,\ n = 1,\ldots,n_2, \tag{14a}$$

$$\text{if } |D_{nn}| \leqq \epsilon_{\text{sparse}} \max_{n<j\leqq l} \left|D_{nj}\right|\text{, then } D_{nn} \coloneqq 0. \tag{14b}$$

Detailed definition of the pivoting strategy and recognizing zero elements other than (13b) and (14b) are defined in [3].

E. Using the 128-bit Arithmetics (of GNU GCC and Fortran)

Although the pivoting methods described in Subsec. II-D improve overall properties of the poles-zeros analysis considerably, their precision is still insufficient for a number of circuits. Therefore, we have changed the standard 64-bit arithmetics to the newly implemented 128-bit (`__float128`) one, and, clearly many (previously problematic) tasks now work well. Moreover, as the GNU compilers have complete libraries for the 128-bit computing, changing the source code was relatively easy.

III. Testing Examples: Different Levels of Complexity

For a clear demonstration how the 128-bit arithmetic can improve the accuracy, we performed the poles-zeros analysis for three practical electronic and microwave circuits of different levels of complexity. Both poles and zeros analyses have been performed for each circuit with both algorithms (partial pivoting, sparse matrix and total pivoting, full matrix) with the original 64-bit arithmetic and with the new 128-bit arithmetic. The partial pivoting sparse matrix case was also run with an arithmetic of a reference accuracy of 2048-bit mantissa, to which all the previous results have been related and compared in tables, ordered with growing order of magnitude. (For an easier orientation in frequency, all poles and zeros presented in tables were divided by 2π, i.e., their physical unit is Hz.)

A. Low Noise Antenna Preamplifier (the simplest, microwave)

The first and simplest example is a radio-frequency circuit, a pHEMT LNA shown in Fig. 3. The analysis of the poles was unproblematic; however, in the zeros analysis, using the `__float128` precisions improved the accuracy considerably.

In the right column of Table I, the accurate zeros (the blue ones) are shown, which were computed with precise variable size arithmetic. (2048-bit was used – this one is extremely precise; however, it is unsuitable for large scale circuits because such computations may be extremely time-consuming.) For the DIFFERING zeros, two phenomena arose, see the 1st–4th column:

- For the partial pivoting, sparse matrix, `double` is completely wrong and `__float128` makes it completely right to all six displayed figures – i.e., `__float128` gave perfect results!
- For the total pivoting, full matrix, it wrongly identifies exact zero values in general, but `__float128` improves the wrong magnitudes by 3–4 orders. (All nonzero values were correct.)

ISBN: 978-988-14048-7-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

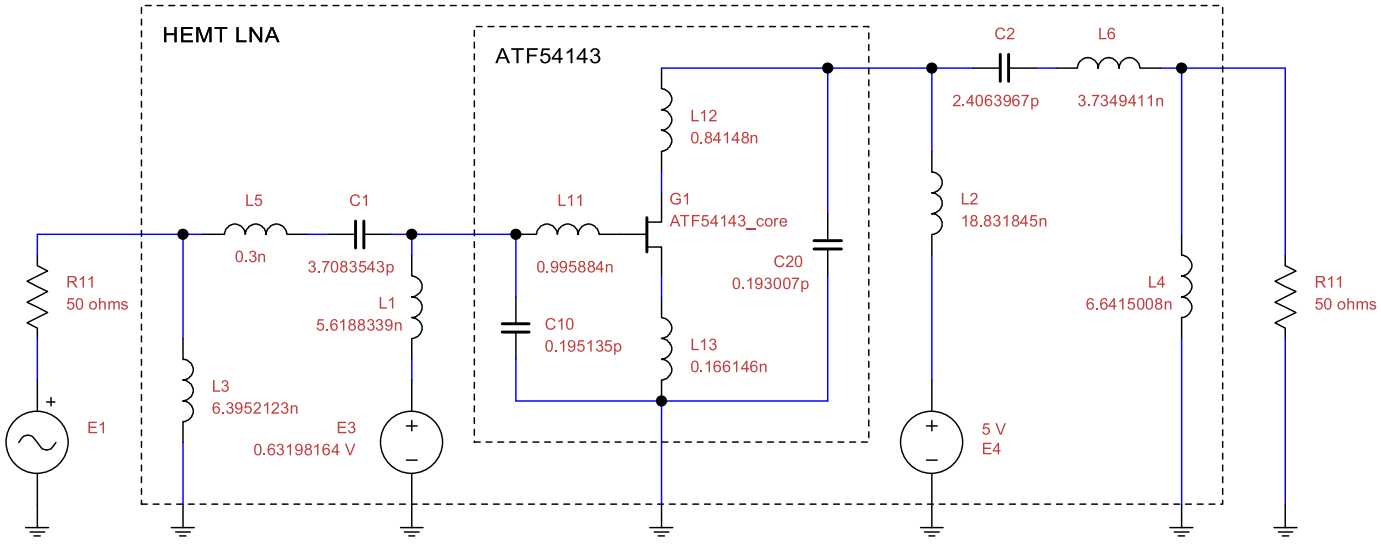

Fig. 3. Diagram of the low noise amplifier for a multiconstellation receiver of satellite navigation. The values were designed by the multiobjective optimization.

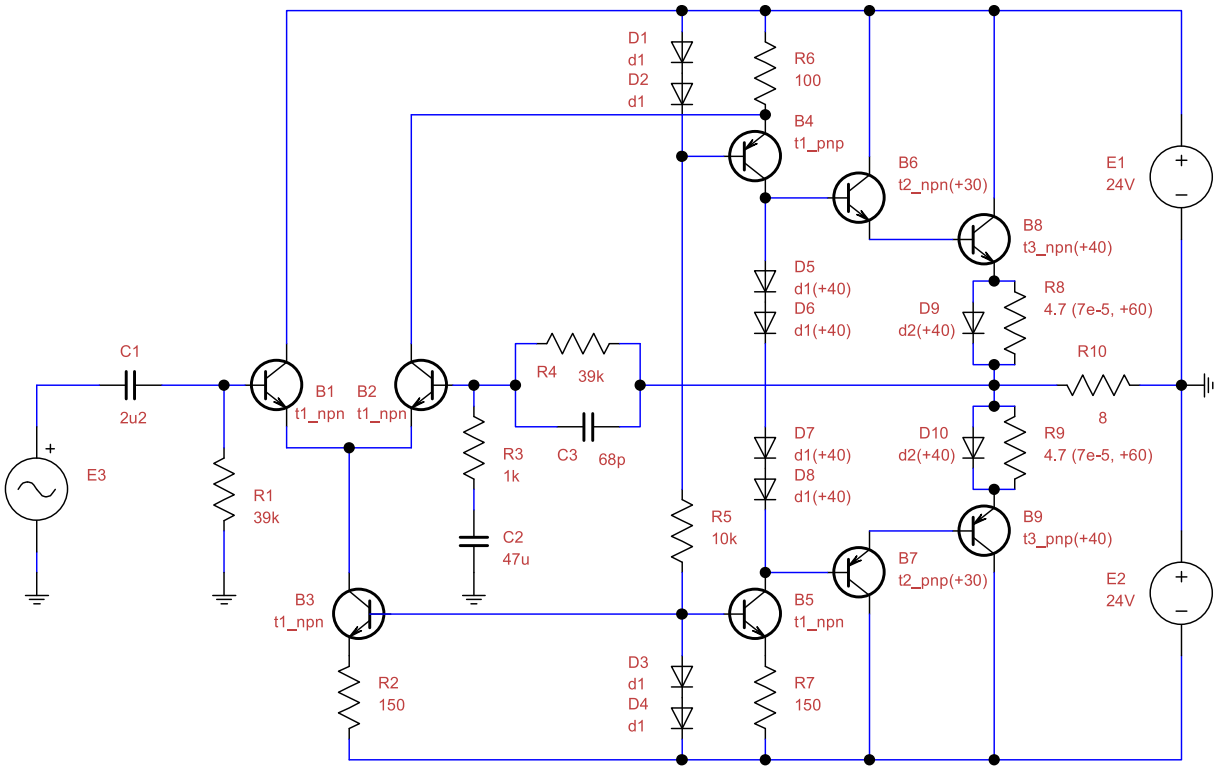

Fig. 4. Diagram of the AB-class power operational amplifier. This is the medium-complexity example of the tests of the accuracy with the results in Table II.

B. Discrete Power Operational Amplifier (medium complexity)

As a medium-complexity example, an AB-class discrete power operational example was analyzed, as shown in Fig. 4. Table II shows comparisons of poles and zeros in its 1st and 2nd parts, respectively. It is clear that the following observations can be made:

- Considering the case of partial pivoting and type `double`, poles that are different from the correct values differ only at the 6th significant place while some zeros differ at the 4th significant place.
- Using the `__float128` and using the full pivoting while keeping `double` have both improved the accuracy of results to a similar extent.
- The use of `__float128` arithmetic improved accuracy for both poles and zeros, partial pivoting and full pivoting.

C. A 272 Integrated Operational Amplifier (the most complex)

The most complex example tried is an integrated operational amplifier of the 272 type. For running the PZ analysis on circuit in Fig. 5, the testbench circuit in Fig. 6 was used. Comparison tables are Tables III and IV. This time the total numbers of poles and zeros are very high: 108 poles and 108 zeros, which makes the tabular comparisons very difficult. Obviously for the partial pivoting algorithm, the differences are very significant, for example:

- The total number of zeros, 105, does not match the correct number of zeros.
- 27 poles and more than 90 zeros differ within 6 significant places.
- Some complex conjugate pairs of zeros have collapsed to a single real-valued zero.

ISBN: 978-988-14048-7-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE I
DIFFERING ZEROS COMPARISON FOR THE LOW NOISE AMPLIFIER EXAMPLE (ALL THE POLES WERE IDENTICAL)

Partial pivoting, sparse matrix		Total pivoting, full matrix		Variable-size ref. (2048-bit used)
`double`	`__float128`	`double`	`__float128`	
$9.91769 \times 10^{-4} \pm 1.71786 \times 10^{-3}$ j	✓	$1.29496 \times 10^{5} \pm 3.98722 \times 10^{5}$ j	8.26655×10^{1}	0
	✓		$-6.68778 \times 10^{1} \pm 4.86649 \times 10^{1}$ j	0
-1.98364×10^{-3}	✓	$-3.39127 \times 10^{5} \pm 2.46349 \times 10^{5}$ j		0
3.17828×10^{3}	✓		$2.55450 \times 10^{1} \pm 7.87415 \times 10^{1}$ j	0
-1.03269×10^{8}	✓	4.19262×10^{5}		0
$3.23532 \times 10^{8} \pm 3.74290 \times 10^{7}$ j	✓	✓	✓	7.76010×10^{10}
-2.01247×10^{10}	✓	✓	✓	$-7.19988 \times 10^{10} \pm 8.82644 \times 10^{10}$ j

TABLE II
DIFFERING POLES (1ST PART) AND ZEROS (2ND PART) COMPARISON FOR THE POWER AMPLIFIER EXAMPLE

Partial pivoting, sparse matrix		Total pivoting, full matrix		Variable-size ref. (2048-bit used)
`double`	`__float128`	`double`	`__float128`	
-6.96956×10^{10}	✓	✓	✓	-6.96958×10^{10}
-7.31332×10^{10}	-7.31331×10^{10}	-7.31331×10^{10}	-7.31331×10^{10}	-7.31330×10^{10}

Partial pivoting, sparse matrix		Total pivoting, full matrix		Variable-size ref. (2048-bit used)
`double`	`__float128`	`double`	`__float128`	
✓	✓	1.47784×10^{-6}	-7.89712×10^{-24}	0
✓	✓	-8.47840×10^{-2}	✓	-8.47825×10^{-2}
-2.39371×10^{6}	✓	✓	✓	-2.39372×10^{6}
-1.32897×10^{7}	✓	✓	✓	-1.32896×10^{7}
-2.51184×10^{7}	✓	✓	✓	-2.51193×10^{7}
-8.18258×10^{7}	✓	✓	✓	-8.18451×10^{7}
$-9.23576 \times 10^{7} \pm 2.16513 \times 10^{8}$ j	✓	✓	✓	$-9.35970 \times 10^{7} \pm 2.16156 \times 10^{8}$ j
-3.47567×10^{8}	✓	✓	✓	-3.47473×10^{8}
-4.92109×10^{8}	✓	✓	✓	-4.91768×10^{8}
-5.15993×10^{8}	✓	✓	✓	-5.16593×10^{8}
-7.54237×10^{8}	✓	✓	✓	-7.53995×10^{8}
-2.37231×10^{9}	✓	✓	✓	-2.42508×10^{9}
-2.42701×10^{9}	✓	✓	✓	-2.42919×10^{9}
unidentified	✓	✓	✓	-9.29471×10^{9}
-1.11849×10^{10}	✓	✓	✓	-1.15561×10^{10}
-7.57337×10^{10}	✓	✓	✓	-6.99856×10^{10}

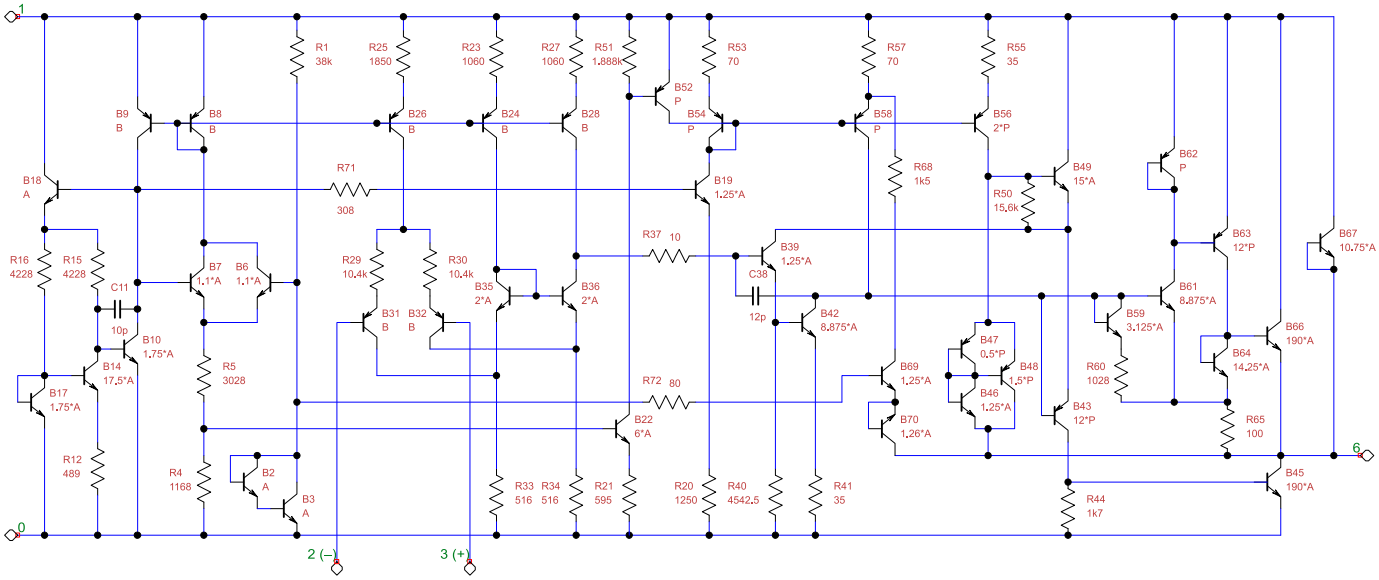

Fig. 5. Diagram of the 272 operational amplifier. This is the most complex example of testing PZ analysis; see also a diagram of a testbench circuit in Fig. 6.

ISBN: 978-988-14048-7-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE III
DIFFERING POLES COMPARISON FOR THE 272 OPERATIONAL AMPLIFIER EXAMPLE

Partial pivoting, sparse matrix		Total pivoting, full matrix		Variable-size ref. (2048-bit used)
`double`	`__float128`	`double`	`__float128`	
-7.82721×10^{8}	✓	✓	✓	-7.82722×10^{8}
-2.56457×10^{9}	✓	✓	✓	-2.56458×10^{9}
-3.09278×10^{9}	-3.09278×10^{9}	-3.09278×10^{9}	-3.09278×10^{9}	-3.09279×10^{9}
-3.15330×10^{9}	-3.15330×10^{9}	-3.15330×10^{9}	-3.15330×10^{9}	-3.15329×10^{9}
-6.05804×10^{9}	✓	✓	✓	-6.05803×10^{9}
-6.85518×10^{9}	✓	✓	✓	-6.85529×10^{9}
-9.24995×10^{9}	✓	✓	✓	-9.24994×10^{9}
-1.00318×10^{10}	✓	✓	✓	-1.00319×10^{10}
-1.53799×10^{10}	✓	✓	✓	-1.53797×10^{10}
$-2.12536 \times 10^{10} \pm 4.03701 \times 10^{8}$ j	$-2.12537 \times 10^{10} \pm 4.03031 \times 10^{8}$ j	$-2.12537 \times 10^{10} \pm 4.03031 \times 10^{8}$ j	$-2.12537 \times 10^{10} \pm 4.03031 \times 10^{8}$ j	$-2.12537 \times 10^{10} \pm 4.03014 \times 10^{8}$ j
-2.65416×10^{10}	✓	✓	✓	-2.65438×10^{10}
-2.90609×10^{10}	✓	✓	✓	-2.90596×10^{10}
-3.60916×10^{10}	✓	✓	✓	-3.60908×10^{10}
-3.66061×10^{10}	✓	✓	✓	-3.66051×10^{10}
-3.75713×10^{10}	✓	✓	✓	-3.75711×10^{10}
-6.24794×10^{10}	-6.24829×10^{10}	-6.24829×10^{10}	-6.24829×10^{10}	-6.24830×10^{10}
-6.28767×10^{10}	✓	✓	✓	-6.28733×10^{10}
-1.04441×10^{11}	✓	✓	✓	-1.04457×10^{11}
-4.03487×10^{11}	✓	✓	✓	-4.03486×10^{11}
-4.39356×10^{11}	✓	✓	✓	-4.38947×10^{11}
-5.26636×10^{11}	✓	✓	✓	-5.26399×10^{11}
-5.66946×10^{11}	-5.66948×10^{11}	-5.66948×10^{11}	-5.66948×10^{11}	-5.66950×10^{11}
-7.12792×10^{11}	✓	✓	✓	-7.12244×10^{11}
-7.18657×10^{11}	-7.18724×10^{11}	-7.18724×10^{11}	-7.18724×10^{11}	-7.18723×10^{11}
-8.75764×10^{11}	-8.75776×10^{11}	-8.75771×10^{11}	-8.75771×10^{11}	-8.75775×10^{11}
-9.65601×10^{11}	✓	-9.65595×10^{11}	-9.65595×10^{11}	-9.65596×10^{11}

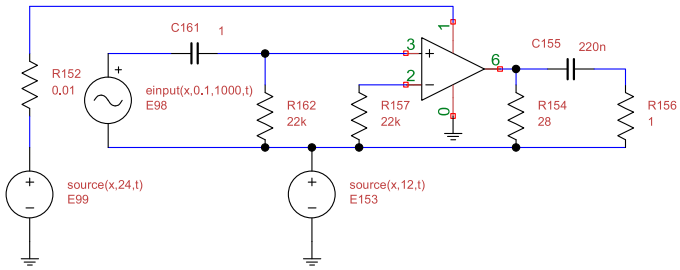

Fig. 6. Diagram of the testbench for the 272 operational amplifier.

- The correct highest-magnitude zero of 7.50920×10^{12} Hz is replaced with one of value -2.57273×10^{15} Hz (of opposite sign!).

We can still conclude that:

- The use of the `__float128` type reduces the numbers of differing poles and zeros to 8 poles and 8 zeros and the maximum differences are at the 6^{th} significant figure (of the real part; with the absolute errors of imaginary parts being similar to those of real parts).
- For the full pivoting algorithm, the results are already correct to 5 significant places and the improvement by the use of libquadmath is less obvious: only one zero changed at the 6^{th} significant place to its correct value.

IV. CONCLUSION

Even though the use of **the `__float128` type arithmetic** is not the ultimate solution to the PZ analysis accuracy issue, **the benefits of its use increase with the circuit complexity, especially when the partial pivoting scheme has to be used**. Furthermore, the algorithm for full matrix and total pivoting is less capable of recognizing truly zero-valued poles and zeros (because the full-matrix algorithm by its nature does not detect 0 values). (And note that **for more extensive circuits**, memory demands may make **the full pivoting algorithms unusable**, which establishes the need for the alternative option of sparse-matrix algorithm.) In this case the `__float128` precision brings the magnitude of such poles/zeros significantly closer to zero.

REFERENCES

[1] T. Rübner-Petersen, "On sparse matrix reduction for computing the poles and zeros of linear systems," in *International Symposium on Network Theory*, Ljubljana, Slovenia, 1979.
[2] T. P. Stefański, "Electromagnetic problems requiring high-precision computations," *IEEE Antennas and Propagation Magazine*, vol. 55, no. 2, pp. 344–353, Apr. 2013.
[3] J. Dobeš and V. Žalud, *Modern Radio Engineering*, 2nd ed. Prague: BEN Publications, Apr. 2010, 768 p., e-book, ISBN 978-80-7300-293-0.
[4] S. M. Rump, "Computational error bounds for multiple or nearly multiple eigenvalues," *Linear Algebra and its Applications*, vol. 324, no. 1–3, pp. 209–226, Feb. 2001.
[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*, Rhodes, Greece, 2009, pp. 178–181.
[6] D. A. Bini and V. Noferini, "Solving polynomial eigenvalue problems by means of the Ehrlich-Aberth method," *Linear Algebra and its Applications*, vol. 439, no. 4, pp. 1130–1149, Aug. 2013.
[7] R. Lehoucq, D. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. Philadelphia, PA: SIAM Publications, 1998.
[8] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, ser. Applied Mathematics. Philadelphia, PA: SIAM Publications, May 2011.

TABLE IV
DIFFERING ZEROS COMPARISON FOR THE 272 OPERATIONAL AMPLIFIER EXAMPLE

Partial pivoting, sparse matrix		Total pivoting, full matrix		Variable-size ref. (2048-bit used)
`double`	`__float128`	`double`	`__float128`	
✓	✓	-1.01548×10^{0}	✓	-1.01546×10^{0}
-4.28541×10^{6}	✓	✓	✓	-4.28569×10^{6}
-5.84862×10^{6}	✓	✓	✓	-5.84856×10^{6}
-6.89483×10^{6}	✓	✓	✓	-6.91173×10^{6}
-8.02368×10^{6}	✓	✓	✓	-8.01724×10^{6}
-8.19455×10^{6}	✓	✓	✓	-8.19448×10^{6}
-8.44644×10^{6}	✓	✓	✓	-8.44644×10^{6}
9.48634×10^{6}	✓	✓	✓	8.60377×10^{6}
-8.96951×10^{6}	✓	✓	✓	-8.96794×10^{6}
-1.14224×10^{7}	✓	✓	✓	-1.14237×10^{7}
-1.50964×10^{7}	✓	✓	✓	-1.52075×10^{7}
1.25189×10^{7}	✓	✓	✓	1.72813×10^{7}
$-1.34644 \times 10^{7} \pm 1.02838 \times 10^{7}\,\mathrm{j}$	✓	✓	✓	$-1.82649 \times 10^{7} \pm 9.54185 \times 10^{6}\,\mathrm{j}$
$-2.45418 \times 10^{7} \pm 1.08143 \times 10^{6}\,\mathrm{j}$	✓	✓	✓	$-2.45418 \times 10^{7} \pm 1.09254 \times 10^{6}\,\mathrm{j}$
$-2.48644 \times 10^{7} \pm 1.55590 \times 10^{7}\,\mathrm{j}$	✓	✓	✓	$-2.48534 \times 10^{7} \pm 1.55435 \times 10^{6}\,\mathrm{j}$
-2.69463×10^{7}	✓	✓	✓	-2.69387×10^{7}
-3.12548×10^{7}	✓	✓	✓	-3.12452×10^{7}
-4.28541×10^{7}	✓	✓	✓	-4.28569×10^{7}
-6.52561×10^{7}	✓	✓	✓	-6.56151×10^{7}
-7.06021×10^{7}	✓	✓	✓	-7.06593×10^{7}
-7.90403×10^{7}	✓	✓	✓	-7.93223×10^{7}
-8.27779×10^{7}	✓	✓	✓	-8.27776×10^{7}
-2.53163×10^{8} unidentified	$-2.13529 \times 10^{8} \pm 1.50255 \times 10^{6}\,\mathrm{j}$	$-2.13529 \times 10^{8} \pm 1.50255 \times 10^{6}\,\mathrm{j}$	$-2.13529 \times 10^{8} \pm 1.50255 \times 10^{6}\,\mathrm{j}$	$-2.13529 \times 10^{8} \pm 1.50252 \times 10^{6}\,\mathrm{j}$
-2.94735×10^{8}	✓	✓	✓	-2.92620×10^{8}
-4.97600×10^{8}	✓	✓	✓	-4.97605×10^{8}
-6.79275×10^{8}	✓	✓	✓	-6.79855×10^{8}
-8.30500×10^{8}	✓	✓	✓	-8.31067×10^{8}
-8.86368×10^{8}	✓	✓	✓	-8.89485×10^{8}
-9.12112×10^{8}	✓	✓	✓	-9.12119×10^{8}
-9.53145×10^{8}	✓	✓	✓	-9.53142×10^{8}
-1.15311×10^{9}	✓	✓	✓	-1.15308×10^{9}
-1.37299×10^{9}	✓	✓	✓	-1.37873×10^{9}
-1.39374×10^{9}	✓	✓	✓	-1.39714×10^{9}
-1.44793×10^{9}	✓	✓	✓	-1.44926×10^{9}
-1.46521×10^{9}	✓	✓	✓	-1.46493×10^{9}
-1.54682×10^{9}	✓	✓	✓	-1.54393×10^{9}
-1.65363×10^{9}	✓	✓	✓	-1.65205×10^{9}
-1.65922×10^{9}	✓	✓	✓	-1.65948×10^{9}
-1.66728×10^{9}	✓	✓	✓	-1.66893×10^{9}
-1.73658×10^{9} unidentified	$-1.75519 \times 10^{9} \pm 8.05508 \times 10^{6}\,\mathrm{j}$	$-1.75519 \times 10^{9} \pm 8.05508 \times 10^{6}\,\mathrm{j}$	$-1.75519 \times 10^{9} \pm 8.05508 \times 10^{6}\,\mathrm{j}$	$-1.75519 \times 10^{9} \pm 8.05507 \times 10^{6}\,\mathrm{j}$
-1.92274×10^{9}	✓	✓	✓	-1.92146×10^{9}
-2.32418×10^{9}	✓	✓	✓	-2.32408×10^{9}
-2.89333×10^{9}	✓	✓	✓	-2.89074×10^{9}
-3.11718×10^{9}	✓	✓	✓	-3.11600×10^{9}
-3.55107×10^{9}	✓	✓	✓	-3.54080×10^{9}
-3.73781×10^{9}	✓	✓	✓	-3.75018×10^{9}
-7.26479×10^{9}	✓	✓	✓	-7.30493×10^{9}
-7.64332×10^{9}	✓	✓	✓	-7.57294×10^{9}
-9.25159×10^{9}	✓	✓	✓	-9.35115×10^{9}
-1.14896×10^{10}	✓	✓	✓	-1.15251×10^{10}
-2.33399×10^{10}	✓	✓	✓	-2.35203×10^{10}
-3.47477×10^{10}	✓	✓	✓	-3.47526×10^{10}
-3.60048×10^{10}	✓	✓	✓	-3.57007×10^{10}
-6.28620×10^{10}	✓	✓	✓	-6.29094×10^{10}
-1.17474×10^{11}	✓	✓	✓	-1.17482×10^{11}
-4.03486×10^{11}	✓	✓	✓	-4.03490×10^{11}
-5.67108×10^{11}	✓	✓	✓	-5.68858×10^{11}
-7.61927×10^{11}	✓	✓	✓	-7.59921×10^{11}
-8.75708×10^{11}	-8.75517×10^{11}	-8.75523×10^{11}	-8.75517×10^{11}	-8.75514×10^{11}
-9.65576×10^{11}	✓	-9.65929×10^{11}	✓	-9.65928×10^{11}
unidentified	$-1.10342 \times 10^{11} \pm 1.15153 \times 10^{12}\,\mathrm{j}$	$-1.10342 \times 10^{11} \pm 1.15153 \times 10^{12}\,\mathrm{j}$	$-1.10342 \times 10^{11} \pm 1.15153 \times 10^{12}\,\mathrm{j}$	$-1.10343 \times 10^{11} \pm 1.15153 \times 10^{12}\,\mathrm{j}$
-2.57273×10^{15}	7.50932×10^{12}	7.50932×10^{12}	7.50932×10^{12}	7.50920×10^{12}

ISBN: 978-988-14048-7-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)