Worst-Case Circuit Analysis using Schematic Conversion and Decomposition

Bálint Bak, Pál Weisz, György Orosz
Department of Measurement and Information Systems
Faculty of Electrical Engineering and Informatics
Budapest University of Technology and Economics
Műegyetem rakpart 3., Budapest, H-1111, Hungary
bakbalint98@gmail.com, weiszpali@gmail.com, orosz@mit.bme.hu

***Abstract*—Worst-case circuit analysis (WCA) is an essential task during developing safety-critical electronic systems. WCA ensures that the system operates correctly under any possible conditions. A practical system contains thousands of requirements that should be checked by WCA, so it is crucial to perform WCA efficiently. This paper proposes a method that can speed up the frequency-domain analysis of linear circuits by converting the schematic into analytical form and then using an automatic decomposition technique.**

***Index Terms*—worst-case analysis, improving simulation performance, converting simulation model into analytical equations, system decomposition**

I. Introduction

During the development process of safety-critical electronic systems, the analysis of the designed product is an important step to ensure that the product is operating correctly, considering component parameter tolerances and possible environmental conditions. The analysis of the effect of extreme conditions on the circuit operation is called worst-case analysis (WCA) [1] [2]. Since a complex circuit may contain several hundreds or thousands of analysis tasks, the efficiency of the WCA solution is crucial.

WCA is typically performed using circuit simulation software (e.g., LTSpice, OrCAD, Tina). These software tools provide some basic and traditional analysis methods for performing WCA, e.g., Extreme Value Analysis (EVA), Monte-Carlo analysis, and sensitivity analysis [1].

However, these programs do not always provide a fast and efficient solution. Simulator programs solve system equations numerically to give solutions for diverse types of systems. Sometimes, it may be possible to describe a problem in analytical form and solve it more quickly than with simulators using numerical algorithms.

Project no. RRF-2.3.1-21-2022-00009, titled National Laboratory for Renewable Energy has been implemented with the support provided by the Recovery and Resilience Facility of the Europian Union within the framework of Programme Széchenyi Plan Plus.

Project no. 2019-1.3.1-KK-2019-00004 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2019-1.3.1-KK funding scheme.

Several methods have been proposed in recent decades to solve WCA tasks using analytical system equations. Some examples are interval arithmetic [3] or affine arithmetic [4] [5]. The analytical form also has the potential advantage of the application of advanced extreme value search techniques [6] [7] [8].

To exploit the advantages of both the simulation software and the analytical solution, we introduce a software tool that is able to convert a circuit schematic into MATLAB equations. This software tool allows us to quickly set up analytical system equations, synchronize the analytical and electrical models, and eliminate human errors while writing system equations manually.

Usually, the operation of electrical systems depends on a lot of parameters and variables; hence, demonstrating and proving that such a system fulfills the requirements is a computationally intensive problem. In order to improve the execution time of analysis tasks, we propose an algorithm for the automatic decomposition of such analysis tasks that involve the evaluation of transfer functions. Breaking the system into smaller subcircuits reduces the complexity of parameter space, so the worst-case value is easier to find.

The paper is structured as follows. Section II provides an overview of our proposed analysis workflow. In Section III, we introduce a MATLAB tool that converts schematics into systems of equations. Section IV introduces the proposed decomposition technique, and an example of our proposed automatic analysis method is presented in Section V.

II. Proposed Method

The illustration of the proposed analysis method is shown in Fig. 1.

The workflow is divided into three steps:

- Creation of system equations according to schematic diagram (netlist file).
- Separation of parameters into disjoint subsets.
- Solution of worst-case problem for the independent parameter sets.

The algorithm for parameter separation into disjoint sets is presented in Section IV. The algorithm works for problems that involve transfer function calculation.

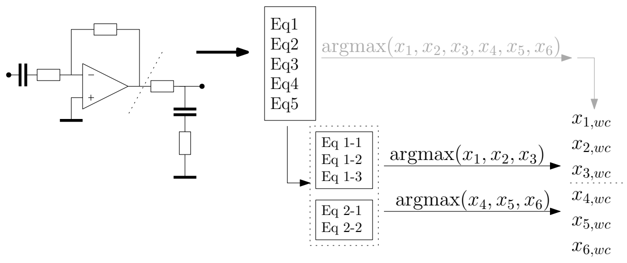

Fig. 1. Illustration of the proposed method.

The above method preserves the advantage of simulation software, i.e., we can use an intuitive graphical representation of circuits and easily create and modify a schematic. Nevertheless, after transforming the circuit into analytical equations, we can take advantage of the analytical form, i.e., faster evaluation and exact solution. Note that since it is an automatic conversation technique, we can eliminate human errors: the equations could also be written manually, but it would be prone to error. The equation generator uses a standard SPICE netlist file [9], which can be generated by most simulation software.

Note that there are some similar conversion tools [10] [11], but they can't be integrated into our MATLAB framework.

A simple example of how the variable decomposition can speed up the WCA process can be illustrated with the EVA method. EVA assumes that minimum or maximum exists at the extreme values of the parameters, so it requires 2^N evaluation of the circuit to be analyzed to find the worst-case solution. If we partition the N parameter into two subsets $N = N_1 + N_2$, the required evaluation number is $2^{N_1} + 2^{N_2}$. For example, if $N = 8$ we have $2^8 = 256$ evaluations, and if $N_1 = 4, N_2 = 4$ we have only $2^4 + 2^4 = 32$ evaluations.

The parameter decomposition could also be used in the case of interval arithmetic since it reduces the complexity of the equations, which generally results in tighter error bounds [12].

The converted analytical model can also be useful when performing an optimization process. In simulation software, generally, we cannot run an arbitrary algorithm using the circuit model; only the built-in functionalities are available. The applicable toolset (e.g., for parameter optimization) increases dramatically by converting the schematic into a MATLAB model.

III. Schematic Conversion

The circuit model of the electrical system is given as a graphical schematic drawn in an arbitrary circuit design or simulator software. Usually, these programs can generate so-called netlist files in which the different types of individual components are represented in text form, as well as their associated nodes, model parameters, etc.

The MATLAB software tool we introduce converts a netlist file automatically into a system of symbolic equations, taking into account different circuit component models and the desired form of the equations.

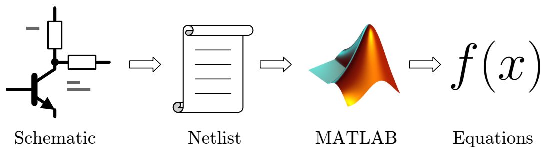

Fig. 3. Workflow of the schematic conversion.

As a first step, schematics to netlist conversion is done using a system call in MATLAB, and then the processing chain continues, as shown in Fig. 2.

Since the circuit may contain non-elemental components whose model or equivalent circuit definition (subcircuit) is located in an external file or parts library, the detailed description of those components must also be included in the netlist. Note that these component model files should be in a similar text format as the netlist describing the system itself. Some libraries may contain others, so these nested libraries should also be imported.

Once all the necessary files are included, a pre-processing step is performed to remove every irrelevant part of the netlist and to standardize the text format to facilitate further processing.

An integrated circuit component may have a multi-level hierarchical structure, i.e., it may contain non-elemental components or other subcircuits in any number of instances or nested at any depth. We aim to simplify the netlist so that equation generation can be done in as few steps as possible. Therefore, extracting the nested subcircuits and decomposing the original hierarchical graph is an important step.

Subcircuit blocks extracted in this way are added to an abstract prototype collection and used as a template in the following steps. The use of templates is practical because there may be several instances of the same type of subcircuit in the original netlist, which do have the same internal structure, but the elemental components they contain are not identical - in terms of the network computation model - for the different instances.

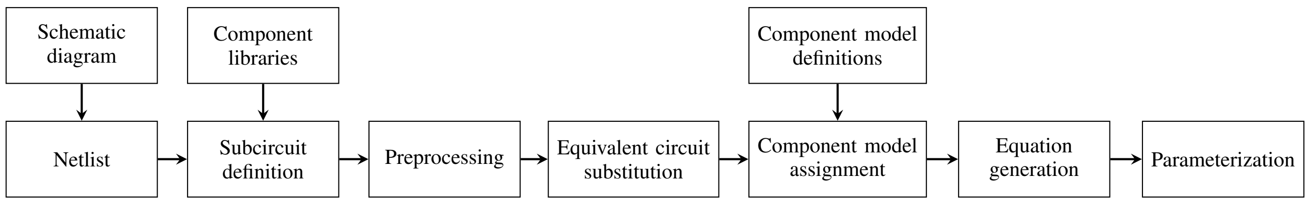

Fig. 2. Software components of the schematic conversion tool.

During instantiation, a copy of the prototype is made, containing elemental components assigned with unique identifiers. Since the subcircuit may contain nodes and components that have the same names as in another - even embedded - subcircuit or in the original netlist, it is essential to ensure that the identifiers remain unique. In this case, the uniqueness of the internal nodes of the sub-circuit must also be provided, while the identifiers of the nodes at higher hierarchical levels must not change.

The instantiation has been done recursively. The recursive approach is justified by the fact that the number of instances required is not known in advance due to the nested nature of the integrated circuits. Theoretically, the iterative method used for the hierarchical template decomposition could solve this problem, but it cannot be used in this case due to the propagation of node identifiers to nested subcircuits.

At this point in the process, we have the final netlist, which no longer contains integrated circuits and is, therefore, directly suitable for the algorithmic generation of the desired equations.

The complete system of network equations can be written based on Kirchhoff's current law, provided that there is a zero voltage reference point in the circuit and the characteristic equations of the components are given, which define the relationship between the currents and the potentials of the terminals.

The equations for the nodal currents are written in terms that express how each component contributes to the node current as a function of the potential of the adjacent nodes. These terms come from the component's characteristic equation, which is defined in the belonging component model file. Such files can also contain system-level constraints or introduce new state variables.

Equation generation creates a standalone MATLAB script, which contains the symbolic variables, an equation for each node arranged to zero, and other necessary equations, e.g., constraints.

These equations can be solved analytically using the MATLAB Symbolic Math Toolbox [13], the value of each variable or component can be parameterized, and the resulting function is ready for further analysis.

IV. System Decomposition

A. Theory

In the case of a linear system, the circuit equations will yield a transfer function in the s-domain, which can be expressed as the fraction of two polynomials [14]:

$$W(s) = \frac{B(s)}{A(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + ... + b_0}{a_n s^n + a_{n-1} s^{n-1} + ... + a_0} \tag{1}$$

In equation (1), the roots of *B(s)* are called zeros, and the roots of *A(s)* are called poles. If *B(s)* and *A(s)* are expressed with irreducible factors, the transfer function can be expressed with the poles, zeros, and the static gain, *K*:

$$W(s) = K \frac{\prod_{i=1}^{m} (s - z_i)}{\prod_{j=1}^{n} (s - p_j)} \tag{2}$$

$$K = \frac{b_m}{a_n} \tag{3}$$

Since it was derived from the circuit, *K*, z_i, and p_j are all functions of the circuit parameters. If the analysis aims to determine the gain or phase of the system, it can be expressed the following way at a given ω frequency, derived from (2):

$$|W(j\omega)| = |K| \frac{\prod_{i=1}^{m} |j\omega - z_i|}{\prod_{j=1}^{n} |j\omega - p_j|} \tag{4}$$

$$arg(W(j\omega)) = \sum_{i=1}^{m} [arg(j\omega - z_i)] - \sum_{j=1}^{n} [arg(j\omega - p_j)] \tag{5}$$

The gain can be calculated as a product of terms, where the terms depend on the poles, zeros, and the static gain. The phase can be calculated as a sum of terms where the terms depend only on the poles and zeros. A pole or zero is not necessarily a function of all circuit parameters ($\boldsymbol{x}$) but only of a subset of them ($\boldsymbol{x_k}$). Following this idea:

$$|W(j\omega)| = |K(\boldsymbol{x})| \prod_{k=1}^{m+n} f_k(\boldsymbol{x_k}) \tag{6}$$

$$arg(W(j\omega)) = \sum_{k=1}^{m+n} g_k(\boldsymbol{x_k}) \tag{7}$$

In equations (6) (7), $\boldsymbol{x_k}$ is a subset of the circuit parameters that influence a given pole or zero. A particular case is present if the poles and zeros are real. In this case, f_k and g_k are monotonic functions of the real poles and zeros. In the case of a WCA task, the objective is to calculate the extreme values (minima and maxima). Therefore, we aim to find the extreme values of K, f_k, and g_k functions in the parameter space defined by $\boldsymbol{x}$. If the terms are functions of disjoint subsets of $\boldsymbol{x}$, they can be analysed independently in the parameter spaces defined by the subsets, which may lead to fewer calculations. Two terms (of a product or sum) are *independent* if they have no common parameters or variables influencing them.

The possibility of decomposition of the transfer function is not the general case; it depends on how the poles, zeros, and static gain are constructed from the circuit parameters. To obtain such a decomposition, we must observe the polynomial roots and the static gain and order them in groups, where disjoint subsets of parameters influence them. In the most simple case, *A(s)* and *B(s)* are polynomials with a maximum degree of three [15]. In this case, the roots can be expressed in closed form, from which it can be seen how the parameters influence them. If *A(s)* and *B(s)* are polynomials of higher degree, numerical methods can be used to obtain the roots. This information is enough to rewrite (7) into a sum of independent terms. If the parameters can be gathered into *L* disjoint subsets:

$$arg(W(j\omega)) = \sum_{l=1}^{L} G_l(\boldsymbol{x_l}) \tag{8}$$

Where $\boldsymbol{x_l}$ are disjoint subsets of the whole set of parameters ($\boldsymbol{x}$), therefore the following stands:

$$\bigcup_{l=1}^{L} \boldsymbol{x_l} = \boldsymbol{x} \tag{9}$$

$$\forall i, i \neq j : \boldsymbol{x_i} \cap \boldsymbol{x_j} = \emptyset \tag{10}$$

Equation (8) can reduce the computational load since the minima and maxima can be searched in the parameter spaces defined by $\boldsymbol{x_l}$ instead of the space defined by $\boldsymbol{x}$. This means that the searches can be done independently and also in a lower dimensional parameter space since $\boldsymbol{x_l} \subseteq \boldsymbol{x}$

With the observation of roots, (7) could be rewritten as the sum of independent terms, but to rewrite (6) into a product of independent terms, additional information is needed since the static gain *K(x)* also needs to be decomposed. K can be obtained from (3) in closed form regardless of polynomial order. If the gain is known, it can be checked analytically if the same partitioning can be applied to decompose K into a product of independent terms. Suppose a partitioning of parameters into L disjoint subsets is found based on the information of roots. If the same partitioning can be applied to *K*, (6) can be decomposed into a product of independent terms:

$$|W(j\omega)| = \prod_{l=1}^{L} F_l(\boldsymbol{x_l}) \tag{11}$$

This decomposition can reduce the computational load for the same reasons as (8).

B. Implementation

Our method is based on numerical sensitivity analysis: we calculate the nominal poles ($p_{i,0}$) and zeros ($z_{i,0}$), then observe the changes if a single parameter is modified. This will be done by reevaluating the transfer function, recalculating the roots, and then comparing them to the nominal roots. Formally:

$$\Delta p_{i,j} = root_i \left\{ A(s, x_1, \ldots, x_j + \Delta x_j, \ldots, x_N) \right\} - p_{i,0} \tag{12}$$

$$\Delta z_{i,j} = root_i \left\{ B(s, x_1, \ldots, x_j + \Delta x_j, \ldots, x_N) \right\} - z_{i,0} \tag{13}$$

If $\Delta p_{i,j} \neq 0$ or $\Delta z_{i,j} \neq 0$, the parameter x_j has an impact on the pole p_i or z_i, respectively. The equality check to zero should be performed with a certain tolerance to avoid numerical imprecision.

Since higher-order polynomials are analysed, numerical methods are used. The observation of roots results in a table where it is noted how the parameters influence the poles and zeros. An example of this can be seen in Fig. 4. K is calculated in closed form from b_m and a_n, then checked if the same partitioning can be applied using symbolic manipulations.

If a partitioning is found, minima and maxima can be searched in the parameter spaces defined by the disjoint subsets. Note that in (8) and (11), F_l and G_l functions are not known explicitly, therefore the search will be done by evaluating (1). If a combination of subset parameters

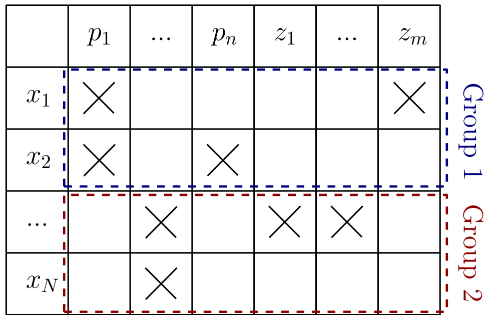

Fig. 4. Example of circuit parameters influencing the roots.

corresponding to an extreme value is found, we'll store the combination and look for the extreme value in the space defined by the following subset. We will do this for all subsets, and due to the independence of terms in (8) and (11), we will end up knowing the parameter combinations corresponding to the extreme values in the entire space of parameters.

V. Example

To demonstrate our method, we use the EVA method to do the WCA of an active low pass filter shown in Fig. 5. The operational amplifiers will be modeled as ideal amplifiers. The passive components have 1% tolerance, and their nominal values are shown in Table I. Capacitance values are in nanofarads; resistance values are in kiloohms.

The netlist is generated from LTSpice. By solving the system equations for $W = \frac{V_{out}}{V_{in}}$, the transfer function was calculated in the s-domain, resulting in a transfer function with six poles and without zeros. The processing of the netlist files, equation solving, symbolic manipulation, and decomposition procedure was done in the MATLAB programming environment using many features of the Symbolic Math Toolbox.

To explore the parameters' influence on the poles, the nominal roots were calculated and then recalculated *2N* times (*N* is the number of parameters, in this case 15). In each iteration, a single parameter was set to its minimum or maximum (based on the tolerance), while the other parameters were set to their nominal values. The roots were obtained with numerical methods. The calculated poles were truncated to 8 significant decimal digits to avoid errors due to numerical imprecision. The transfer function's static gain was expressed from the coefficients based on (3). K was decomposed symbolically into components. After the decomposition, three disjoint subsets of parameters were formed, as shown in Table II.

In the example, the gain characteristic of the circuit was evaluated over a logarithmic frequency scale ranging from

TABLE I
Parameter values (nF, kΩ)

C₁₁	***C₁₂***	***C₂₁***	***C₂₂***	***C₃₁***	***C₃₂***	***R₁₁***	***R₁₂***
10	33	6.8	47	2.2	100	6.2	6.2

R₁₃	***R₂₁***	***R₂₂***	***R₂₃***	***R₃₁***	***C₃₂***	***R₃₃***
12	5.6	5.6	15	7.5	7.5	15

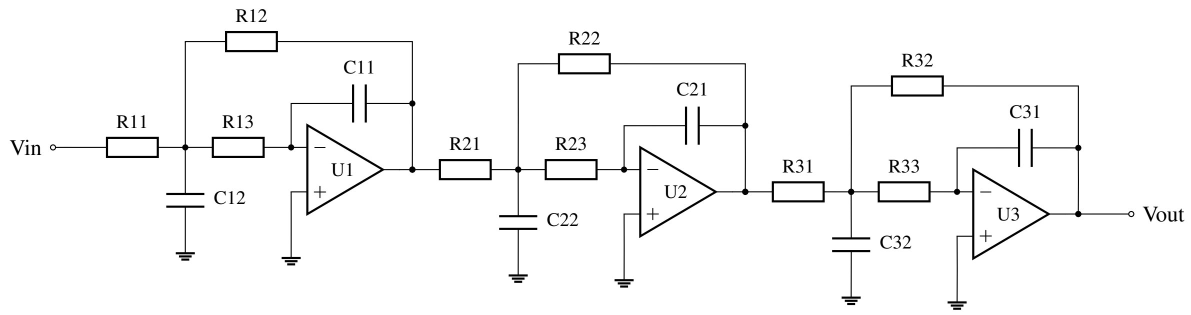

Fig. 5. Schematic of example circuit.

1 Hz to 1 MHz, with 500 points in each decade. The EVA was done for each subset. The parameter combinations corresponding to the extreme values were stored. The minimal and maximal gain were calculated for each point at the frequency scale, and the corresponding sets of parameters were saved. Results are shown in Fig. 6.

The EVA was also done for the same circuit in LTSpice and in the MATLAB environment based on the analytical transfer function but without the decomposition for comparison. All three methods ended with the same results. The runtime of the methods and some of their subsections was also measured with MATLAB. Since the runtimes can be influenced by the processor's and operation system's state, ten consecutive runs were measured and averaged. All the simulations were done on the same computer equipped with a 2.30 GHz Intel Core i7-11800H CPU. The runtimes are summarized in Table III.

We can make the following conclusions. Even the conversion of the schematic model into an analytic equation resulted in a reasonable decrease in the runtime, i.e., the original analysis time decreased from 247 sec to 11.5 sec. The decomposition of parameters into disjoint subsets further reduces the analysis time to 3.24 sec. The analysis time decreased almost by two orders of magnitude.

The limitations of the proposed method are the following. The results show that the conversion from schematic to equation takes almost three seconds, which is not a negligible runtime. If the execution time of the WCA task in the simulator software is in the same order of magnitude, the user can decide whether to use this method. In the actual form, the decomposition can be solved for transfer function evaluation; other analysis tasks are not currently supported. It is also a design decision whether to try to decompose the system because there could be cases where decomposition cannot be solved, but it consumes additional time.

TABLE II
DISJOINT SUBSETS OF PARAMETERS

$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{x_3}$
C_{31}, C_{32}, R_{31}, R_{32}, R_{33}	C_{11}, C_{12}, R_{11}, R_{12}, R_{13}	C_{21}, C_{22}, R_{21}, R_{22}, R_{23}

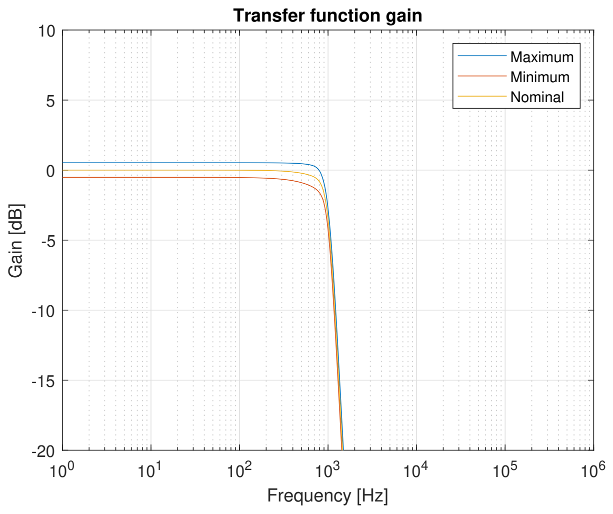

Fig. 6. Minimal, nominal and maximal gain of the circuit in the example.

VI. Conclusion

This paper presented an analysis method that can be used to speed up the process of circuit analysis tasks where the evaluation of the transfer function is required. This tool automatically converts a schematic into an analytic equation, and a further post-processing step is available where system parameters are decomposed into disjoint subsets. The automatic conversion tool reduces the risk of errors during composing the system equations. The decomposition algorithm uses the pole-zero form of the transfer function. By partitioning the parameter space into several independent subspaces, it is easier to find global extreme values.

TABLE III
RUNTIME SUMMARY

runtime in [sec]	LTSpice EVA	MATLAB EVA		MATLAB decomp+EVA		
		eq. from schem.	EVA	eq. from schem.	decomp.	EVA
subtotal	247	2.94	8.57	2.94	0.245	0.059
total	247	11.5		3.24		

Our method generally ensures faster evaluation in those cases where the evaluation of analytic equations can be solved more efficiently than with numerical methods. The analytical form also has the potential advantage of applying advanced extreme value search techniques.

Plans involve extending the decomposition methods on other analysis tasks like time-domain analysis. The decomposition step assumes, at this time, total decoupling between the subsystems. The extension of the separation method in the case of loosely coupled subsystems could be promising.

References

[1] B. Johanson, D. Russell, W. Swavely, Reliability Analysis Center, "Worst case circuit analysis application guidelines," Reliability Analysis Center, 1993

[2] B. A. Lenertz, "Electrical design worst-case circuit analysis: guidelines and draft standard (REV A)," AEROSPACE REPORT NO. TOR-2013-00297, The Aerospace Corporation, June 3, 2013.

[3] C. M. Rocco, "Variability analysis of electronic systems: classical and interval methods," in *Annual Reliability and Maintainability Symposium*, Philadelphia, PA, USA, 1997, pp. 188-193, doi: 10.1109/RAMS.1997.571704.

[4] N. Femia and G. Spagnuolo, "True worst-case circuit tolerance analysis using genetic algorithms and affine arithmetic," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 9, pp. 1285-1296, Sept. 2000, doi: 10.1109/81.883323.

[5] T. Ding, L. Zhang, R. Trinchero, I. S. Stievano and F. G. Canavero, "Worst-Case analysis of electrical and electronic equipment via affine arithmetic," in *2017 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, Verona, 2017, pp. 991-993,

[6] M. W. Tian and R. C.-J. Shi, "Worst case tolerance analysis of linear analog circuits using sensitivity bands," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 8, pp. 1138-1145, Aug. 2000

[7] M. Sunun and S. Uatrongjit, "Improvement of sensitivity band technique for worst case tolerance analysis of linear circuits," in *2008 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, Krabi, Thailand, 2008, pp. 713-716

[8] E. Niculescu, D-M. Purcaru, M. Niculescu, "An approach to worst-case circuit analysis," *WSEAS Transactions on Electronics*, ISSN: 1109-9445, vol. 4, no. 10, Oct. 2007., pp. 237-244

[9] L. W. Nagel, D. O. Pederson, "SPICE (Simulation Program with Integrated Circuit Emphasis)," Memorandum No. ERL-M382, University of California, Berkeley, Apr. 1973

[10] A. Luchetta, S. Manetti and A. Reatti, "SAPWIN-a symbolic simulator as a support in electrical engineering education," *IEEE Transactions on Education*, vol. 44, no. 2, pp. 9 pp.-, May 2001.

[11] CircuitNAV, https://circuitnav.pythonanywhere.com/ [Accessed: Feb. 6., 2024.]

[12] N. Femia and G. Spagnuolo, "Genetic optimization of interval arithmetic-based worst case circuit tolerance analysis," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 46, no. 12, pp. 1441-1456, Dec. 1999

[13] MathWorks. (n.d.). Symbolic Math Toolbox - MATLAB. [Online]. Available: https://www.mathworks.com/products/symbolic.html. [Accessed: Feb. 7., 2024.]

[14] K. Warwick, An Introduction to Control Systems. World Scientific Publishing Company, 1996.

[15] Y.-B. Jia, "Roots of Polynomials," 2020. [Online]. Available: https://faculty.sites.iastate.edu/jia/files/inline-files/polyroots.pdf [Accessed: Feb. 7., 2024.]