An LTSpice - MATLAB Interface for Mitigating Convergence Problems in Circuit Optimization with SPICE

Kubulus, Pawel Piotr; Jørgensen, Asger Bjørn; Beczkowski, Szymon Michal; Munk-Nielsen, Stig

Published in:
2022 IEEE International Workshop on Integrated Power Packaging (IWIPP)

DOI (link to publication from Publisher):
10.1109/IWIPP50752.2022.9894135

Publication date:
2022

Document Version
Accepted author manuscript, peer reviewed version

An LTSpice – MATLAB Interface for Mitigating Convergence Problems in Circuit Optimization with SPICE

Paweł Piotr Kubulus
AAU Energy
Aalborg University
Aalborg, Denmark
ppk@energy.aau.dk

Asger Bjørn Jørgensen
AAU Energy
Aalborg University
Aalborg, Denmark

Szymon Michał Beczkowski
AAU Energy
Aalborg University
Aalborg, Denmark

Stig Munk-Nielsen
AAU Energy
Aalborg University
Aalborg, Denmark

***Abstract*—With the recent emergence of wide-bandgap semiconductors the power electronics design process moves gradually towards the digital space. However, SPICE software that is a dominant tool for accurate transient analysis still lacks in both functionality and robustness needed in order to explore large design spaces unveiled by the digital twin models. In this work, a SPICE circuit solver has been interfaced with MATLAB programming environment. Compared to the literature, main focus has been put on mitigating convergence errors inherent to SPICE solvers without altering the solver algorithm itself. Convergence becomes a crucial problem when SPICE is used for optimization within a big solution space, therefore it is necessary to ensure it. This was achieved by implementing parameter variation based on circuit element tolerance, solver parameter adaptation and initial condition inheritance. The effects of each approach are compared for the case of Double-Pulse Test circuit, by the means of Not-a-Number (NaN) results proportion and runtime.**

***Index Terms*—SPICE, Convergence, Interface, MATLAB, Optimization**

I. Introduction

The fast switching speeds of wide bandgap semiconductors and their small device capacitances, make them sensitive to parasitics elements. For robust and safe operation they require tuning of the circuit elements such as the gate impedance, damping of DC-link capacitors, common-mode chokes to dampen resonances etc. Such tuning is tedious and challenging to do manually by trial and error experimentally, thus using optimization is advantageous. However, this requires the SPICE solver to converge reliably over thousands or millions of simulation runs.

The attempts of integrating SPICE circuit simulators with programming environments enabling optimization date back to the beginning of 21st century [1]. Interfaces to MATLAB, Python and recently Julia have been developed, enabling circuit optimization with different SPICE solvers. However, they did not gain widespread usage due to factors ranging from simple unavailability of software, to limitation by graphical interface and scarce supports for SPICE commands. Additionally, convergence issues have not been addressed on the interface level, as works in this area focus on solver modification [2]. Recent works provide alterations to the ngspice algorithm, improving the pseudo-transient analysis (PTA) methods [3] and attempting to incorporate machine learning methods in SPICE simulation [4]. Improved convergence criterion has been proposed in [2], in order to combat the recurring issue of false convergence due to oscillatory behaviour. Multiple works focus on switching device SPICE modelling, with particular attention given to the convergence [5].

In this work, LTSpice-MATLAB interface with convergence improvement techniques for circuits involving parasitic elements is presented. Optimization in this type of problems brings out convergence issues, due to the unknown initial conditions for parasitic network combined with the heavy non-linearities of switching devices. The main target of the interface and proposed methods are time domain simulations for circuit optimisation problems over a large design space.

II. SPICE convergence in time domain simulation

Commercial SPICE software nowadays is based on the modified SPICE3, using iterative Newton-Rhapson algorithm in order to solve non-linear nodal equations in the time domain [7]. For Newton-Rhapson algorithm, convergence is only guaranteed if the initial guess is sufficiently close to the solution [6]. This can prove challenging, when only partial or no information on initial conditions are available. In order to mitigate the convergence issues of Newton-Rhapson algorithm, source stepping, Gmin stepping and pseudo-transient analysis are commonly deployed [8]. However, aside from very poorly scaling Gmin stepping, none of these methods guarantee achieving a valid DC solution [3]. Moreover, Gmin stepping tends to be prone to bifurcations, folds and discontinuities [3]. Additionally, a case worth mentioning is when convergence criteria are considered fulfilled even though the solution is incorrect. This false convergence can be caused by a slow enough change in device current between iterations to pass the convergence tests despite not fulfilling the Kirchoff's Current Law [2].

In the field of power electronics, the convergence conditions are oftentimes difficult to fulfill. Firstly, switching device models can contain discontinuities. Next, when parasitic elements are involved it is challenging to achieve an initial guess close enough for achieving convergence. Additionally, high differences in parameter values may require increased numerical precision. Finally, the sensitivity of the gate circuits might require a precise tuning of solver settings in order to converge. The above mentioned issues are addressed by methods of mitigating convergence problems proposed by this work.

III. LTSpice - MATLAB interface

Before going further towards the proposed convergence improvement methods, LTSpice - MATLAB interface used is described.

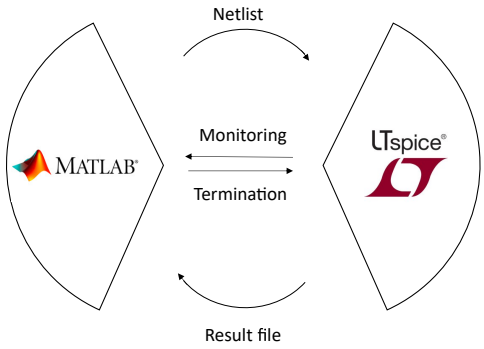

Fig. 1. Simplified interface flowchart

The interface converts MATLAB object description of circuit elements, libraries, initial conditions and settings into a SPICE netlist. Those elements can be freely manipulated within MATLAB environment. The composed netlist is then simulated by calling LTSpice using a DOS command, which also can be freely parametrized for switching between default and alternate solver. As LTSpice continuously saves the transient simulation results point by point, the simulation can be monitored from MATLAB in order to detect switching failures, unexpected excessive changes in waveforms and simulation points that are most computationally heavy early on. When simulation is completed, result file is read into the MATLAB workspace using the LTSpice raw file reader [10] and user-defined parameters are computed. The data collected from simulation supervision and results are later on used in order to improve convergence and simulation speed in the next simulation.

IV. Proposed Methods

As modifying the solver algorithm is normally not a viable option in common engineering practice due to extensive usage of proprietary algorithms, proposed methods focus on improving the convergence conditions in the parametric or model sweeps without altering the algorithm itself.

A. Solver parameter variation (Method 1)

In the first method, the possibility to automatically adapt solver settings in the developed interface is taken advantage of. A linear adaptation scheme is used, in which the option command parameters are multiplied by user-defined constants upon detection of NaN result returned by the solver. Even though this approach relies on a decent solver parameter guess at the start of the parameter sweep, it still provides a certain degree of flexibility, much needed for parametric sweeps to converge over the whole design space. Even a simple relative tolerance adaptation proved effective in the considered test circuit.

B. Element value variation (Method 2)

The second method proposed mimics the sensitivity analysis in order to ensure that convergence issues are not caused by an unfortunate design parameters selection. Since all of the real circuit elements have some parameter variation originating from the manufacturing process, the parameter value can be moved within tolerance band without compromising the result validity. For example, a 10Ω resistor with +/-10% tolerance can be safely assumed to be 10,5Ω for the analysis sake.

Based on practical observations, there are certain design points in which convergence problems emerge even though the surrounding design points exhibit proper convergence. While the cause is usually challenging to pinpoint, the problem can be resolved by tightening the solver parameters using previously described method. However, this increases the computational cost and can be avoided in some cases by a slight change of circuit element parameter value and re-simulation with unchanged solver settings. Usually, simulation time for NaN-returning case is much lower than with a proper convergence, therefore extra computational cost when this method fails can be considered negligible.
For this reason, element specification in the interface includes element tolerance and deployment of this method shifts specified elements to a random value within the tolerance band upon encountering a NaN return from the solver.

There are two main drawbacks to this method. Firstly, the actual distribution of element tolerance is usually unknown and the circuit shifted by maximum tolerance value is most likely different from what can be expected from physical implementation. This problem can be solved by limiting the tolerance band further or by limiting the number of design parameters shifted. Secondly, if another underlying issue preventing convergence is present, applying this method can result in just increasing the overall computational cost. This issue can be limited by monitoring the time consumption

and effectiveness of this method on the run and disabling it when the extra computation cost exceeds a chosen threshold.

C. Initial condition inheritance (Method 3)

Unlike the other two, previous convergence is a prerequisite for this method. Since the quadratic convergence of Newton-Rhapson algorithm relies strongly on the good initial guess, using the results of previous simulations in the parametric sweep can significantly decrease convergence issues and improve convergence speed.

While LTSPICE provides a possibility of doing so using .savebias and .loadbias commands, it proves to be insufficient for the case of wide parametric analysis or optimisation. This is due to the usage of .nodeset command, which only provides node voltage settings for the initial DC operating point analysis. Inductor currents are ignored, and the start of transient simulation may differ from conditions in the simulation from which the bias was saved. This can be improved by using .ic commands that force the inductor currents at the beginning of transient simulation, turning the simulation start into a de-facto pseudo-transient analysis. It was observed that this approach is more effective in improving convergence.

An additional advantage of storing the initial condition data is being able to choose the most suitable design point to inherit from.

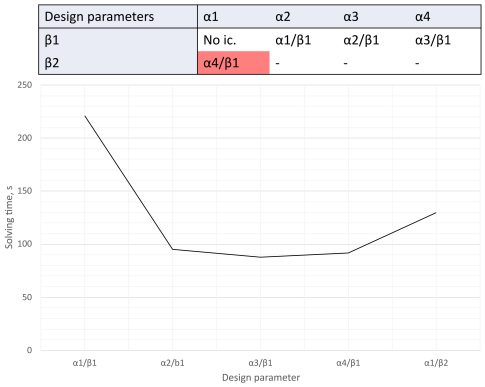

Fig. 2. Normal initial condition inheritance.

To illustrate this, let the two-parameter (α and β) sweep be taken into account. Using the built-in .savebias and .loadbias commands, the new simulation uses either the bias from the previous simulation, or from one predefined design point (preferably in the middle of design space). In the first case, after sweeping over all of the α values for the first value of β, a jump back to the first α value occures and the next simulation would be loading a bias from a distant design point resulting in a compromised performance. In the second case, all of the design space edges experience a similar problem. Utilising stored initial condition data, normalised parameter distance can be used in order to use the closest design point for initial condition inheritance.

Fig. 2 describes the problem of initial condition inheritance without normalised parameter distance.

D. Interface algorithm including proposed method combination

The final proposed algorithm, implemented in the interface has been depicted in the Fig. 3. It was assumed, that M1 can achieve convergence if the relative tolerance is sufficiently tightened.

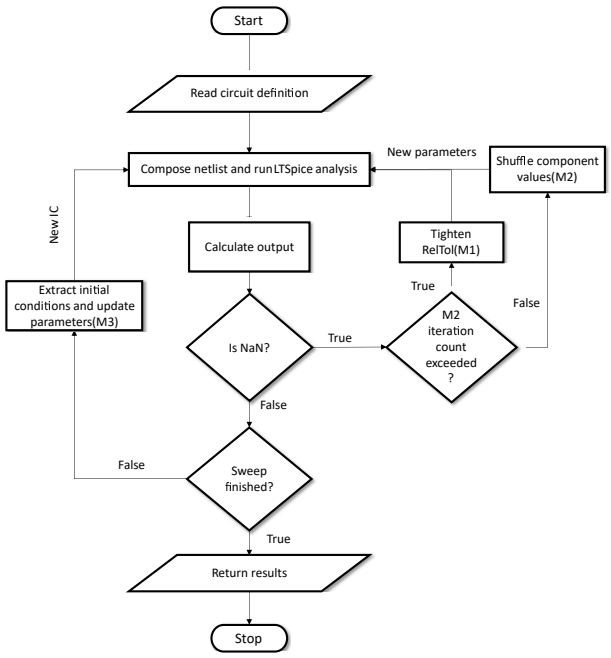

Fig. 3. Interface algorithm incorporating proposed methods.

Upon encountering a NaN return from LTSpice solver, the element parameter variation method is deployed first. In case the iteration limit specified for this method is reached, solver parameter adaptation method takes over as the ultimate way to force convergence. The converged results are used in order to prepare a full set of initial conditions for the next design point.

V. Performance Assessment

In order to assess the performance of proposed methods, simulations were performed using different solver settings, method combinations and iteration limit settings. Tight and loose solver settings were obtained in a following manner. First, the relative tolerance was decreased until test simulations would have convergence problems at every design point giving the 'loose' settings. Second, the relative tolerance was tightened until the full sweep was convergent, and would constitute the 'tight' settings. The method iteration limits, adaptation settings and combinations used are described in the Section V-B.

For each of the assessment cases, total computation time and NaN return density, defined as a percentage of NaN returns in the whole sweep results, are recorded.

A. Test case and setup description

In order to benchmark the performance of proposed methods, a Double Pulse Test circuit with full parasitic elements model extracted using ANSYS Q3D was used. The simplified schematic can be seen below.

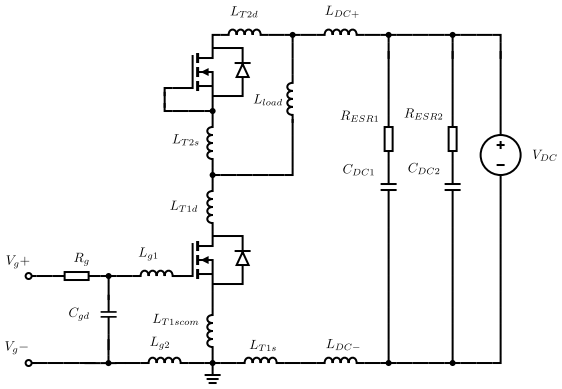

Fig. 4. Assessment circuit schematic.

The elements used were listed in the table below. Parasitic network and netlist used is available online [9].

TABLE I
ELEMENTS

Element	Type/Value	Unit
MOSFETs	CREE C3M0030090K	-
R_g	1-10	Ω
C_g	1-10	nF
C_{DC1}	100	nF
C_{DC2}	1	μF
R_{DC1}	1	mΩ
R_{DC2}	16	mΩ
L_{load}	10	μH
V_{DC}	400	V

The SiC MOSFET models provided by manufacturer include the temperature dependencies and parasitics of TO-247-4 package. The model uses voltage dependent gate-drain and drain-source capacitances, however the gate-source capacitance is constant. Parasitic BJT effects is not modelled.

Initial solver settings deployed were listed in the table below.

TABLE II
SOLVER SETTINGS

Setting	Loose	Tight
gmin	1e-12	1e-12
abstol	1e-12	1e-12
reltol	0.01	0.001
chgtol	1e-14	1e-14
trtol	1	1
vntol	1e-7	1e-7
sstol	0.001	0.001
cshunt	1e-15	1e-15
mindeltagmin	0.00001	0.00001

The assessment was performed on a Lenovo Thinkpad, equipped with Intel(R) Core(TM) i5-10210U @ 1.60GHz-2.11 GHz processor and 32GB RAM.

B. Assessment results

The time consumption of M1 depending on relative tolerance adaptation constant chosen has been investigated. The results were depicted in the Fig. 5. No iteration limit has been placed in this investigation.

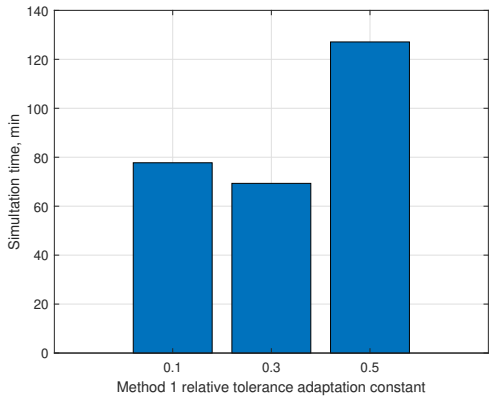

Fig. 5. Sweep time consumption variation with the Method 1 adaptation constant.

The dependence of Method 2 effectiveness on the iteration limit set was investigated with the results depicted in Fig. 6. Loose relative tolerance settings were applied.

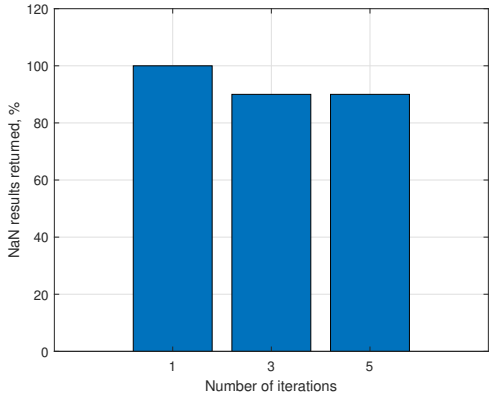

Fig. 6. NaN return density variation with iteration number.

The comparison of proposed methods in terms of computational cost and NaN result returns has been depicted in Fig. 7. The iteration limit of 3 was used for Method 2, and the standalone operation of Method 3 has been enabled by simulating the first design point using tightened relative tolerance. M0 and M0+ denote simulation results without any of the proposed convergence improvement methods, with loose and tightened relative tolerance respectively.

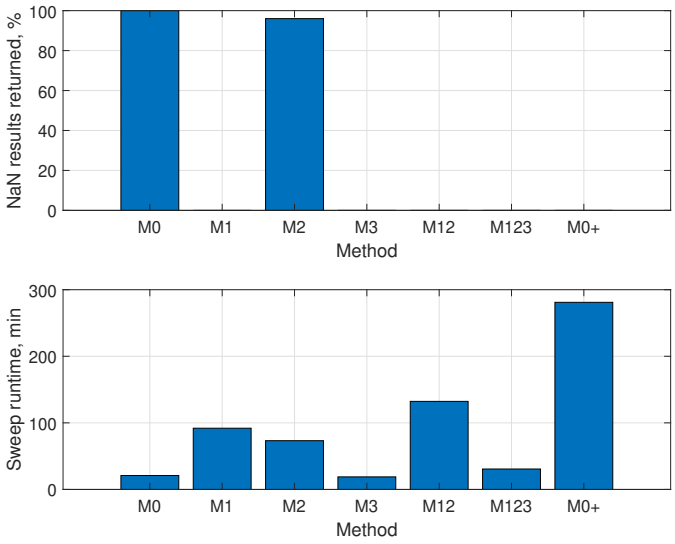

Fig. 7. Comparison of NaN results returned (top) and the runtime (bottom) of each proposed method.

VI. Discussion

Method 1 has proven effective in fully eliminating the convergence issues in the test circuit. The aggressive adaptation constants of 0,1 and 0,2 seem to be more effective in removing the NaN returns than more conservative value of 0,5. This is due to the re-simulation cost starting to outweigh the profits of not over-tightening the relative tolerance. Method 2 proved only partially effective and an increase in maximum iteration number of this method above 3 does not increase the effectiveness any further for the test case used. For this reason, a limit of three iterations has been deployed in the final algorithm. Method 3 significantly decreases the computational cost to about ten times lower, compared to fully tightened relative tolerance sweep without any of the proposed methods applied. However, it should still be noted that it depends on successful convergence of the first design point.

In the test circuit, convergence issues requiring relaxation of convergence criteria have not been encountered. This should not be an issue if starting relative tolerance is chosen relatively high, however it has to be noted that there is a risk of it still being too tight. In such a case solver adaptation method would be counter-effective, unless a modification implementing occasional relaxation attempts is deployed.

Conclusions

In this work, convergence issues present in power electronic simulations and their their origins were described. Three methods of mitigating the convergence issues were presented and evaluated in regards of computational cost and effectiveness in eliminating the NaN returns. The combination of proposed methods effectively eliminated convergence issues in the test circuit parameters sweeps and achieved multiple times lower computational cost. Only relative tolerance adaptation has been considered, in the future works a wider adaptation can be linked with circuit type recognition in order to achieve a better simulation robustness. In order to find out further limitations of proposed methods, more test cases can be considered in the future.

References

[1] M. Madbouly, M. Dessouky, M. Zakaria, R. A. Latif and A. Farid, "MATLAB - SPICE interface (MATSPICE) and its applications," Proceedings of the 12th IEEE International Conference on Fuzzy Systems (Cat. No.03CH37442), 2003, pp. 37-40, doi: 10.1109/ICM.2003.238301.

[2] F. Lannutti et al., "A new algorithm for convergence verification in circuit level simulations," 2014 10th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME), 2014, pp. 1-4.

[3] Z. Jin, "Study on Pseudo-transient Analysis Methods for Solving Nonlinear DC Circuits," Doctoral thesis, Graduate School of Information, Production and Systems, Waseda University, Tokyo, , 2015. Available: https://irdb.nii.ac.jp/en/00835/0002064716

[4] X. Wei et al. "BoA-PTA, A Bayesian Optimization Accelerated Error-Free SPICE Solver", 2021.

[5] B. W. Nelson et al., "Computational Efficiency Analysis of SiC MOSFET Models in SPICE: Dynamic Behavior," in IEEE Open Journal of Power Electronics, vol. 2, pp. 106-123, 2021, doi: 10.1109/OJPEL.2021.3056075.

[6] Z. Q. Shang, "The convergence problem in SPICE," IEE Colloquium on SPICE: Surviving Problems in Circuit Evaluation, 1993, pp. 10/1-10/5.

[7] Quarles, T. Linwood and A. Newton. "Analysis of performance and convergence issues for circuit simulation." , 1989. Available: https://apps.dtic.mil/sti/citations/ADA637166

[8] K. S. Kundert and I. H. Clifford, "Achieving accurate results with a circuit simulator," IEE Colloquium on SPICE: Surviving Problems in Circuit Evaluation, 1993, pp. 4/1-4/5.

[9] https://github.com/AAU-CoDE/Matlab-Spice-Interface

[10] Paul Wagner (2022). Fast Import of Compressed Binary .RAW Files Created with LTspice Circuit Simulator (https://www.mathworks.com/matlabcentral/fileexchange/23394-fast-import-of-compressed-binary-raw-files-created-with-ltspice-circuit-simulator), MATLAB Central File Exchange. Retrieved June 24, 2022.