New Approach for Enhancing Efficiency of Computer Aided Design in Circuit Simulation

David Černý and Josef Dobeš
Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Radio Engineering
Technická 2
166 27 Praha 6, Czech Republic
Emails: {cernyd1, dobes}@fel.cvut.cz

Abstract—All versions of SPICE can perform various kinds of analysis such as operating point, sensitivity, and transient analyses, and many others. The algorithms used for computations are based on an iterative numerical solution of nonlinear differential equations. This paper gives a simple overview about these algorithms, their principles, function and disadvantages. It introduces a novel approach and implementation of implicit and explicit algorithms for solving the differential equations in circuit design with the aim to improve the reliability and efficiency of the solution. Furthermore, it presents a comparison between the standard SPICE algorithm and the new developed ones.

I. INTRODUCTION

For computer-aided design (CAD) and analyzing the integrated circuits (ICs), a number of various programs were developed during the last four decades. There is no doubt that the program SPICE has become one world standard. First release of the program was formed between years 1967 and 1971 in the EECS Department of the University of California at Berkeley. Almost immediately (1975) the second version of simulator SPICE2 followed. SPICE2 came up with the new types of analysis, with improved and robust iteration algorithms, and also with new models of nonlinear devices. The accuracy and speed of the analyses were improved by the addition of time-step control mechanism and the new integration method.

A free redistribution of the program source under the Berkeley licence helped to popularization of SPICE. In the second half of the 1970s, Interactive SPICE (ISPICE) was introduced, the first commercially supported version. In following decade SPICE core was transferred into many others CAD programs as a tool for professional circuit simulation. For example, PSPICE (Cadence), SPECTRE (Cadence), HSPICE (Synopsys), ELDO (Mentor Graphics), SmartSPICE (Simucad), NgSpice, and many others. More on the history of the SPICE can be found in [1], [2].

II. TRANSIENT ANALYSIS IMPLEMENTED IN SPICE

Programs for computer-aided design of integrated circuits usually come with a variety of simulation processes and analyses, i.e., operating point (OP), sensitivity analysis, transient analysis, and many others. In this paper, we will focus mainly on the time-domain solution algorithm or more precisely on the transient analysis (.TRAN).

As it has been proven in many papers [3]–[7], .TRAN is the most problematic part of the analysis regarding accuracy and convergence of a solution. A computation of .TRAN is divided into a sequence of quasi-static evaluations. During this sequence, various algorithms and techniques must be applied to obtain correct solution such as Newton-Raphson (NR) iterative algorithm, implicit numerical integration method, Gaussian elimination and sparse matrix techniques. A simple overview of the simulation process in SPICE can be made from the following algorithm (a more rigorous description of SPICE algorithms can be found in the books of [8]–[10]):

- Initial computation of operating point.
- repeat {Time interval loop: \( T_{\text{min}} \to T_{\text{max}} \)}
  - repeat {Newton-Raphson iterative loop}
    - Linearize semiconductor device around operating point.
    - Load linear conductances in circuit matrix.
    - Solve linear equation.
    - if Convergence then
      - Increment time.
      - Break.
    - else
      - Define new trial OP.
    - end if
  - until Maximum number of iteration loops.
  - if not (Convergence) then
    - Error: Maximum number of iteration loops exceeded.
  - end if
- Discretize differential equations in time.
  - until End of time interval.

A. Integration Methods in SPICE

The .TRAN analysis solution process implemented in SPICE first solves DC operating point for all voltage nodes. This is not always necessary, sometimes DC solution can be automatically skipped (e.g., for oscillators). Then, the result is passed as an initial value for the first time step. Seldom, all devices are linear in the circuit. Therefore, simulator linearizes all nonlinear differential equations by iteration of NR iterative algorithm and implicit numerical integration (NI) method. General category of the polynomial integration method can
be defined by the equation
\[ x_{n+1} = \sum_{i=0}^{k} a_i x_{n-i} + \sum_{i=1}^{k} b_i x_{n-i}. \] 
(1)

If \( b_{-1} \) is zero, method is explicit, and if \( b_{-1} \) is nonzero, the method is implicit. The algorithm is a multi-step one if \( k > 1 \), that is, if more than one point from the past is needed to compute the following \( x_{n+1} \).

In the SPICE family programs, user can choose one from two different NI methods. The default one is trapezoidal (TRAPEZ) method, and the second one is Gear method (GEAR). Stability of the integration has been proved already and does not need to be presented here. More details about these methods can be found in the works by [8], [11], [12].

Both methods work as implicit NI, but explicit NI method should be added to enhance a robustness of the algorithm. A form of the TRAPEZ method, which is used in SPICE, can be simply derived from following equation
\[ \dot{x} = f(x, t). \] 
(2)
The general solution of (2) at time \( t_{n+1} \) is following
\[ x_{n+1} = x_n + \frac{1}{2} h(f(x_n, t_n) + f(x_{n+1}, t_{n+1})), \] 
(3)
from which we finally obtain
\[ \dot{x}_{n+1} = f(x_{n+1}, t_{n+1}) = -f(x_n, t_n) + \frac{2}{h}(x_{n+1} - x_n). \] 
(4)

Therefore, the TRAPEZ method is recognized as a second order method. When convergence test failed, SPICE changes the order of method to one, which is the simple implicit Euler method. A more complex description of the implemented algorithms can be found in [4]. An implementation of the GEAR algorithm in SPICE is done by the standard implicit formulation of this method, which can be found in various mathematical references [12].

B. Automatic Time Step Control

The automatic time step control (ATSC) is a part of NI algorithm, which chooses the size of step during the time domain analysis. The size of these steps may vary over many decades to improve accuracy or speed of the NI algorithm.

Let’s assume that we have a given interval by \( t_{\text{start}}, t_{\text{stop}} \) and \( t_{\text{step}} \), which denotes a beginning, an end and the step size of the NI method, respectively. When the transient analysis is started, SPICE takes this interval and evaluates number of time steps and step size. It is done the by following algorithm
\[ \text{if } (t_{\text{stop}} - t_{\text{start}}) / 50 > t_{\text{step}} \text{ then} \]
\[ t_{\text{max}} := t_{\text{step}} \]
\[ \text{else} \]
\[ t_{\text{max}} := t_{\text{step}} := (t_{\text{stop}} - t_{\text{start}}) / 50 \]
\[ \text{end if} \]
It should be pointed out that it computes transient analysis at least in 50 points insensibly of step size and interval bounds defined by the user.

Because the solution is evaluated by iteration algorithm, it may occur that the program could not be able to find a solution and the maximal number of iterations will be exceeded. In that case ATSC decreases the time step and change the order of NI to one.

C. Accuracy of Numerical Integration Method

Furthermore, it is necessary to check whether the solution is sufficiently close to the right solution. Every solution, which was computed by the numerical integration method, must be checked for its accuracy. It is done by the local truncation error algorithm (LTE), which is based on the upper bound \( E \), a maximal value of heuristically scaled solutions of \( x \) and \( \dot{x} \).

The solution of LTE is equal to upper bound of size of the next step and is computed by following inequality:
\[ h_{n+1} \leq \sqrt{\frac{6E}{\max(DD, \epsilon_n)}}. \] 
(5)

The equation above is of course unsuitable for numerical evaluation. Therefore, the SPICE implementation introduces the LTE equation in a better form for numerical computing:
\[ h_{n+1} = \sqrt{\frac{7 \cdot E}{\max(DD, \epsilon_n)}.} \] 
(6)
where \( DD \) is the divided difference of order three.

D. Time Selection Algorithm

Let’s assume that we have just obtained the solution \( x_n \) by the step size \( h_n \). The algorithm first computes size of next step \( h_{n+1} \) by LTE function. If the size of the next step is smaller than 90 percent of the actual step \( h_n \), the present result \( x_n \) is rejected and the solution is recomputed (from the previous solution \( x_{n-1} \)) by the smaller step size \( h_n = h_{n+1} \).

Computes \( h_{n+1} = f(LTE) \).
\[ \text{if } h_{n+1} < 0.9 \cdot h_n \text{ then} \]
\[ \text{Reject } x_n. \]
\[ \text{Set } h_n = h_{n+1}. \]
\[ \text{Recompute new } x_n. \]
\[ \text{else} \]
\[ \text{Accept } x_n. \]
\[ \text{Increase order of the method.} \]
\[ \text{end if} \]

The order of the method is increased if and only if the new value of \( x_n \) is accepted.

III. NEW NUMERICAL INTEGRATION APPROACH

As it was mentioned above, the most SPICE programs support two integration methods, TRAPEZ and GEAR up to the order six. Each of the method comes up with disadvantages, which can lead in a worst-case to non-convergence or wrong solution. To avoid this, an additional integration approach was developed and implemented into the SPICE family simulator. For our purposes, we decided to use an open-source mixed-level circuit simulator NgSpice, which is based on the three open source software packages: Spice3f5, Cider1b1 and Xspice.
A. Numerical Integration With Predictor

The NgSpice simulator includes various methods for evaluation. In our simulation, we have used two methods as a base for checking accuracy of the simulation. We have used implicit TRAPEZ and GEAR method without predictor, and TRAPEZ and GEAR method with appropriate explicit predictor. These two methods are already implemented in the NgSpice, and therefore can be assumed as correct ones.

B. Numerical Integration With Modified Predictor-Corrector

Our new approach is letting the simulator to obtain a solution by both methods together, TRAPEZ and GEAR, and to use the first solution as a leading one, and the second as a supporting of the first one. The several heuristic approaches can be performed to obtain the final solution. We have used the simplest one in our simulations, an arithmetic mean:

\[ x_n = \frac{x_n^* + x_n^{**}}{2}, \quad (7) \]

where \( x_n^* \) and \( x_n^{**} \) are the vectors of node solutions for the GEAR and TRAPEZ method at the time \( t_n \) respectively. It can be shown that this may lead to the less accurate solution, compared to one, obtained by using the standard approach on some circumstances. For example, in the case when both, TRAPEZ and GEAR methods are close to the solution from a same side. Regardless, using this approach, we can suppress a solution in some situations, where particular method has some divergence problems.

We implemented this approach for both above methods. One important thing must be noticed. This algorithm is bypassed when an order of the method increases to higher value.

C. Numerical Integration With New Newton Predictor

We also implemented a new method utilizing divided differences to the NgSpice. From a theory, it is well-known that the Newton interpolation polynomial is a modification of the Lagrange interpolation polynomial with respect to optimize number of evaluations, when a new value is added. This is a very convenient for the numerical algorithm not only for save of time of computation, but also because of better memory handling.

Our implementation of the explicit Newton predictor (NEWTONPRED) method is composed by divided differences. Let’s assume that we know \( k \) steps of NI \( h_n \cdots h_{n-k} \). Then, we also know \( k \)-th order of Newton interpolation polynomial from which we can derive a basis for NEWTONPRED by the formula:

\[ \Psi_k(t_{n+1}) = \prod_{i=0}^{k-1} (t_{n+1} - t_{n-i}), \quad (8) \]

where \( k \) now denote the order of the method and could vary from \( k = 1, \ldots, n \), where \( n \) is maximal order of the predictor. In our case, the maximal order was 6, same like in the GEAR method. \( \Psi_0 = 1 \) is defined by the convention.

Zero-order difference is calculated by the simple formula:

\[ \alpha_{n-k} = f(t_{n-k}) = f[t_{n-k}] = x_{n-k}, \quad (9) \]

actually it is just evaluating the same value. The first order is then defined by

\[ \alpha_{n-k+1} = \frac{f(t_{n-k+1}) - f(t_{n-k})}{t_{n-k} - t_{n-k+1}} = f[t_{n-k}, t_{n-k+1}], \quad (10) \]

Using mathematical induction, we will get the solution of the k-th divide difference:

\[ \alpha_n = \frac{f(t_{n-1}, \ldots, t_{n-k}) - f(t_n, \ldots, t_{n-k-1})}{t_{n-k} - t_n}, \quad (11) \]

Finally we can write down the explicit Newton’s integration method of degree \( k \) as

\[ x_{n+1} = f[t_{n-k}] + \sum_{i=1}^{k} f[t_{n-k}, \ldots, t_{n-k+i}] \Psi_i(t_{n+1}). \quad (12) \]

It is important to mention that our implementation is using the implicit GEAR corrector together with explicit NEWTONPRED.

IV. Analyzed Circuits

For simulation, we were picked three different circuits, which are well described in various literature, and can be easily gathered, for example, from the Web. In a following sections, we listed only parts of the SPICE sources, which were changed for a purpose of the simulation.

A. CMOS Multiplier

This circuit is the multiplier which origins from [13]. It is a low-voltage low-power CMOS four-quadrant microwave mixer. It uses semi-empirical models of transistors of level 3.

`.MODEL MN NMOS LEVEL=3 UO=460.5 TOX=1.0E-8 +TPG=1 VTO=0.62 JS=1.08E-6 Xl=0.15I +RS=417 RSH=2.73 LD=0.04U VMAX=130E3 +NSUB=1.71E17 PB=0.761 ETA=0.00 THETA=0.129 +PHI=0.905 GAMMA=0.69 KAPPA=0.10 CJ=76.4E-5 +MJ=0.357 CJSW=5.6E-10 MSW=0.302 +CGSO=1.38E-10 CGDO=1.38E-10 CGBO=3.45E-10 +KF=3.07E-28 AF=1

`.MODEL MP PMOS LEVEL=3 UO=100 TOX=1.0E-8 +TPG=1 VTO=0.58 JS=0.38E-6 Xl=0.10U RS=886 +RSH=1.81 LD=0.03U VMAX=113E3 NSUB=2.08E17 +PB=0.911 ETA=0.00 THETA=0.120 PH1=905 +GAMMA=0.76 KAPPA=2 CJ=85E-5 MJ=0.429 +CJSW=6.67E-10 MSW=0.631 CGSO=1.38E-10 +KF=1.08E-29 AF=1

.TRAN 100P 40N 0 20P`
C. MOS Amplifier

This circuit is a MOS transistor amplifier. It origins from the NgSpice testing repository and can be downloaded with actual version of NgSpice21. It is modeled by MOS transistors with substrate doping NSUB, gate-source overlap capacitor CGSO, gate-drain overlap cap CGDO, surface state density NSS and surface mobility MO, oxide thickness TOX, lateral diffusion LD, and critical field exponent for mobility degradation UEXP.

```
.OPTIONS ABSTOL=10n VNTOL=10n NOACCTO
.TRAN 0.1us 10us
.MODEL m NMOS NSUB=2.2e15 UO=575 UCITR=49k
+UEXP=0.1 TOX=0.11u J3=2.95u LEVEL=2
+LEVEL=2 CGSO=1.5 CGDO=1.5 CBS=6.5f LD=2.4485u NSS=3.2e10
+KP=2e-5 PHI=0.6
```

The statistics of the analyses of all the three circuits are shown in Tables I through III. The numbers in all three tables showed that the TRAPEZ method (with or without predictor) is clearly the best one in the first task, the TRAPEZ method with the GEAR predictor is the best one in the second task, and the GEAR method with the GEAR predictor is the best one in the third task. Therefore, the suggested inclusion of the predictor to the method always improves the properties of the analysis. It is also obvious that a final decision, which combination is the best needs more circuits to testing.

V. CONCLUSION

Several methods for enhancing the transient analysis have been suggested. It seems that the best integration method is TRAPEZ by comparing the obtained results. It can be true until the fact that although this method computed a final solution with the lowest number of iteration steps, it was unable to find any solution when some non-convergence occurred, in contrast to the other methods.

A combination of two algorithms for numerical integration seems to be promising solution. The results of simulation done by GEAR and TRAPEZ method together give a good trade off between number of non-convergences and simulation accuracy.

GEAR with predictor did not provide satisfactory results and finished with same values as explicit GEAR. It was due to the fact that all results from GEAR predictor were very close to a values that would be evaluated by standard GEAR corrector. Thus, the implicit GEAR corrector was able to correct a solution in all cases likewise it would be supplemented by explicit GEAR predictor. It should imply that GEAR predictor must be supported by implicit Newton corrector.

ACKNOWLEDGMENTS

This paper was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS10/286/OKH3/3T/13, and by the Grant Agency of the Czech Republic, grant No. P102/10/1614.

REFERENCES