Reliable and Efficient Procedure for Steady-State Analysis of Nonautonomous and Autonomous Systems

Josef DOBEŠ[1], Viera BIOLKOVÁ[2]

[1]Department of Radio Engineering, Czech Technical University in Prague, Technická 2, 166 27 Praha 6, Czech Republic
[2]Department of Radio Electronics, Brno University of Technology, Purkyňova 118, 612 00 Brno, Czech Republic

dobes@fel.cvut.cz, biolkova@feec.vutbr.cz

Abstract. *The majority of contemporary design tools do not still contain steady-state algorithms, especially for the autonomous systems. This is mainly caused by insufficient accuracy of the algorithm for numerical integration, but also by unreliable steady-state algorithms themselves. Therefore, in the paper, a very stable and efficient procedure for the numerical integration of nonlinear differential-algebraic systems is defined first. Afterwards, two improved methods are defined for finding the steady state, which use this integration algorithm in their iteration loops. The first is based on the idea of extrapolation, and the second utilizes nonstandard time-domain sensitivity analysis. The two steady-state algorithms are compared by analyses of a rectifier and a C-class amplifier, and the extrapolation algorithm is primarily selected as a more reliable alternative. Finally, the method based on the extrapolation naturally cooperating with the algorithm for solving the differential-algebraic systems is thoroughly tested on various electronic circuits: and Colpitts oscillators, fragment of a large bipolar logical circuit, feedback and distributed microwave oscillators, and power amplifier. The results confirm that the extrapolation method is faster than a classical plain numerical integration, especially for larger circuits with complicated transients.*

Keywords

Numerical integration, steady-state algorithm, period, extrapolation, epsilon-algorithm, sensitivity analysis.

1. Introduction

Although the mathematical background for finding the steady state is available for several decades, the majority of contemporary software tools do not still contain reliable procedures for this type of analysis, especially for autonomous circuits. It is particularly caused by insufficient precision of solving the nonlinear systems of differential-algebraic equations that always serve as a core of a steady-state algorithm of any kind. With such an imprecision, it is impossible to detect a period because the solution does not return to the same value after a period-long numerical integration. Moreover, identification of the period is even more difficult for the autonomous circuits in such cases.

Those problems are reflected by the status of the significant software tools of this area. The majority of the PSpice programs do not contain a steady-state algorithm at all [1]. In the software tools that are specialized in the radio engineering, a semiautomatic method is only implemented [2]. In the contemporary rapidly developing programs such as [3], new procedures for the steady-state analysis have been implemented recently, and their code is still developed.

A status of the methods for solving the differential-algebraic systems that are implemented in the majority of the fundamental simulators is reflected in [4]. However, even the standard methods are continuously developed, and the updates are published in novel editions of the classical books such as [5]. Moreover, radically new methods have arisen recently that are dedicated to the still remaining problem with the stability of the algorithms, which are represented by [6].

2. Core of the Procedure – Algorithm for Numerical Solution of Nonlinear Differential-Algebraic Systems

2.1 Transient Analysis

A system of nonlinear algebraic-differential equations of a circuit is generally defined in the implicit form [4], [5], [6]

$$\boldsymbol{f}\left[\,\boldsymbol{x}(t),\dot{\boldsymbol{x}}(t),t\,\right]=\boldsymbol{0}. \tag{1}$$

Let us now assume that the first n steps of the numerical integration of (1) have been finished. Let us mark $\boldsymbol{x}(t_n)$ by $\boldsymbol{x}_n$ and define the backward scaled differences

$$\begin{aligned}&\delta^{(0)}\boldsymbol{x}_n=\boldsymbol{x}_n,\\&\delta^{(k)}\boldsymbol{x}_n=\delta^{(k-1)}\boldsymbol{x}_n-\alpha_n^{(k-1)}\,\delta^{(k-1)}\boldsymbol{x}_{n-1},\\&\qquad\qquad n=1,\ldots,\quad k=1,\ldots,k_n+2\end{aligned} \tag{2}$$

where k_n is the order of the polynomial interpolation used in the last integration step, and the $\alpha_n^{(\dots)}$ multipliers are also determined in the recurrent way

$$\begin{aligned}&\alpha_n^{(0)} = 1,\\ &\alpha_n^{(k)} = \alpha_n^{(k-1)} \frac{t_n - t_{n-k}}{t_{n-1} - t_{n-1-k}},\ k = 1, \dots, k_n + 1.\end{aligned} \quad (3)$$

A prediction of circuit variables (predictor) for the next chosen time (i.e., for t_{n+1}) marked by $\boldsymbol{x}_{n+1}^{(0)}$ is determined by the polynomial extrapolation using the backward scaled differences (2)

$$\boldsymbol{x}_{n+1}^{(0)} = \sum_{k=0}^{k_{n+1}} \alpha_{n+1}^{(k)}\, \delta^{(k)} \boldsymbol{x}_n, \quad (4)$$

which is a more sophisticated form of the Newton interpolation polynomial – a comprehensive proof can be found in Theorem 1 in [7].

By differentiating (4) with respect to t_{n+1}, the predictor of time derivatives can also be expressed in a similar way

$$\dot{\boldsymbol{x}}_{n+1}^{(0)} = \sum_{i=0}^{k_{n+1}} \beta_{n+1}^{(i)} \delta^{(i)} \boldsymbol{x}_n \quad (5)$$

where the β multipliers may simply be derived from the recurrent form (3) in terms of the α ones

$$\begin{aligned}&\beta_n^{(0)} = 0,\\ &\beta_n^{(i)} = \frac{\alpha_n^{(i-1)} + (t_n - t_{n-i})\beta_n^{(i-1)}}{t_{n-1} - t_{n-1-i}},\ i = 1, \dots, k_n.\end{aligned} \quad (6)$$

(Using (6) in (5) needs replacing the subscript n by $n+1$, of course.)

The correction of circuit variables $\boldsymbol{x}_{n+1} := \boldsymbol{x}_{n+1}^{(j_{\max n+1})}$ (corrector) for t_{n+1} is determined by the iterations of the modified Newton-Raphson method

$$\left[\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)_{n+1}^{(j)} + \gamma_{n+1}\left(\frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{x}}}\right)_{n+1}^{(j)}\right] \Delta \boldsymbol{x}_{n+1}^{(j)} = -\boldsymbol{f}_{n+1}^{(j)},$$
$$n = 0, \dots, \quad j = 0, \dots, j_{\max n+1}, \quad (7)$$

i.e., by repeated solving the linear system (7) with the application of the γ_{n+1} factor determined in the following way (more details can be found in the proof of Theorem 2 in [7]):

$$\begin{aligned}\dot{\boldsymbol{x}}_{n+1}^{(j)} &= \lim_{t_{n+2} \to t_{n+1}} \frac{\boldsymbol{x}_{n+2}^{(j)} - \boldsymbol{x}_{n+1}}{t_{n+2} - t_{n+1}}\\ &= \sum_{i=1}^{k_{n+1}} \frac{1}{t_{n+1} - t_{n+1-i}}\, \delta^{(i)} \boldsymbol{x}_{n+1}^{(j)} \quad (8)\\ \Rightarrow \gamma_{n+1} &= \sum_{i=1}^{k_{n+1}} \frac{1}{t_{n+1} - t_{n+1-i}},\end{aligned}$$

which gives the classical formula $\gamma_{n+1} = 1/(t_{n+1} - t_n) = 1/\Delta t_{n+1}$ if the first-order method (i.e., Euler's one) is used.

After resolving the linear system (7), the vectors $\boldsymbol{x}_{n+1}^{(\dots)}$ and $\dot{\boldsymbol{x}}_{n+1}^{(\dots)}$ are modified by the following standard equations:

$$\boldsymbol{x}_{n+1}^{(j+1)} = \boldsymbol{x}_{n+1}^{(j)} + \Delta \boldsymbol{x}_{n+1}^{(j)}, \quad \dot{\boldsymbol{x}}_{n+1}^{(j+1)} = \dot{\boldsymbol{x}}_{n+1}^{(j)} + \gamma_{n+1} \Delta \boldsymbol{x}_{n+1}^{(j)}, \quad (9)$$

which completes the $(j+1)^{\text{th}}$ iteration of the $(n+1)^{\text{th}}$ step.

The stability of the corrector iterations can be improved using the logarithmic damping of divergence, which consists in replacing each element $\Delta x_{n+1_i}^{(j)}$ of the vector $\Delta \boldsymbol{x}_{n+1}^{(j)}$ by the following sophisticated formula before executing (9)[1]:

$$\begin{aligned}\Delta x_{n+1_i}^{(j)} &:= \operatorname{sgn}\left(\Delta x_{n+1_i}^{(j)}\right) \frac{\left|x_{n+1_i}^{(j)}\right| + x_{\text{null}i}}{d_i}\\ &\times \ln\left(1 + d_i \frac{\left|\Delta x_{n+1_i}^{(j)}\right|}{\left|x_{n+1_i}^{(j)}\right| + x_{\text{null}_i}}\right),\ i = 1, \dots, n_x. \quad (10)\end{aligned}$$

The parameters d_i are appropriate damping factors (which are mostly the same $\forall\ i$), the parameters $x_{\text{null}i} \to 0+$ prevent from possible division by zero, and n_x is the dimension of the vector $\boldsymbol{x}$.

A new length of the integration step Δt_{n+2} and a new order of the polynomial interpolation k_{n+2} are to be chosen after the convergence of the corrector (7) and (9). Firstly, an estimation of the interpolation error in the last step must be determined. In general, the absolute truncation error for the k_{n+1} order caused by the derivatives approximation (8) may be written as

$$e_{n+1_i} = \frac{\Delta t_{n+1}}{t_{n+1} - t_{n-k_{n+1}}}\, \delta^{(k_{n+1}+1)} x_{n+1_i} \quad (11)$$

for any element x_{n+1_i} of the vector $\boldsymbol{x}_{n+1}$. The relation (11) for the element e_{n+1_i} of the absolute truncation error vector $\boldsymbol{e}_{n+1}$ must be modified in the following way:

- the absolute errors must be replaced with relative ones (for any error to be comparable with another one and with an algorithm control parameter),
- the fraction $\Delta t_{n+1}/(t_{n+1} - t_{n-k_{n+1}})$ will be omitted (to give a preference to simpler and more stable lower interpolation orders).

Consequently, the relative truncation error acquires the simple following form[2]

$$\begin{aligned}\varepsilon_{n+1} &= \max_{\forall x_{n+1_i} \in \boldsymbol{x}_{n+1}} \frac{\left|\delta^{(k_{n+1}+1)} x_{n+1_i}\right|}{\left|x_{n+1_i}\right| + x_{\text{null}_i}}\\ &= \max_{\forall x_{n+1_i} \in \boldsymbol{x}_{n+1}} \frac{\left|x_{n+1_i} - x_{n+1_i}^{(0)}\right|}{\left|x_{n+1_i}\right| + x_{\text{null}_i}}\end{aligned} \quad (12)$$

[1] The idea of the logarithmic damping is based on the Maclaurin series $\ln(1+x) = x - x^2/2 + x^3/3 - + \dots \approx x$ for $x \to 0$.

[2] Similarly to that in the logarithmic damping, $|x_{n+1_i}^{(j)}| + x_{\text{null}_i}$ is used instead of $|x_{n+1_i}^{(j)}|$ to avoid possible zero division.

(see the final equality in the proof of Theorem 1 in [7], and (4) as a simple derivation of the second equality in (12)). Therefore, the truncation error can be checked by the difference between the corrector and predictor values — the step may be rejected and halved even after the *first* iteration of the corrector if the truncation error seems too big.

Secondly, the new step and order are determined by means of the calculated error (12). Generally, the truncation errors of the i^{th} interpolation order can be determined by the formulae (consider m^{th} components of $\boldsymbol{e}_{n+1|n+2}$ with the greatest error)

$$e^{(i)}_{n+1\,m} = \text{const.}^{(i)}_{n+1\,m} \left(\frac{\mathrm{d}^{i+1}x_m}{\mathrm{d}t^{i+1}}\right)_{n+1} \Delta t^{i+1}_{n+1},$$
$$e^{(i)}_{n+2\,m} = \text{const.}^{(i)}_{n+2\,m} \left(\frac{\mathrm{d}^{i+1}x_m}{\mathrm{d}t^{i+1}}\right)_{n+2} \Delta t^{i+1}_{n+2}.$$

The new step estimate is based on the assumption of a similarity of neighboring steps

$$\text{const.}^{(i)}_{n+1\,m} \approx \text{const.}^{(i)}_{n+2\,m},$$
$$\left(\frac{\mathrm{d}^{i+1}x_m}{\mathrm{d}t^{i+1}}\right)_{n+1} \approx \left(\frac{\mathrm{d}^{i+1}x_m}{\mathrm{d}t^{i+1}}\right)_{n+2},$$

which gives the relation

$$\frac{e^{(i)}_{n+2\,m}}{e^{(i)}_{n+1\,m}} \approx \left(\frac{\Delta t_{n+2}}{\Delta t_{n+1}}\right)^{i+1} \triangleq \frac{\varepsilon^{(i)}_{n+2}}{\varepsilon^{(i)}_{n+1}}.$$

The error in the following integration step is to be equal to ε — prescribed relative truncation tolerance, i.e. (Δt_{n+1} is already known when $\Delta t^{(i)}_{n+2}$ are to be compared $\forall$ possible interpolation orders i)

$$\Delta t^{(i)}_{n+2} = \Delta t_{n+1} \sqrt[i+1]{\frac{\varepsilon}{\varepsilon^{(i)}_{n+1}}},\ i = 1, \ldots, k_{n+1}+1 \quad (13)$$

where all the possible truncation relative errors $\varepsilon^{(i)}_{n+1}$ are computed directly by the $\delta^{(i+1)}\boldsymbol{x}_{n+1}$ (this is why the differences (2) are defined up to the $k_n + 2$ order — so that the order of the polynomial interpolation can sequentially increase). However, the step increase is limited due to the stability conditions, especially for the higher orders of the interpolation — the proof for that can be found in the stability comparisons of the basic implicit integration formulae in [8]. Therefore, the relation (13) must be modified by a semiempirical factor (for stiff systems, it is some limitation)

$$\Delta t^{(i)}_{n+2} = \begin{cases} \Delta t_{n+1} \sqrt[i+1]{\dfrac{\varepsilon}{\varepsilon^{(i)}_{n+1}}} & \text{for } \dfrac{\varepsilon}{\varepsilon^{(i)}_{n+1}} < 4, \\ \Delta t_{n+1} \sqrt[i+1]{4} & \text{otherwise,} \end{cases}$$
$$i = 1, \ldots, k_{n+1}+1 \quad (14)$$

where the limiting factor $\sqrt[i+1]{4}$ can be theoretically derived under special circumstances only: in [8], a proof is created for a scheme with the fixed step size. However, this factor has also been proven by thousands of practical analyses with schemes with largely variable step size, as well. In conclusion, the new k_{n+2} order ($k_{n+2} \in \{1, \ldots, k_{n+1}+1\}$) is chosen whose step determined by (14) is the longest one.

The whole method fulfills the A-stability condition [6].

2.2 Operating Point Analysis

The operating point analysis is performed using the static variant of (1)

$$\boldsymbol{f}(\boldsymbol{x}_0, \boldsymbol{0}, t_0) = \boldsymbol{f}_0(\boldsymbol{x}_0) = \boldsymbol{0}, \quad (15)$$

which is solved by the static variants of (7) and (9):

$$\left(\frac{\partial \boldsymbol{f}_0}{\partial \boldsymbol{x}_0}\right)^{(j)} \Delta \boldsymbol{x}^{(j)}_0 = -\boldsymbol{f}^{(j)}_0, \quad \boldsymbol{x}^{(j+1)}_0 = \boldsymbol{x}^{(j)}_0 + \Delta \boldsymbol{x}^{(j)}_0,$$
$$j = 0, \ldots, j_{\max_0}. \quad (16)$$

2.3 Novel Methods for Suppressing Divergence

However, the convergence in the operating point analysis is often more problematic than that in the transient analysis (in the transient analysis, the final values of the previous time step serve as a good estimation for the following time step). To avoid possible divergence in *both static and dynamic* domains, a new control procedure has been developed for handling the differences $\Delta \boldsymbol{x}^{(j)}_{n+1}$ during each iteration in the following way:

```
if j = 0 then
    x* := x^(0)_{n+1},
    ẋ* := ẋ^(0)_{n+1},
    Δx* := Δx^(0)_{n+1},
    f* := f^(0)_{n+1}, and the (first) iteration is accepted,
else
    if ‖f^(j)_{n+1}‖ < 1 then
        x* := x^(j)_{n+1},
        ẋ* := ẋ^(j)_{n+1},
        Δx* := Δx^(j)_{n+1},
        f* := f^(j)_{n+1}, and the iteration is accepted,
    else
        Δx* := Δx*/2,
        x^(j)_{n+1} := x*,
        ẋ^(j)_{n+1} := ẋ*,
        Δx^(j)_{n+1} := Δx*, and the iteration is rejected,
```
(17)

where the Euclidian norm $\|\boldsymbol{f}_{n+1}^{(j)}\|$ has been confirmed in the two of possible four forms:

$$\|\boldsymbol{f}_{n+1}^{(j)}\| = \begin{cases} \dfrac{1}{n_x}\displaystyle\sum_{i=1}^{n_x}\frac{|f_{n+1_i}^{(j)}|}{|f_i^*|+f_{\mathrm{null}_i}} \text{ (the firstly used form [9]),} \\ \sqrt{\dfrac{1}{n_x}\displaystyle\sum_{i=1}^{n_x}\left(\frac{f_{n+1_i}^{(j)}}{|f_i^*|+f_{\mathrm{null}_i}}\right)^2} \text{ (a novel form [7]).} \end{cases} \tag{18}$$

Note that in [7], two additional forms have been tested. However, from the totally four tested forms, only the two ones shown in (18) have been approved as sufficiently reliable. The modified method checks the function values (i.e., each element $f_{n+1_i}^{(j)}$ of $\boldsymbol{f}_{n+1}^{(j)}$) after each iteration. The basic idea of handling the differences $\Delta\boldsymbol{x}_{n+1}^{(j)}$ in accordance with (17) and (18) relates to the fundamental property of the Newton-Raphson method. If the function values do not decrease, then the differences are halved and the iteration is repeated. The process of halving continues until the function values decrease. It is *doubtless* that the occurrence of the decrease will be found, and therefore, the algorithm does *not* even contain a check for a potential infinite loop. As a result, only such $\Delta\boldsymbol{x}_{n+1}^{(j)}$ is used for updating the vector $\boldsymbol{x}_{n+1}^{(j)}$ that ensures decreasing the function values according to the criteria (18). The parameters $f_{\mathrm{null}_i} \to 0+$ prevent from potential division by zero, and $\boldsymbol{x}^*$, $\dot{\boldsymbol{x}}^*$, $\Delta\boldsymbol{x}^*$, and $\boldsymbol{f}^*$ are auxiliary vectors. The procedure (17) can be considered a generalized and improved form of the static procedure [10, p. 149] for the dynamic case. (In substance, this method is a variant of the globally convergent ones.)

3. Description of the Two Steady-State Algorithms

3.1 Procedure Built on the ϵ-Algorithm

The problem of computing the periodic steady state can be simply formulated as a solution of the nonlinear symbolic equation [11]

$$\boldsymbol{x}_{\mathrm{ss}} = \mathcal{I}\left(\boldsymbol{x}_{\mathrm{ss}}, t_0, t_0 + T_{\mathrm{ss}}\right) \tag{19}$$

where $\mathcal{I}\left(\boldsymbol{x}_{\mathrm{ic}}, t_0, t_0 + T_{\mathrm{int}}\right)$ marks the solution vector after the implicit numerical integration of (1) with an initial condition $\boldsymbol{x}_{\mathrm{ic}}$ on the interval T_{int}.

Instead of usual (and often very long) numerical integration of (1), much shorter one is performed. Samples of the solution are immediately collected after each of the periods (which must be determined for autonomous circuits). The samples become input for the scalar ϵ-algorithm [12], [13] which is able to estimate a state of the system in future. The outcome of the ϵ-algorithm becomes a new initial condition for the system (1) and the process is repeated until the steady-state is detected.

The number of periods needed for the extrapolation loop depends on the number of slowly decaying transients. This number can be reduced by a low-pass filtering performed by numerical integration of a suitable length

$$\boldsymbol{x}_j^{(0)} := \boldsymbol{x}_j\left(t_0 + \Delta t_{\mathrm{extpol}}\right) = \int_{t_0}^{t_0+\Delta t_{\mathrm{extpol}}} \mathcal{F}\left[\,\boldsymbol{x}(t), t\,\right]\mathrm{d}t, \quad j = 1, \ldots, j_{\max} \tag{20}$$

where j marks an iteration of the ϵ-algorithm, and $\dot{\boldsymbol{x}}(t) = \mathcal{F}\left[\,\boldsymbol{x}(t), t\,\right]$ indicates a vector of functions to be integrated in the numerical way. This is only a symbolic notation because the original system (1) is always implicit and cannot be transferred to an explicit form. The parameter $\Delta t_{\mathrm{extpol}}$ is chosen so that the rapidly decaying transients are to be insignificant at $t_0 + \Delta t_{\mathrm{extpol}}$. After that low-pass filtering, the extrapolation is assumed to be only concerned with the slowly decaying transients and the order of the extrapolation algorithm marked below by k_{extpol} can be decreased correspondingly. Note that the algorithm is relatively insensitive to chosen values $\Delta t_{\mathrm{extpol}}$ and k_{extpol}, which will be demonstrated on practical examples.

The vector $\boldsymbol{x}_j^{(0)}$ represents the first sample for the extrapolation algorithm. Entire sequence of the samples is also obtained by the implicit numerical integration of (1)

$$\boldsymbol{x}_j^{(k)} := \boldsymbol{x}_j\left(t_0 + \Delta t_{\mathrm{extpol}} + \textstyle\sum_{i=1}^{k} T_j^{(i)}\right) = \int_{t_0+\Delta t_{\mathrm{extpol}}}^{t_0+\Delta t_{\mathrm{extpol}}+\sum_{i=1}^{k} T_j^{(i)}} \mathcal{F}\left[\,\boldsymbol{x}(t), t\,\right]\mathrm{d}t,\ k = 1, \ldots, 2k_{\mathrm{extpol}}, \quad j = 1, \ldots, j_{\max} \tag{21}$$

where $T_j^{(i)}$ are (generally unequal) periods that must be detected for the autonomous circuits during the integration using (27). (They are known for the nonautonomous circuits.)

When the sampling is finished after the integration, the scalar ϵ-algorithm starts after the initialization (note that due to many other subscripts and superscripts necessary for the exact definition of the ϵ-algorithm, the upper-left superscript is used for marking an element of a vector in this subsection)

$$\begin{aligned} {}^{i}\epsilon_{-1}^{(k)} &:= 0, \qquad k = 1, \ldots, 2k_{\mathrm{extpol}}, \\ {}^{i}\epsilon_{0}^{(k)} &:= {}^{i}x_j^{(k)}, \quad k = 0, \ldots, 2k_{\mathrm{extpol}}, \end{aligned} \quad i = 1, \ldots, n_x \tag{22}$$

by the extrapolation formulae

$${}^{i}\epsilon_{m+1}^{(k)} := {}^{i}\epsilon_{m-1}^{(k+1)} + \frac{1}{{}^{i}\epsilon_{m}^{(k+1)} - {}^{i}\epsilon_{m}^{(k)}}, \quad m = 0, \ldots, 2k_{\mathrm{extpol}} - 1,\ k = 0, \ldots, 2k_{\mathrm{extpol}} - 1 - m \tag{23}$$

for all elements ${}^{i}\epsilon_{m+1}^{(k)},\ i = 1, \ldots, n_x$ of the vector $\boldsymbol{\epsilon}_{m+1}^{(k)}$. The outcome of the ϵ-algorithm becomes the new initial condition of the system (1), i.e.,

$$\boldsymbol{x}_{j+1}\left(t_0\right) := \boldsymbol{\epsilon}_{2k_{\mathrm{extpol}}}^{(0)} \tag{24}$$

and the procedure is repeated until convergence is detected. For breaking the iterations, an algorithm parameter $\epsilon_{\mathrm{extpol}}$ — allowable extrapolation error — is used in the following way:

$$\texttt{if} \quad \max_{i=1,\ldots,n_x} \frac{\left|{}^{i}x_j^{(k)} - {}^{i}x_j^{(k-1)}\right|}{\left|{}^{i}x_j^{(k)}\right| + {}^{i}x_{\mathrm{null}}} \leqq \epsilon_{\mathrm{extpol}} \quad \texttt{then}$$
$$\boldsymbol{x}_{\mathrm{ss}} := \boldsymbol{x}_j^{(k)},$$
$$k \in \langle 1, \ldots, 2k_{\mathrm{extpol}} \rangle. \quad (25)$$

For the nonautonomous circuits, breaking the iterations of the ϵ-algorithm by (25) is possible only at forced periods. For the autonomous circuits, the unknown periods are sequentially detected solving (27), and when the ϵ-algorithm (and the integration algorithm, too) is stopped by (25), just the last determined period is considered to be the final one.

The evolution of the ϵ-algorithm (22) and (23) can be represented by the following diagram (the values in the corners of the triangles create new values sequentially in the directions of the arrows):

$$\begin{array}{ccccccc}
 & & {}^{i}\epsilon_0^{(0)} & & & & \\
 & {}^{i}\epsilon_{-1}^{(1)} & \triangleleft & \rightarrow & {}^{i}\epsilon_1^{(0)} & & \\
 & & {}^{i}\epsilon_0^{(1)} & \triangleleft & \rightarrow & {}^{i}\epsilon_2^{(0)} & \\
 & {}^{i}\epsilon_{-1}^{(2)} & \triangleleft & \rightarrow & {}^{i}\epsilon_1^{(1)} & & \\
 & & {}^{i}\epsilon_0^{(2)} & & & \cdots & \\
 & \cdots & & & \cdots & & \\
 & & \cdots & & & &
\end{array} \quad (26)$$

An advantage of the above algorithm is a possibility of its application on the autonomous circuits. The detection of unknown periods is performed by searching the intersections of suitably chosen $i_{\mathrm{fix}}^{\mathrm{th}}$ element of the vector of circuit variables $\boldsymbol{x}_j$ with a suitably selected level x_{fix}, i.e., by finding t_{per} that fulfills the condition

$${}^{i_{\mathrm{fix}}}x_j(t_{\mathrm{per}}) - x_{\mathrm{fix}} = 0, \; i_{\mathrm{fix}} \in \langle 1, n_x \rangle. \quad (27)$$

The equation (27) can be solved in the similar way as that in the operating point analysis (16) by the Newton-Raphson method

$${}^{i_{\mathrm{fix}}}\dot{x}_j(t_{\mathrm{per}}^{(\ell)})\Delta t_{\mathrm{per}}^{(\ell)} = x_{\mathrm{fix}} - {}^{i_{\mathrm{fix}}}x_j(t_{\mathrm{per}}^{(\ell)}),$$
$$t_{\mathrm{per}}^{(\ell+1)} = t_{\mathrm{per}}^{(\ell)} + \Delta t_{\mathrm{per}}^{(\ell)}, \quad \ell = 1, \ldots, \ell_{\max}.$$

Since the integration algorithm can interpolate both output values ($\boldsymbol{x}_j$ by (4)) and the derivatives of the output values with respect to time ($\dot{\boldsymbol{x}}_j$ by (5)), a software arrangement of such procedure is straightforward.

3.2 Procedure Built on the Sensitivity Analysis

Let the symbol notation be the same as that in the previous subsection, and an equation is defined for an error vector $\boldsymbol{\varepsilon}$ in the classical way [11]

$$\boldsymbol{\varepsilon}\left(\boldsymbol{x}_{\mathrm{ic}}, t_0, t_0 + T_{\mathrm{int}}\right) = \mathcal{I}\left(\boldsymbol{x}_{\mathrm{ic}}, t_0, t_0 + T_{\mathrm{int}}\right) - \boldsymbol{x}_{\mathrm{ic}} \quad (28)$$

with some initial condition $\boldsymbol{x}_{\mathrm{ic}}$, and we want the vector $\boldsymbol{\varepsilon}$ to be zero. The solution of (28) can be found by means of the Newton-Raphson method in the similar way to (7)

$$\frac{\partial \boldsymbol{\varepsilon}}{\partial \boldsymbol{x}_{\mathrm{ic}}}^{(j)}(\boldsymbol{x}_{\mathrm{ic}}, t_0, t_0 + T_{\mathrm{int}})\Delta \boldsymbol{x}_{\mathrm{ic}}^{(j)} = -\boldsymbol{\varepsilon}^{(j)}(\boldsymbol{x}_{\mathrm{ic}}, t_0, t_0 + T_{\mathrm{int}})$$

where $j = 1, \ldots, j_{\max}$ marks the iteration index of the procedure. Let us note that in substance, the algorithm consists in simultaneous running of the two procedures: integration over the period, and minimization of the artificial equation for the periodic steady state (28).

Computing the Jacobian $\frac{\partial \boldsymbol{\varepsilon}}{\partial \boldsymbol{x}_{\mathrm{ic}}}^{(j)}$ by means of (28), we obtain

$$\left[\frac{\partial \mathcal{I}}{\partial \boldsymbol{x}_{\mathrm{ic}}}^{(j)}(\boldsymbol{x}_{\mathrm{ic}}, t_0, t_0 + T_{\mathrm{int}}) - \mathbf{1}\right]\Delta \boldsymbol{x}_{\mathrm{ic}}^{(j)} = - \boldsymbol{\varepsilon}^{(j)}\left(\boldsymbol{x}_{\mathrm{ic}}, t_0, t_0 + T_{\mathrm{int}}\right).$$

However, the expression $\frac{\partial \mathcal{I}}{\partial \boldsymbol{x}_{\mathrm{ic}}}^{(j)}(\boldsymbol{x}_{\mathrm{ic}}, t_0, t_0 + T_{\mathrm{int}})$ can be efficiently calculated if nonstandard time-domain sensitivities $\boldsymbol{x}'\left(t_0 + T_{\mathrm{int}}, \boldsymbol{x}_{\mathrm{ic}}\right) \equiv \partial \boldsymbol{x}/\partial \boldsymbol{x}_{\mathrm{ic}}\left(t_0 + T_{\mathrm{int}}, \boldsymbol{x}_{\mathrm{ic}}\right)$ are available:

$$\frac{\partial \mathcal{I}}{\partial \boldsymbol{x}_{\mathrm{ic}}}^{(j)}(\boldsymbol{x}_{\mathrm{ic}}, t_0, t_0 + T_{\mathrm{int}}) = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}_{\mathrm{ic}}}^{(j)}(t_0 + T_{\mathrm{int}}, \boldsymbol{x}_{\mathrm{ic}}) \quad (29)$$

where the right side of (29) is an output of a procedure with the built-in (and atypical) time-domain sensitivity analysis. The method is built on the notation used in Sec. 2.1. A system of parametric differential-algebraic equations of a circuit can be symbolically written in the following implicit form

$$\boldsymbol{f}\left[\boldsymbol{x}(t, x_{\mathrm{ic}_i}), \dot{\boldsymbol{x}}(t, x_{\mathrm{ic}_i}), t, x_{\mathrm{ic}_i}\right] = \mathbf{0} \quad (30)$$

where x_{ic_i} is an element of the vector of the initial conditions $\boldsymbol{x}_{\mathrm{ic}}$. By differentiating (30) with respect to x_{ic_i} and using the abbreviations $\boldsymbol{x}'(t, x_{\mathrm{ic}_i}) \equiv \partial \boldsymbol{x}(t, x_{\mathrm{ic}_i})/\partial x_{\mathrm{ic}_i}$ and $\dot{\boldsymbol{x}}'(t, x_{\mathrm{ic}_i}) \equiv \partial \dot{\boldsymbol{x}}(t, x_{\mathrm{ic}_i})/\partial x_{\mathrm{ic}_i}$, we obtain base for deriving:

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{x}'(t, x_{\mathrm{ic}_i}) + \frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{x}}}\dot{\boldsymbol{x}}'(t, x_{\mathrm{ic}_i}) + \frac{\partial \boldsymbol{f}}{\partial x_{\mathrm{ic}_i}} = \mathbf{0}. \quad (31)$$

Utilizing the formulae of Sec. 2.1, we can find a general recurrent formula for the requested time-domain sensitivities:

$$\left[\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)_{n+1} + \gamma_{n+1}\left(\frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{x}}}\right)_{n+1}\right]\boldsymbol{x}'_{n+1} = -\left(\frac{\partial \boldsymbol{f}}{\partial x_{\mathrm{ic}_i}}\right)_{n+1}$$
$$+ \left(\frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{x}}}\right)_{n+1}\sum_{l=1}^{k_{n+1}} \alpha_{n+1}^{(l-1)}\,\delta^{(l-1)}\boldsymbol{x}'_n \sum_{k=l}^{k_{n+1}} \frac{1}{t_{n+1} - t_{n+1-k}}, \quad (32)$$

which has the same Jacobian as that in (7) (for each x_{ic_i}) — therefore, the time-consuming LU factorization of this matrix must be executed *only once* for each t_{n+1}, $n = 0, \ldots$. Note that the proof of (32) is considerably complicated and can be found in a very comprehensive form in [9]. Moreover, let us emphasize that using the novel powerful recurrent formula (32) substantially enhances the efficiency of the algorithm based on the sensitivity analysis.

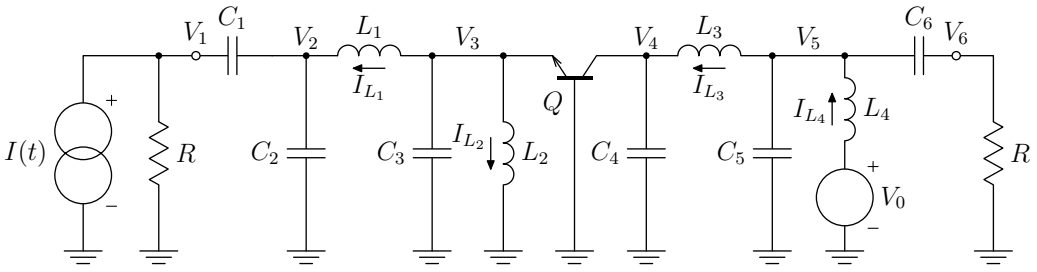

Fig. 1. C-class amplifier used for comparison of two steady-state procedures based on ϵ-algorithm and time-domain sensitivity analysis.

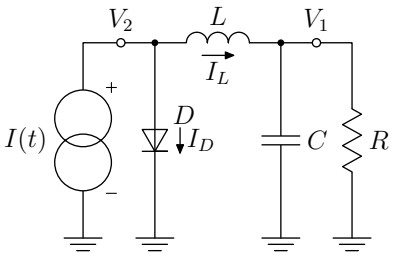

Fig. 2. Rectifier used for comparison of the two steady-state procedures.

Utilized method	The ϵ-algorithm	Sensitivity analysis
Requested precision	0.000001	0.000001
Number of iterations	2	3
Number of periods	8	3

Tab. 1. Numbers of algorithm's iterations and corresponding periods of numerical integration for rectifier.

Utilized method	The ϵ-algorithm		Sensitivity analysis	
Requested precision	0.01	0.0001	0.01	0.0001
Number of iterations	1	2	4	No conv.
Number of periods	10	20	4	No conv.

Tab. 2. Numbers of algorithm's iterations and corresponding periods of numerical integration for C-class amplifier.

3.3 Comparing Efficiencies of the Procedures

We have compared the efficiencies of the two steady-state procedures based on the ϵ-algorithm and time-domain sensitivity analysis by means of two typical examples. The first one is a rectifier that is shown in Fig. 2, and the results of the comparison are presented in Tab. 1. Seemingly, the ϵ-algorithm looks more successful because it needs the lesser number of iterations. However, from the point of view of the number of periods of the numerical integration, the procedure based on the sensitivity analysis is better because it needs the shorter total interval of integration.

Similar properties can be observed by analyzing a C-class amplifier in Fig. 1 with the results that are shown in Tab. 2. For the lower precisions, the algorithm based on the sensitivity analysis is more efficient. Although it needs more iterations again, the total number of the necessary periods is lesser for this method. However, for the more precise analyses, the algorithm based on the extrapolation is better. This is mostly caused by a numerical way that software tools predominantly use for a computation of the sensitivities. (In this example, the numerical method has also been used.) Therefore, the ϵ-algorithm is generally better if a large precision is claimed. However, if an analytical way is used for computing the time-domain sensitivities [9], the algorithm based on sensitivities could be better. Unfortunately, the fixed built-in models of semiconductor devices (that are very complicated) do not contain necessary code for the symbolic evaluation of the derivatives yet. That is why we prefer the ϵ-algorithm.

4. A Series of Tests of the ϵ-Algorithm

4.1 Van der Pol Oscillator

The Van der Pol oscillator represents a classical test for the algorithms. We have analyzed it using a circuit analogy in Fig. 3 with the `quad` function defined by an *input* language

```
function quad(qi, qu)
implicit real*10 (a-z)
quad = (qi ** 2 - 1) * qu
end
```

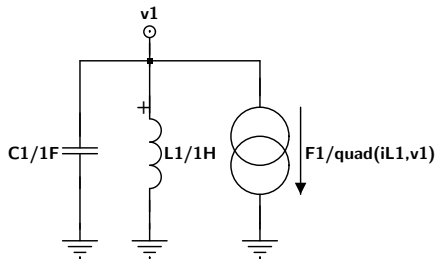

Fig. 3. Circuit analogy of Van der Pol oscillator.

Utilized method	Numbers of integration steps for numerical integration						
	1st	2nd	3rd	4th	5th	6th	Total
Extrapol.	13	7	12	55	115	108	**310**

Tab. 3. Numbers of integration steps necessary for the core of the extrapolation algorithm. Interpolation polynomials of up to 6th order are generally used at integrating.

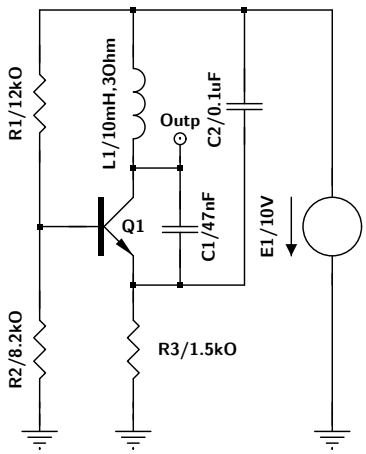

Fig. 4. Colpitts oscillator used as a test of the ϵ-algorithm.

Utilized method	Numbers of integration steps for numerical integration						
	1st	2nd	3rd	4th	5th	6th	Total
1st iter.	22	88	171	182	126	108	697
2nd iter.	19	50	98	91	57	35	350
Extrapol.	41	138	269	273	183	143	**1047**
Classical	32	159	316	339	219	176	**1241**

Tab. 4. Comparison of the numbers of integration steps in the extrapolation algorithm and classical numerical integration (polynomials of up to 6th order were again used by the two methods) for the Colpitts oscillator.

The input language of used our original software tool C.I.A. (Circuit Interactive Analyzer) contains a part fully compatible with a subset of the Fortran 95 programming language.

The level k_{extpol} of the extrapolation algorithm was 2, of course, and only one iteration was necessary for finding the steady state. Therefore, the computational effort of the ϵ-algorithm was the same as that for the classical numerical integration. Moreover, Tab. 3 clearly shows that the 5th and 6th orders were mainly used, which is typical for this circuit.

4.2 Colpitts Oscillator

The second tested circuit was the standard Colpitts oscillator in Fig. 4 with the results of the comparison of the ϵ-algorithm with the classical numerical integration in Tab. 4.

The level k_{extpol} of the extrapolation algorithm was 3, and the fast decaying transients were filtered by $\Delta t_{\mathrm{extpol}} = 0.125$ ms. As shown, 1241 and 1047 integration steps were necessary in the cases of the classical numerical integration and the ϵ-algorithm, respectively, which indicates a moderate improvement with usage of the extrapolation method.

4.3 Fragment of a Large Switched Circuit

The third tested circuit was a small fragment of a large switched circuit in Fig. 5 with the results in Tab. 5. The level k_{extpol} of the extrapolation algorithm was estimated to 3 too.

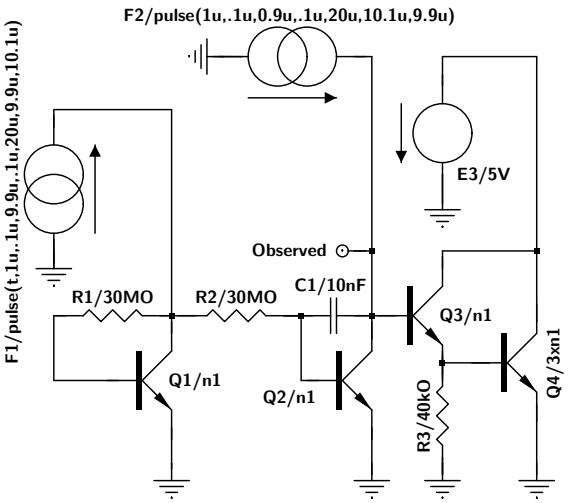

Fig. 5. Fragment of a large switched circuit. (ON Semicond.)

Utilized method	Numbers of integration steps for numerical integration						
	1st	2nd	3rd	4th	5th	6th	Total
1st iter.	169	133	53	4	0	0	359
2nd iter.	175	149	51	35	34	9	453
3rd iter.	70	49	36	25	13	5	198
Extrapol.	414	331	140	64	47	14	**1010**
Classical	1545	1278	414	33	1	0	**3271**

Tab. 5. Comparison of the numbers of integration steps in the extrapolation algorithm and classical numerical integration for the fragment of a large switched circuit.

The two current sources are defined by the Fortran 95 subset:

```
 function pulse(
-               time, delay,
-               rise, width, fall,
-               period, v1, v2)
 implicit real*10 (a-z)
 t = time
 do while (t > period)
    t = t - period
 end do
 t1 =      delay
 t2 = t1 + rise
 t3 = t2 + width
 t4 = t3 + fall
 if      (t <= t1.or.t >= t4) then
    pulse = v1
 else if (t <= t2) then
    pulse = v1 + (t - t1) / (t2 - t1) *
/           (v2 - v1)
 else if (t <= t3) then
    pulse = v2
 else
    pulse = v2 + (t - t3) / (t4 - t3) *
/           (v1 - v2)
 end if
 end
```

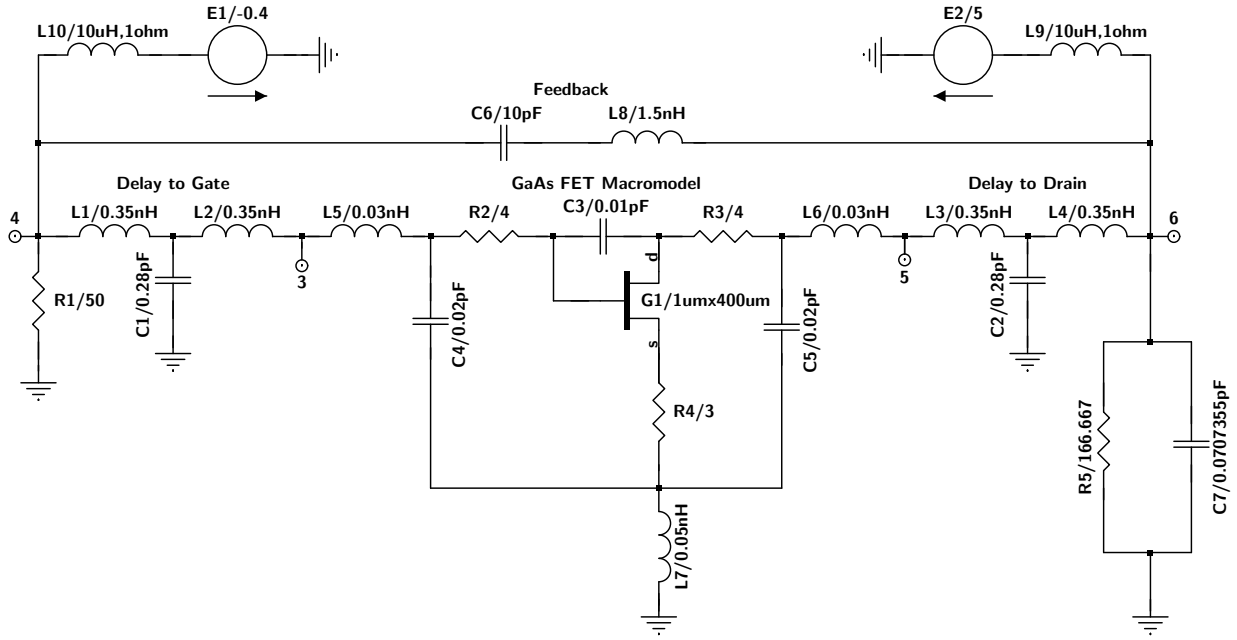

Fig. 6. Feedback microwave oscillator used for comparison of the extrapolation algorithm and classical implicit numerical integration.

Utilized method	Numbers of integration steps for numerical integration						
	1st	2nd	3rd	4th	5th	6th	Total
1st iter.	25	285	877	1023	778	723	3711
2nd iter.	4	84	318	723	1611	3179	5919
Extrapol.	29	369	1195	1746	2389	3902	**9630**
Classical	29	424	1573	3047	7558	16183	**28814**

Tab. 6. Comparison of the numbers of integration steps in the extrapolation algorithm and classical numerical integration for the feedback microwave oscillator.

Let us emphasize that the usage of the subset of Fortran 95 language by the C.I.A. own input compiler enables possible characterization of a practically *arbitrary* dynamic system. The comparison in Tab. 5 also shows clearly that the extrapolation algorithm is approximately three times better than the classical numerical integration. Note that the bipolar junction transistors were modeled according to the standard [14].

4.4 Feedback Microwave Oscillator

The fourth tested circuit was the feedback microwave oscillator in Fig. 6 [15] with the results of the comparison in Tab. 6. The ϵ-algorithm parameters were $k_{\mathrm{extpol}} = 2$ and $\Delta t_{\mathrm{extpol}} = 0.25$ ns. As shown, 28814 and 9630 integration steps were necessary in the cases of the classical numerical integration and the ϵ-algorithm, respectively. (3:1, as well!)

4.5 Distributed Microwave Oscillator

The fifth tested circuit was the tunable distributed microwave oscillator in Fig. 9 [16] with the complicated transient in Fig. 7, and an automatically detected period in Fig. 8.

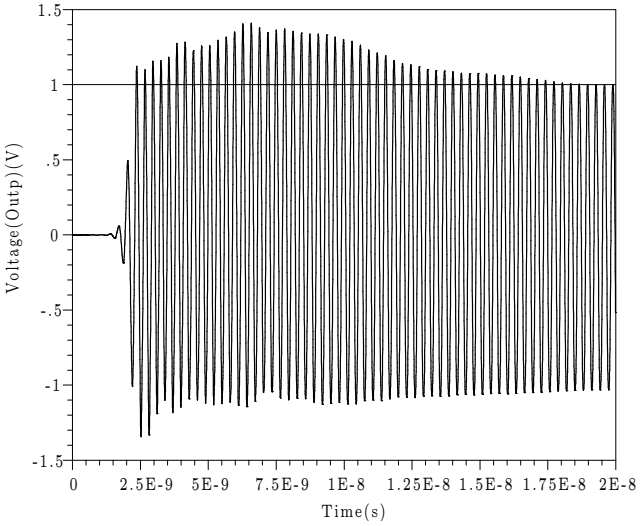

Fig. 7. Long transient of distributed microwave oscillator.

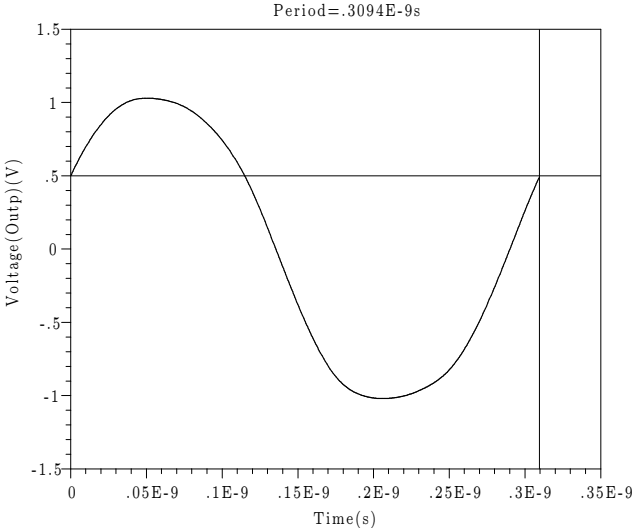

Fig. 8. Detected period of distributed microwave oscillator.

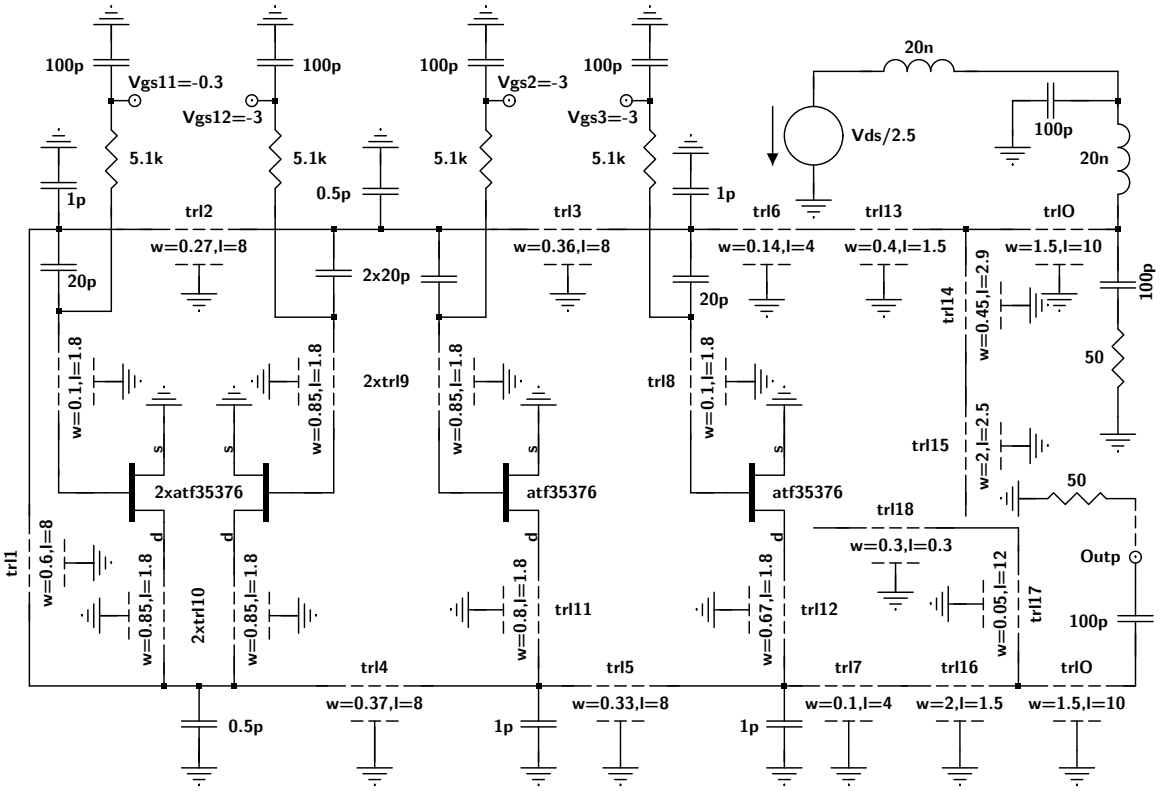

Fig. 9. Tunable distributed microwave oscillator as a sophisticated test of accelerating convergence by the steady-state algorithm. The line dimensions are in millimeters, and the line models are defined by the circuit diagram in Fig. 10 with the parameters in Tab. 9.

Utilized method	Numbers of integration steps for numerical integration						
	1st	2nd	3rd	4th	5th	6th	Total
1st iter.	10	42	166	512	1893	9931	12554
2nd iter.	3	2	24	419	2258	11454	14160
3rd iter.	3	2	21	463	2467	11256	14212
4th iter.	3	2	28	430	2078	8827	11368
Extrapol.	19	48	239	1824	8696	41468	**52294**
Classical	10	38	254	3631	20172	94829	**118934**

Tab. 7. Comparison of the numbers of integration steps in the extrapolation algorithm and classical numerical integration for the tunable microwave oscillator, the case $k_{\mathrm{extpol}} = 15$, $\Delta t_{\mathrm{extpol}} = 5$ ns, and $j_{\max 1} = 100$.

Utilized method	Numbers of integration steps for numerical integration						
	1st	2nd	3rd	4th	5th	6th	Total
1st iter.	8	29	127	534	2066	10678	13442
2nd iter.	3	2	12	333	2286	12566	15202
3rd iter.	3	2	8	405	2384	12488	15290
4th iter.	3	2	7	409	2399	12470	15290
5th iter.	3	2	7	188	1217	6228	7645
Extrapol.	20	37	161	1869	10352	54430	**66869**
Classical	8	29	210	3233	19234	97172	**119886**

Tab. 8. Comparison of the numbers of integration steps in the extrapolation algorithm and classical numerical integration for the tunable microwave oscillator, the case $k_{\mathrm{extpol}} = 20$, $\Delta t_{\mathrm{extpol}} = 3$ ns, and $j_{\max 1} = 500$.

As the transient of this circuit is extraordinarily complicated, two tests have been performed — the first and second with $k_{\mathrm{extpol}} = 15$ and 20, respectively. Furthermore, the delays $\Delta t_{\mathrm{extpol}} = 5$ and 3 ns have been chosen, and the maximum numbers of iterations within a single integration step $j_{\max 1} = 100$ and 500 have been selected, respectively. The results of the two experiments are summarized in Tab. 7 and Tab. 8. As shown, 4 and 5 iterations of the ϵ-algorithm have been needed with necessary total numbers of the integration steps 52294 and 66869, respectively. Therefore, the necessity of total numbers of integration steps is about twice lesser for the ϵ-algorithm than that for the numerical integration.

Note that the transmission lines always enforce large numbers of the integration steps due to their classical LCRG models, which is clearly shown in Tabs. 7 and 8. However, contemporary models of the transmission lines could be even more demanding; approximations based both on semidiscretization of telegrapher's equations [17], [18] and second-order Bessel functions [19] are comparably time-consuming. Hence, the gain given by the steady-state algorithm is useful.

The automatically detected period 0.3094 ns was determined with the amplitude of the output about 1 V – a measured period for similar gate voltages [16] was 0.333 ns with the output level at about 11 dBm, which agrees quite well.

The atf35376 2–18 GHz low noise pseudomorphic HEMT has been characterized by the model [20] – its accuracy comparison with other ones has been performed in [21].

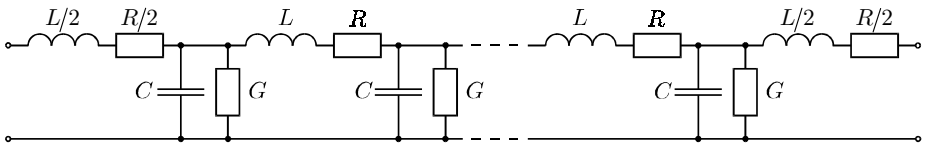

Fig. 10. Standard LCRG model of the lossy transmission lines. The model is not dispersive, i.e., the L, C, R, and G parameters are not a function of frequency, and the TEM propagation mode is assumed.

Trans. line	No. of sections	L (pH)	C (fF)	R (Ω)	$1/G$ (Ω)
trl1	6	509.506	72.037	0.1614	43828
trl2	6	691.5	51.125	0.2232	60608.8
trl3	6	621.07	57.643	0.1992	54094.5
trl4	6	618.677	57.866	0.1984	53886.1
trl5	6	643.059	55.439	0.2067	56127
trl6	3	844.8	40.741	0.2763	75045.7
trl7	3	908.379	37.567	0.2984	81038.9
trl8	1	824.15	34.084	0.3581	67532.7
trl9	1	384.06	80.202	0.1593	30053.2
trl10	2	284.585	59.429	0.0797	60106.3
trl11	2	296.511	57.039	0.083	62625.2
trl12	2	322.095	51.872	0.0907	68444.7
trl13	1	661.5	68.878	0.2257	42560.8
trl14	2	617.385	68.697	0.191	47052.6
trl15	2	239.456	145.988	0.0816	20101.6
trl16	2	143.775	87.654	0.0466	35177.8
trl17	10	933.887	28.76	0.4149	78253
trl18	1	153.63	11.863	0.0655	197690.8
trlO	8	250	100	0.0792	31584.2

Tab. 9. Numbers of sections, inductances, capacitances, resistances, and conductances per unit length of the models of the transmission lines.

The atf35376 model parameters were adjusted using an auxiliary simple extraction process to the values $V_{T0} = -1$ V, $\beta = 0.08$ A V^{-2}, $n_2 = 2$, $\lambda = 0.0015$ V^{-1}, $\alpha = 2$ V^{-1}, $\sigma = 0.002$, $r_D = 0.62\ \Omega$, $r_S = 1.38\ \Omega$, $r_G = 0.98\ \Omega$, $\epsilon W = 5$ fF, $C_{GD0} = 0.05$ pF, $C_{GS0} = 0.3$ pF, $\phi_0 = 1$ V, $C_{DS} = 0.2$ pF, and $m = 0.5$; all were set for the model [20].

Note that used model [20] is noticeably simpler than those with precise higher-order derivatives like [22], which are appropriate for analyzing mixers. However, the comparison [21] confirmed that its precision is fully sufficient, especially for analyzing amplifiers and oscillators.

4.6 Optimized C-Class LDMOS Amplifier

The examples solved above demonstrated that the typical acceleration of the extrapolation algorithm is about two to three times. This is useful for more complicated circuits but maybe not decisive if fast multicore computers are used. However, when finding the steady state is necessary as a part of an optimization process, this acceleration is fundamental. It can be clearly demonstrated by the multiobjective optimization of the last stage of an RF power amplifier in Fig. 11,

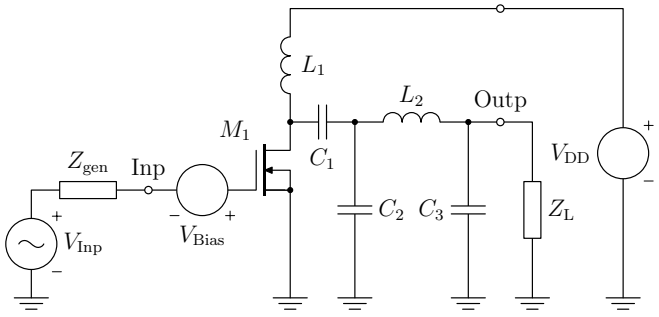

Fig. 11. C-class amplifier with optimized steady-state period.

which serves for a narrow-band signal with an analog modulation at the frequency $f_1 = 300$ MHz. The source and load impedances are both 50 Ω, and the supply voltage V_{DD} is 12 V. The goal is to explore the tradeoffs between achievable output power, power efficiency and total harmonic distortion. We use the lp821 LDMOS transistor as an active component and a topology typical for the C-class mode of operation. The transistor is followed by an LC filter to suppress harmonic distortion and provide good impedance matching. The combination of the elements L_1, C_1, and C_2 can also be seen as a tapped resonant circuit.

A detailed technical definition of the circuit, including a comprehensive RF LDMOS model was described in [23] as well as a steady-state period found for selected solution on the Pareto front. However, the behavior of used ϵ-algorithm operating inside the optimization was not discussed in [23].

The first reason why we mention this circuit is extreme laboriousness, which was necessary for a total of 84 obtained solutions covering the three-dimensional Pareto front [23]. Even on a fast eight-core computer, creating the set of non-inferior solutions on the Pareto front necessitated *hours* of computing. Therefore, the acceleration of finding the steady states inside the optimization loops was fundamental here.

The second reason why this task should be mentioned is a question of the stability of the numerical integration. In fact, there is a hierarchy of three iterative processes here: optimization, steady-state algorithm, and numerical integration that can cause difficultly predictable states inside iterations. As a consequence, the interpolation order determined by the application of (14) had to be limited by a lesser number than six for successful obtaining some points of the Pareto front. In other words, the theoretical limit six derived in [8] *cannot* be kept generally if the state-state algorithm is controlled by an optimization. This is an interesting and important finding.

5. Conclusion

Two possible add-ons have been suggested for the algorithm for the implicit numerical integration, which are able to determine the steady state of both nonautonomous and autonomous circuits. A finally preferred method uses an extrapolation technique based on the scalar ϵ-algorithm. For determining unknown periods of autonomous circuits, the method works as a Newton-Raphson-like one using the ability of the improved numerical integration algorithm to determine the derivatives of a circuit output with respect to time at any point. The procedure has been checked on a large number of nonautonomous and autonomous circuits, and six of them — the Van der Pol and Colpitts oscillators, fragment of a large switched circuit, feedback and distributed microwave oscillators, and optimized C-class RF power amplifier with LDMOS — have been demonstrated in a detailed way. The convergence of the ϵ-algorithm is unproblematic in most cases because the method is relatively insensitive to the control parameters. The accuracy is also not a problem because the algorithm has proven to return to the very similar value after a period-long integration.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under the grant agreement no. 230126. The practical works were performed in laboratories supported by the SIX project; the registration number CZ.1.05/2.1.00/03.0072, the operational program Research and Development for Innovation. The paper has also been supported by the Grant Agency of the Czech Republic, grant no. P102/10/1665, and by the internal grants of the Czech Technical University in Prague SGS 10/286/OHK3/3T/13, SGS 11/160/OHK3/3T/13, and SGS 12/151/OHK3/2T/13.

References

[1] ROBERTS, W. G., SEDRA, A. S. *SPICE.* Second Edition. New-York (USA): Oxford University Press, 1997.

[2] *Affirma RF Simulator User Guide, v. 4.4.6.* Technical Report. San Jose (California): Cadence Design Systems, April 2001.

[3] *Micro-Cap 10 2010 Circuit Simulation Newsletter.* Technical Report. Sunnyvale (CA, USA): Spectrum Software, June 2010.

[4] BRENAN, K. E., CAMPBELL, S. L., PETZOLD, L. R. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* Philadelphia (PA, USA): SIAM, 1996.

[5] SHAMPINE, L. F. *Numerical Solution of Ordinary Differential Equations.* New Edition. New York (USA): Chapman & Hall, 2011.

[6] ZHOU, Y., GAD, E., NAKHLA, M. S., ACHAR, R. Structural characterization and efficient implementation technique for *A*-Stable High-Order Integration Methods. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 2012, vol. 31, no. 1, p. 101–108.

[7] DOBEŠ, J., YADAV, A., ČERNÝ, D. Efficient algorithm for solving systems of circuit differential-algebraic equations with reliable divergence suppression in DC and time domains. In *Proceedings of the 54th Midwest Symposium on Circuits and Systems.* Seoul (Korea), 2011, 4 p., wos:000296057200325.

[8] PETRENKO, A. I., VLASOV, A. I., TIMTSCHENKO, A. P. *Tabular Methods of Computer-Aided Modeling.* (In Russian.) Kiyv (Ukraine): Higher School, 1977.

[9] DOBEŠ, J. Advanced types of the sensitivity analysis in frequency and time domains. *AEÜ—International Journal of Electronics and Communications,* 2009, vol. 63, no. 1, p. 52–64, wos: 000263477800008.

[10] ENGELN-MÜLLGES, G., UHLIG, F. *Numerical Algorithms with C.* Berlin (Germany): Springer-Verlag, 1996.

[11] VLACH, J., SINGHAL, K. *Computer Methods for Circuit Analysis and Design.* New York (USA): Van Nostrand Reinhold Company, 1982.

[12] SKELBOE, S. Computation of the periodic steady-state response of non-linear networks by extrapolation methods. *IEEE Transactions on Circuits and Systems,* 1980, vol. 27, no. 3, p. 161–175, wos: a1980jn39700001.

[13] SKELBOE, S. Time-domain steady-state analysis of non-linear electrical systems. *Proceedings of the IEEE,* 1982, vol. 70, no. 10, p. 1210–1228, wos:a1982pq55400004.

[14] MASSOBRIO, G., ANTOGNETTI, P. *Semiconductor Device Modeling With SPICE.* 2nd ed. New York (USA): McGraw-Hill, 1993.

[15] TAJIMA, Y., WRONA, B., MISHIMA, K. GaAs FET large-signal model and its application to circuit designs. *IEEE Transactions on Electron Devices,* 1981, vol. ED-28, no. 2, p. 171–175, wos: a1981kz60700008.

[16] DIVINA, L., ŠKVOR, Z. The distributed oscillator at 4 GHz, *IEEE Transactions on Microwave Theory and Techniques,* 1998, vol. 46, no. 12, p. 2240–2243, wos:000078099000009.

[17] GUO, Y.-S. Transient analysis of transmission line circuits based on the semidiscretization of telegraph equations. *Journal of Electronics,* 2001, vol. 18, no. 1, p. 46–55.

[18] GUO, Y.-S. Transient simulation of high-speed interconnects based on the semidiscretization of telegrapher's equations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 2002, vol. 21, no. 7, p. 799–809, wos:000176508500005.

[19] DOBEŠ, J., SLÁMA, L. A modified Branin model of lossless transmission lines. In *Proceedings of the 52nd Midwest Symposium on Circuits and Systems.* Cancun (Mexico), 2009, p. 236–239, wos: 000277574000059.

[20] DOBEŠ, J., POSPÍŠIL, L. Enhancing the accuracy of microwave element models by artificial neural networks, *Radioengineering,* 2004, vol. 13, no. 3, p. 7–12, wos:000208050000002.

[21] MEMON, N. M., AHMED, M. M., REHMAN, F. A comprehensive four parameters I–V model for GaAs MESFET output characteristics. *Solid-State Electronics,* 2007, vol. 51, no. 3, p. 511–516, wos: 000246313400027.

[22] PARKER, A. E., SKELLERN, D. J. A realistic large-signal MESFET model for SPICE. *IEEE Transactions on Microwave Theory and Techniques,* 1997, vol. 45, no. 9, p. 1563–1571, wos: a1997xu75500004.

[23] DOBEŠ, J., MÍCHAL, J., PAŇKO, V., POSPÍŠIL, L. Reliable procedure for electrical characterization of MOS-based devices, *Solid-State Electronics,* 2010, vol. 54, no. 10, p. 1173–1184, wos: 000281019100022.

About Authors...

Josef DOBEŠ received the Ph.D. degree in microelectronics at the Czech Technical University in Prague in 1986. From 1986 to 1992, he was a researcher of the TESLA Research Institute, where he performed analyses on algorithms for CMOS Technology Simulators. Currently, he works at the Department of Radio Electronics of the Czech Technical University in Prague. His research interests include the physical modeling of radio electronic circuit elements, especially RF and microwave transistors and transmission lines, creating or improving special algorithms for the circuit analysis and optimization, such as time- and frequency-domain sensitivity, poles-zeros or steady-state analyses, and creating a comprehensive CAD tool for the analysis and optimization of RF and microwave circuits.

Viera BIOLKOVÁ received her MSc degree in electrical engineering from the Brno University of Technology, Czech Republic, in 1983. She joined the Department of Radio Electronics in 1985 and is currently working as a research assistant at the Department of Radio Electronics, Brno University of Technology (BUT), Czech Republic. Her research and educational interests include signal theory, analogue signal processing and digital electronics.