An Efficient Procedure for the Time-Domain Sensitivity Analysis

Josef DOBEŠ

Department of Radio Engineering, Czech Technical University in Prague, Technická 2, 166 27 Praha 6, Czech Republic

dobes@feld.cvut.cz

Abstract. *Standard tools for CAD have limited modes of the sensitivity analysis: PSPICE only contains a static mode and SPECTRE includes frequency-domain and static modes. However, many RF systems use symmetrical structures for enhancing the circuit properties. For such systems, the static sensitivities are zero on principle and hence the time-domain sensitivity analysis should be used. In the paper, a novel recurrent formula for the time-domain sensitivity analysis is derived which uses by-products of an efficient implicit integration algorithm. As the selected integration algorithm is more flexible than the Gear's one that is ordinarily used, the sensitivity analysis is more efficient in comparison with the standard CAD tools. An implementation of the method is demonstrated using the analysis of a low-voltage four-quadrant RF multiplier. Nonstandard temperature sensitivity analyses are also tested in the static and dynamic modes.*

Keywords

Implicit integration algorithm, sensitivity analysis, RF CMOS circuits, analog multiplier, four-quadrant mixer.

1. Introduction

The circuit equations can only be defined in an implicit form in most cases. Therefore, a robust and effective algorithm for the implicit integration of a system of algebraic-differential equations must be used. Hence, main features of such algorithm are described at first. Thereafter, a novel formula is derived for the time-domain sensitivity analysis that cooperates with this integration algorithm in a natural way.

2. Definition of the Algorithms

The system of nonlinear algebraic-differential equations of a circuit is generally defined in the implicit form

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

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

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

(a concept of the backward scaled differences with a detailed stability investigation can be found in [1], e.g.), where k_n is the order of the polynomial interpolation used in the last integration step, and the $\alpha_n^{(\ldots)}$ multipliers are also determined using the recurrent scheme:

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

The *predictor* of the variables 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):

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

((4) is a more sophisticated form of the Newton interpolation polynomial—see the proof of Theorem 1 in the Appendix).

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

$$\begin{aligned}&\left[\left(\frac{\partial\boldsymbol{f}}{\partial\boldsymbol{x}}\right)_{n+1}^{(j)}+\gamma_{n+1}\left(\frac{\partial\boldsymbol{f}}{\partial\dot{\boldsymbol{x}}}\right)_{n+1}^{(j)}\right]\Delta\boldsymbol{x}_{n+1}^{(j)}=-\boldsymbol{f}_{n+1}^{(j)},\\&\qquad\qquad j=0,\ldots,j_{\max n},\end{aligned} \tag{5}$$

i.e., by repeated solving the linear system (5) with applying the γ_{n+1} factor approximated by (see Theorem 2 in the Appendix)

$$\gamma_{n+1}=\sum_{k=1}^{k_{n+1}}\frac{1}{t_{n+1}-t_{n+1-k}}, \tag{6}$$

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

After resolving the linear system (5), the vectors $\boldsymbol{x}_{n+1}^{(\ldots)}$ and $\dot{\boldsymbol{x}}_{n+1}^{(\ldots)}$ are updated in the standard way:

$$\begin{aligned}\boldsymbol{x}_{n+1}^{(j+1)} &= \boldsymbol{x}_{n+1}^{(j)} + \Delta\boldsymbol{x}_{n+1}^{(j)},\\ \dot{\boldsymbol{x}}_{n+1}^{(j+1)} &= \dot{\boldsymbol{x}}_{n+1}^{(j)} + \gamma_{n+1}\Delta\boldsymbol{x}_{n+1}^{(j)},\end{aligned} \tag{7}$$

which completes the $j+1$ iteration of the $n+1$ time step. However, if an indication of divergence is detected during the iterations, then the logarithmic damping [2] is applied to each component ${}^i\Delta x_{n+1}^{(j)}$ of the vector $\Delta\boldsymbol{x}_{n+1}^{(j)}$

$${}^i\Delta x_{n+1}^{(j)} := \operatorname{sgn}\left({}^i\Delta x_{n+1}^{(j)}\right)\left|{}^i x_{n+1}^{(j)}\right| \ln\left(1+\frac{\left|{}^i\Delta x_{n+1}^{(j)}\right|}{\left|{}^i x_{n+1}^{(j)}\right|}\right), \quad i=1,\ldots,m \tag{8}$$

before the execution of (7) (m is the dimension of $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$).

Operating-point analysis is performed using the static variant of (1)

$$\boldsymbol{f}\left(\boldsymbol{x}_0, \boldsymbol{0}, t_0\right) = \boldsymbol{f}_0\left(\boldsymbol{x}_0\right) = \boldsymbol{0}, \tag{9}$$

which is solved by the static variants of (5) and (7):

$$\left(\frac{\partial \boldsymbol{f}_0}{\partial \boldsymbol{x}_0}\right)^{(j)} \Delta\boldsymbol{x}_0^{(j)} = -\boldsymbol{f}_0^{(j)},\ \boldsymbol{x}_0^{(j+1)} = \boldsymbol{x}_0^{(j)} + \Delta\boldsymbol{x}_0^{(j)}, \quad j=0,\ldots,j_{\max_0}. \tag{10}$$

However, the convergence in the operating-point analysis is often more problematic than that in the transient analysis. (In the transient analysis, the results of the previous step serve as a good estimation for the following step.) To avoid possible divergence, a novel control mechanism has been developed [3] for handling the differences $\Delta\boldsymbol{x}_0^{(j)}$ during each iteration:

```
if j = 0 then
    x̃ := x_0^(0),
    Δx̃ := Δx_0^(0),
    f̃ := f_0^(0),
    iteration is accepted,
else
    if (1/m) Σ_{i=1}^{m} |^i f_0^(j)| / (|^i f̃| + ^i f_null) < 1 then
        x̃ := x_0^(j),
        Δx̃ := Δx_0^(j),
        f̃ := f_0^(j),
        iteration is accepted,
    else
        Δx̃ := Δx̃ / 2,
        x_0^(j) := x̃,
        Δx_0^(j) := Δx̃,
        iteration is rejected,
```

(11)

which checks the residual value of $\boldsymbol{f}_0^{(j)}$ after each iteration. The basic idea of handling the differences $\Delta\boldsymbol{x}_0^{(j)}$ in accord with (11) relates to the fundamental property of the Newton-Raphson method. If the average value of the residues does not decrease then the difference is halved and the iteration is repeated. The halving continues until the average residual value has decreased. It is sure that the occurrence of the decreased average residue will be found and therefore the algorithm does not even contain a check for a possible infinite loop (!). As a result, only such $\Delta\boldsymbol{x}_0^{(j)}$ is used for updating the vector $\boldsymbol{x}_0^{(j)}$ that ensures the decrease of the average value of the residues.

Note that the parameters ${}^i f_{\mathrm{null}}$ prevent possible division by zero and $\tilde{\boldsymbol{x}}$, $\Delta\tilde{\boldsymbol{x}}$, and $\tilde{\boldsymbol{f}}$ are auxiliary vectors.

Using the Newton-Raphson method (10) with the controlling procedure (11) leads to a very reliable convergence. However, the number of iterations could be very large in that case and therefore the logarithmic damping (8) should not be applied if its use is not necessary for another reason.

2.1 A Novel Recurrent Formula for the Time-Domain Sensitivity Analysis

With respect to the notation used above, a system of the parametric algebraic-differential equations of a circuit can be symbolically written in the form

$$\boldsymbol{f}\left[\,\boldsymbol{x}(t,p), \dot{\boldsymbol{x}}(t,p), t, p\,\right] = \boldsymbol{0}, \tag{12}$$

where p is one of the circuit parameters on which the sensitivities are requested. Differentiating (12) with respect to p and using the abbreviations $\boldsymbol{x'}(t,p) \equiv \partial\boldsymbol{x}(t,p)/\partial p$ and $\dot{\boldsymbol{x}}'(t,p) \equiv \partial\dot{\boldsymbol{x}}(t,p)/\partial p$, we obtain

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\,\boldsymbol{x'}(t,p) + \frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{x}}}\,\dot{\boldsymbol{x}}'(t,p) + \frac{\partial \boldsymbol{f}}{\partial p} = \boldsymbol{0}. \tag{13}$$

After a derivation with using the backward scaled differences (2, 3) and corrector (5, 6) (see the proof of Theorem 3 in the Appendix), we obtain the novel recurrent formula

$$\left[\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)_{n+1} + \gamma_{n+1}\left(\frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{x}}}\right)_{n+1}\right]\boldsymbol{x'}_{n+1} = -\left(\frac{\partial \boldsymbol{f}}{\partial p}\right)_{n+1} + \left(\frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{x}}}\right)_{n+1} \times \boxed{\sum_{l=1}^{k_{n+1}} \alpha_{n+1}^{(l-1)}\,\delta^{(l-1)}\boldsymbol{x'}_n \sum_{k=l}^{k_{n+1}} \frac{1}{t_{n+1}-t_{n+1-k}}}, \tag{14}$$

which has the same Jacobian as that in (5) (for each p)—therefore, the laborious LU factorization of that matrix must be executed only *once* (of course) for each t_{n+1}, $n=0,\ldots$.

The part of (14) which is enclosed by the box enables more accurate and efficient computing the sensitivities in comparison with other interpolation formulae defined in [4].

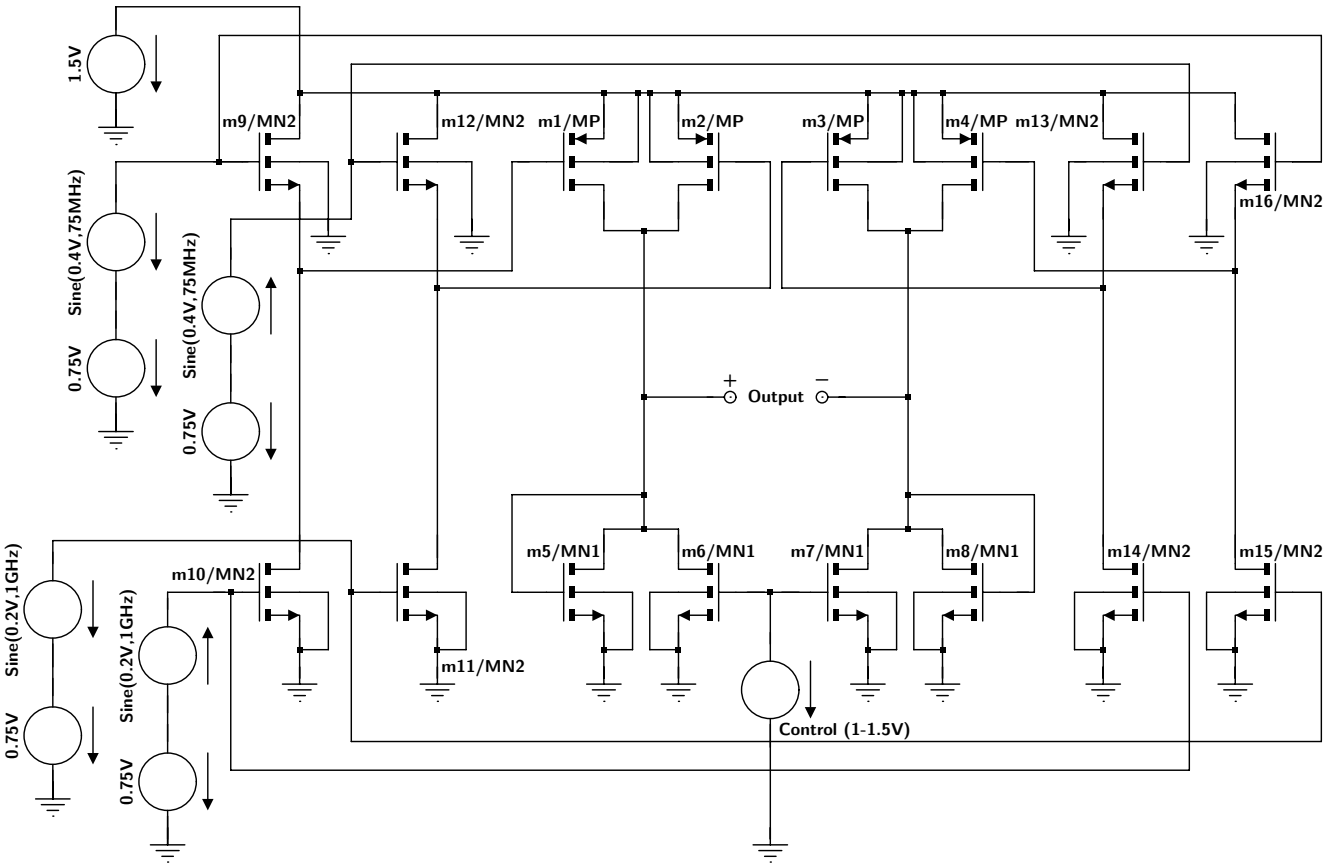

Fig. 1. Low-voltage low-power CMOS RF four-quadrant multiplier with symmetrical low-frequency (input signal) and high-frequency (local oscillator) sources.

2.2 Operating-Point Sensitivity Analysis

For determining initial parametric derivative $\boldsymbol{x}'(\Delta t, p)$ by the recurrent formula (14) (note that the implicit integration algorithm must always start with the first-order method), the static parametric derivative $\boldsymbol{x}'(0,p) = \boldsymbol{x}'_0(p)$ *must* be computed in advance. This vector can be obtained by differentiating a parametric version of (9)

$$\boldsymbol{f}_0\big[\,\boldsymbol{x}_0(p), p\,\big] = \boldsymbol{0}, \tag{15}$$

which gives the simpler system of linear equations than (14)

$$\frac{\partial \boldsymbol{f}_0}{\partial \boldsymbol{x}_0}\big[\,\boldsymbol{x}_0(p), p\,\big]\ \boldsymbol{x}'_0(p) = -\frac{\partial \boldsymbol{f}_0}{\partial p}\big[\,\boldsymbol{x}_0(p), p\,\big]. \tag{16}$$

3. Checking the Implementation

3.1 Four-Quadrant CMOS RF Multiplier

Let us consider a four-quadrant CMOS RF multiplier in Fig. 1 [5], which has been checked using the sensitivity analysis described in Section 2. Note that a medium-wave version of the circuit has been realized at our department, and we plan to perform the IM3 measurement. The output voltage of the multiplier is highly dependent on the controlling one applied to the gates of $\mathrm{m6}$ and $\mathrm{m7}$ transistors.

A comparison of the output signals corresponding to the two controlling voltages is shown in Fig. 2. For the controlling voltages 1 and 1.5 V, the magnitudes of the output signal are about 20 and 50 mV, respectively.

The output voltage of the multiplier is also very dependent on the zero-bias threshold voltages V_{T0} ([6], [7]) of the transistors. The sensitivities of the output voltage on these important model parameters, i.e., the functions

$$\frac{\partial V_{\mathrm{Output}}}{\partial V_{T0,\mathrm{MN1}}}(t),\ \frac{\partial V_{\mathrm{Output}}}{\partial V_{T0,\mathrm{MN2}}}(t),\ \text{and}\ \frac{\partial V_{\mathrm{Output}}}{\partial V_{T0,\mathrm{MP}}}(t)$$

are shown in Fig. 3. As we can observe, the sensitivity on the threshold voltage $V_{T0,\mathrm{MP}}$ is the most significant. Let us emphasize that the utilized software tool [2] uses another definition for the zero-bias threshold voltage than PSPICE—the original values for the standard PSPICE type model were 0.62 and -0.58 V [5] (in the modified model [2], the threshold voltage is positive for both N and P channel enhancement mode transistors—this is the same convention as that in the PSPICE JFET modeling [8]).

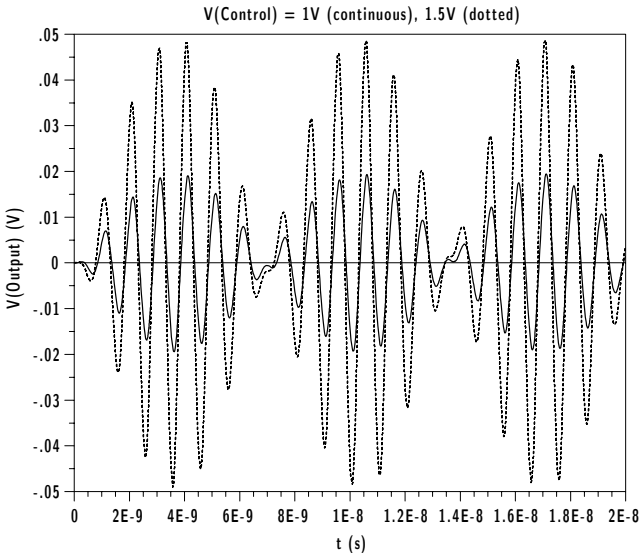

Fig. 2. Dependence of the output voltage of the multiplier on the controlling voltage.

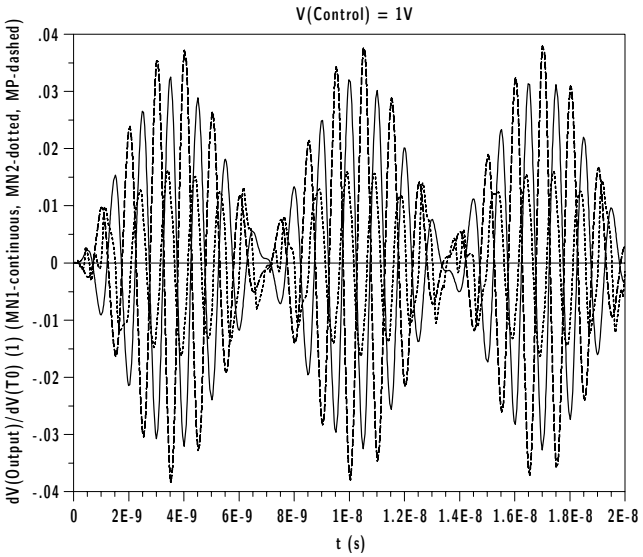

Fig. 3. Sensitivities of the output voltage of the multiplier on the zero-bias threshold voltages of the transistors.

The results in Fig. 3 can simply be checked. The zero-bias threshold voltages of the MNx and MP transistors were 0.47 and 0.44 V, respectively. We can execute another analysis with somewhat changed $V_{T0,\mathrm{MP}}$ parameter and estimate the sensitivity numerically. For example, let us use a modified value $V'_{T0,\mathrm{MP}} = 0.4$ V. At 17 ns (where the sensitivity of V_{Output} on $V_{T0,\mathrm{MP}}$ has the maximum 0.0379411 as seen in Fig. 3), the values of the output voltage V_{Output} and V'_{Output} (computed using the values $V_{T0,\mathrm{MP}} = 0.44$ V and $V'_{T0,\mathrm{MP}} = 0.4$ V) were 0.0139211 V and 0.0118593 V, respectively. Now let us compare these results: the predicted value obtained using the sensitivity is $V''_{\mathrm{Output}} = 0.0139211 - 0.04 \times 0.0379411 = 0.0124035$ V, the actual output obtained using the value $V'_{T0,\mathrm{MP}}$ is $V'_{\mathrm{Output}} = 0.0118593$ V, so the error of the prediction is about 4.59 %. At 20 ns, the values of V_{Output}, $\partial V_{\mathrm{Output}}/\partial V_{T0,\mathrm{MP}}$, and V'_{Output} were 0.00152467 V, 0.00303789, and 0.00138769 V, respectively. Comparing again, the predicted value obtained using the sensitivity is $V''_{\mathrm{Output}} = 0.00152467 - 0.04 \times 0.00303789 = 0.00140315$ V, the actual voltage obtained using the modified value $V'_{T0,\mathrm{MP}}$ is $V'_{\mathrm{Output}} = 0.00138769$ V, so the error of the prediction is about 1.11 % now—of course, the error is lesser for the smaller magnitude of the output signal.

If we only want to compute the sensitivities on the zero-bias threshold voltages, the above experiments using the modified values V'_{T0} are possible. However, the semiempirical, BSIM [9], and EKV MOSFET models have many parameters and therefore it is impossible to check the circuit using small differences of the model parameters if we want to check *all* the sensitivities. For complex testing a circuit with respect to all its parameters, the sensitivity analysis seems to be the only effective way. Emphasize that the symmetrical circuit in Fig. 1 has *zero* static sensitivities (on principle). Hence, the PSPICE sensitivity analysis does not offer any usable results here.

3.2 Operating-Point and Time-Domain Temperature Sensitivity Analyses

Consider a power operational amplifier in Fig. 4 for which both operating-point and time-domain temperature sensitivity analyses will be tested. The amplifier has input transistors symmetrically connected as the standard differential pair. Therefore, a sensitivity of the output voltage on local warming up ΔT_1 of Q1 is to be complementary to a sensitivity of the output voltage on local warming up ΔT_2 of Q2. The values of these nonstandard sensitivities will be monitored as an appropriate and unconventional test of the algorithm.

3.2.1 Operating-Point Sensitivity Analysis

The operating-point sensitivities (i.e., the initial static ones) obtained by solving the static modification of (14) (i.e., by (16)) are precisely zero-symmetrical as expected

$$\begin{aligned} \frac{\partial V_{\mathrm{Output}}}{\partial(\Delta T_1)} &= \boxed{+0.00192}51 \text{ V/K}, \\ \frac{\partial V_{\mathrm{Output}}}{\partial(\Delta T_2)} &= \boxed{-0.00192}97 \text{ V/K}. \end{aligned} \tag{17}$$

In other words, if the two transistors have the same warming up (which is natural) by 10 K, e.g., the output voltage changes approximately by -46 μV—the precise symmetry is very important because the output voltage is only about 1.3 mV when the amplifier works at its operating point.

3.2.2 Time-Domain Sensitivity Analysis

The results of solving the recurrent system (14) for the first period of transient with the starting vector (17) are shown in Fig. 5. The best symmetry occurred (as also expected) when the input and output voltages are close to zero values. For other voltages, however, the symmetry is not accurate but sufficient which the analysis clearly validates.

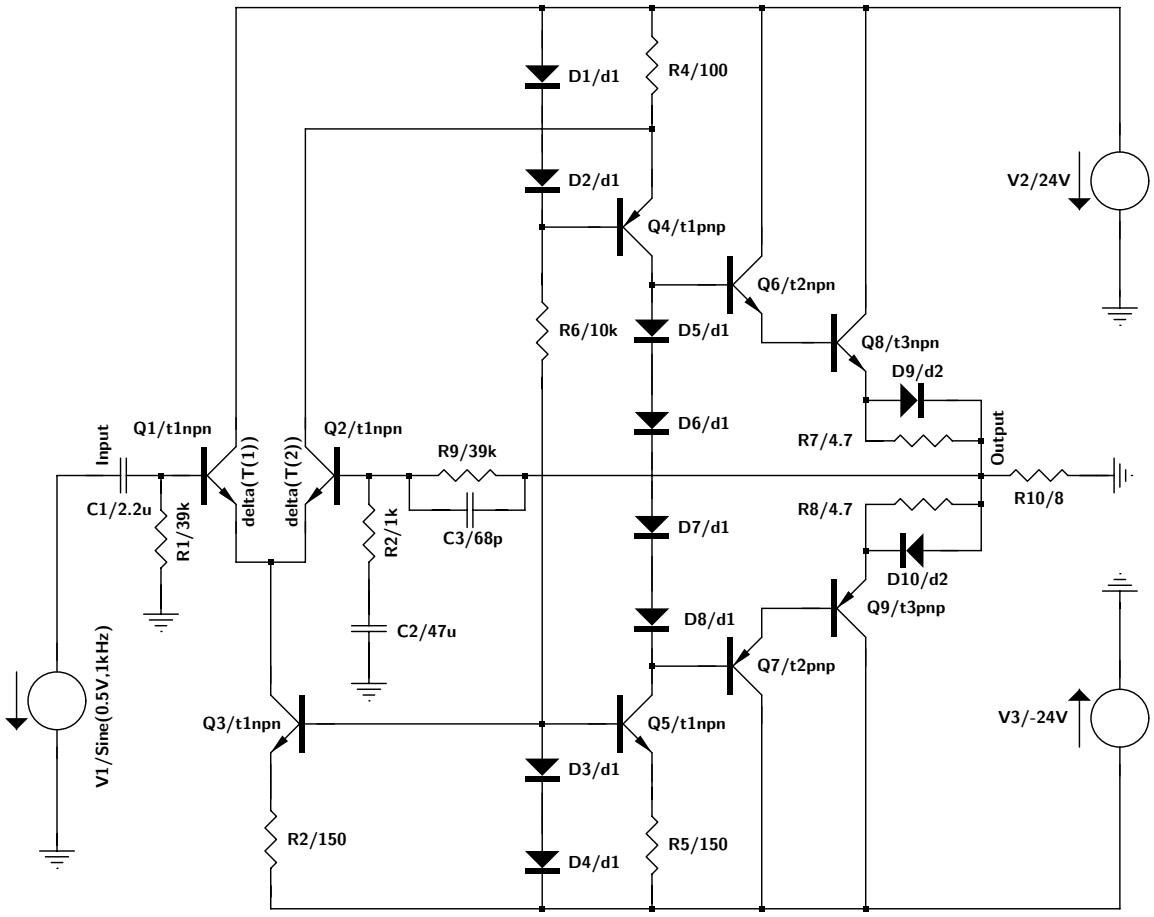

Fig. 4. Power operational amplifier used for testing the operating-point and time-domain temperature sensitivity analyses, respectively.

4. Conclusion

A novel recurrent procedure for the time-domain sensitivity analysis has been proposed that efficiently uses the computational by-products of implicit integration algorithm. As the selected integration algorithm uses the backward scaled differences based on flexible Newton interpolation polynomial, the sensitivity analysis is more efficient than similar ones based on other interpolation schemes. The implementation has successfully been checked analyzing the sophisticated low-voltage low-power CMOS RF multiplier. The correctness of programming the formulae has been checked by means of the classical finite difference analysis assuming that the novel way gives more accurate results on principle. The implementation has also been checked from the physical point of view using the nonstandard temperature sensitivity analysis in both static and dynamic domains.

Acknowledgements

This paper has been supported by the Grant of the European Commission TARGET (FET modeling); and also by the Grant Agency of the Czech Republic, grant No. 102/05/0277, and by the Czech Technical University Research Project MSM 6840770014 (the sensitivity analysis).

Many thanks to Prof. Salama for sending all the MOSFET model parameters.

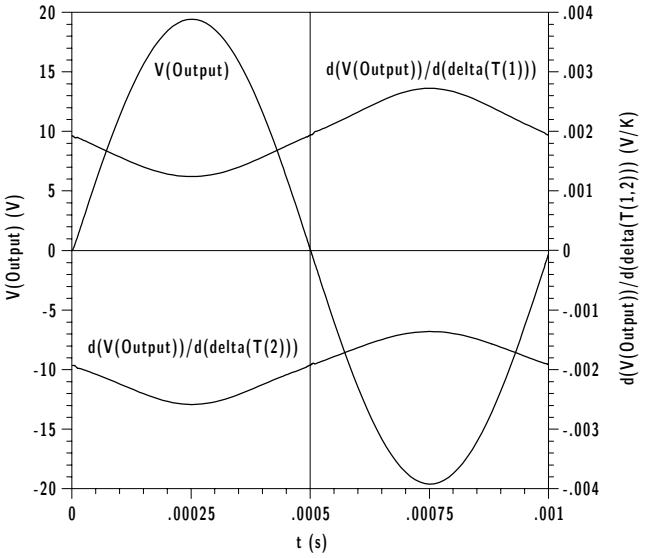

Fig. 5. Symmetrical temperature time-domain sensitivities of the output voltage of the amplifier on ΔT_1 and ΔT_2.

Appendix

Theorem 1 *The formula* (4) *can be regarded as a transformed Newton interpolation polynomial.*

Proof: A member of the interpolation polynomial

$$\begin{aligned}\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + (t_{n+1}-t_n)\,\boldsymbol{x}_{n,n-1} + \\ (t_{n+1}-t_n)\,(t_{n+1}-t_{n-1})\,\boldsymbol{x}_{n,n-1,n-2} + \cdots + \\ (t_{n+1}-t_n)\cdots\left(t_{n+1}-t_{n+1-k_{n+1}}\right)\boldsymbol{x}_{n,\ldots,n-k_{n+1}}\end{aligned} \tag{18a}$$

with the backward scaled differences defined as

$$\begin{aligned}\boldsymbol{x}_{n,n-1} &= \frac{\boldsymbol{x}_n - \boldsymbol{x}_{n-1}}{t_n - t_{n-1}}, \\ &\cdots \\ \boldsymbol{x}_{n,\ldots,n-k_{n+1}} &= \frac{\boldsymbol{x}_{n,\ldots,n+1-k_{n+1}} - \boldsymbol{x}_{n-1,\ldots,n-k_{n+1}}}{t_n - t_{n-k_{n+1}}}\end{aligned} \tag{18b}$$

can sequentially be transformed in the following way:

$$\begin{aligned}&(t_{n+1}-t_n)\cdots(t_{n+1}-t_{n+1-k})\,\boldsymbol{x}_{n,\ldots,n-k} = \\ &\frac{(t_{n+1}-t_n)\cdots(t_{n+1}-t_{n+1-k})}{t_n - t_{n-k}}\,(\boldsymbol{x}_{n,\ldots,n+1-k} - \\ &\boldsymbol{x}_{n-1,\ldots,n-k}) = \frac{(t_{n+1}-t_n)\cdots(t_{n+1}-t_{n+1-k})}{(t_n-t_{n-k})\,(t_n-t_{n+1-k})}\times \\ &\big[\,\boldsymbol{x}_{n,\ldots,n+2-k} - \boldsymbol{x}_{n-1,\ldots,n+1-k} - (t_n - t_{n+1-k})\times \\ &\boldsymbol{x}_{n-1,\ldots,n-k}\,\big] = \cdots = \frac{(t_{n+1}-t_n)\cdots(t_{n+1}-t_{n+1-k})}{(t_n-t_{n-k})\cdots(t_n-t_{n-1})} \\ &\times\big[\,\boldsymbol{x}_n - \boldsymbol{x}_{n-1} - (t_n-t_{n-1})\,\boldsymbol{x}_{n-1,n-2} - \\ &(t_n-t_{n-1})\,(t_n-t_{n-2})\,\boldsymbol{x}_{n-1,n-2,n-3} - \cdots - \\ &(t_n-t_{n-1})\cdots(t_n-t_{n+1-k})\,\boldsymbol{x}_{n-1,\ldots,n-k}\,\big] = \\ &\alpha_{n+1}^{(k)}\Bigg[\boldsymbol{x}_n - \boldsymbol{x}_{n-1} - \frac{t_n-t_{n-1}}{t_{n-1}-t_{n-2}}\,(\boldsymbol{x}_{n-1}-\boldsymbol{x}_{n-2}) - \\ &\overbrace{\frac{(t_n-t_{n-1})\,(t_n-t_{n-2})}{(t_{n-1}-t_{n-3})\,(t_{n-1}-t_{n-2})}}^{\alpha_n^{(2)}}\times \\ &\Bigg(\underbrace{\overbrace{\boldsymbol{x}_{n-1}-\boldsymbol{x}_{n-2}}^{\delta^{(1)}\boldsymbol{x}_{n-1}} - \overbrace{\frac{(t_{n-1}-t_{n-2})}{(t_{n-2}-t_{n-3})}}^{\alpha_{n-1}^{(1)}}\overbrace{(\boldsymbol{x}_{n-2}-\boldsymbol{x}_{n-3})}^{\delta^{(1)}\boldsymbol{x}_{n-2}}}_{\delta^{(2)}\boldsymbol{x}_{n-1}}\Bigg) - \\ &\cdots\Bigg] = \alpha_{n+1}^{(k)}\Big(\boldsymbol{x}_n - \boldsymbol{x}_{n-1} - \alpha_n^{(1)}\delta^{(1)}\boldsymbol{x}_{n-1} - \cdots - \\ &\qquad\alpha_n^{(k-1)}\delta^{(k-1)}\boldsymbol{x}_{n-1}\Big) = \alpha_{n+1}^{(k)}\delta^{(k)}\boldsymbol{x}_n.\end{aligned} \tag{19}$$

Therefore, the Newton interpolation polynomial (18) can be reordered to a more convenient form (4). ■

Theorem 2 *The formula* (6) *can be regarded as a consequence of "borderline" using the formula* (4)*.*

Proof: The vector $\boldsymbol{x}_{n+2}^{(j)}$ is to be expressed as a polynomial created by (4) and (3):

$$\begin{aligned}&\dot{\boldsymbol{x}}_{n+1}^{(j)} = \\ &\lim_{t_{n+2}\to t_{n+1}}\frac{\boldsymbol{x}_{n+2}^{(j)} - \boldsymbol{x}_{n+1}}{t_{n+2}-t_{n+1}} = \\ &\lim_{t_{n+2}\to t_{n+1}}\Bigg(\frac{1}{t_{n+1}-t_n}\,\delta^{(1)}\boldsymbol{x}_{n+1}^{(j)} + \\ &\frac{1}{t_{n+1}-t_n}\,\frac{t_{n+2}-t_n}{t_{n+1}-t_{n-1}}\,\delta^{(2)}\boldsymbol{x}_{n+1}^{(j)} + \cdots + \\ &\frac{1}{t_{n+1}-t_n}\cdots\frac{t_{n+2}-t_{n+2-k_{n+1}}}{t_{n+1}-t_{n+1-k_{n+1}}}\,\delta^{(k_{n+1})}\boldsymbol{x}_{n+1}^{(j)}\Bigg) = \\ &\qquad\sum_{k=1}^{k_{n+1}}\frac{1}{t_{n+1}-t_{n+1-k}}\,\delta^{(k)}\boldsymbol{x}_{n+1}^{(j)},\end{aligned} \tag{20}$$

and the last formula in (20) gives the γ_{n+1} factor in (5) directly—this process is often called "algebraization". ■

Theorem 3 *The derivatives of the sensitivities $\boldsymbol{x}'_{n+1}$ with respect to t_{n+1} can be expressed by the formula*

$$\begin{aligned}\dot{\boldsymbol{x}}'_{n+1} = \gamma_{n+1}\boldsymbol{x}'_{n+1} - \\ \sum_{l=1}^{k_{n+1}}\alpha_{n+1}^{(l-1)}\,\delta^{(l-1)}\boldsymbol{x}'_n\sum_{k=l}^{k_{n+1}}\frac{1}{t_{n+1}-t_{n+1-k}}.\end{aligned} \tag{21}$$

Proof: For the substitution of $\dot{\boldsymbol{x}}'_{n+1}$, we can use the main interpolation system (see the last formula in (20) again)

$$\dot{\boldsymbol{x}}'_{n+1} = \sum_{k=1}^{k_{n+1}}\frac{1}{t_{n+1}-t_{n+1-k}}\,\delta^{(k)}\boldsymbol{x}'_{n+1}. \tag{22}$$

Using (2) gives the backward difference of kth order as

$$\delta^{(k)}\boldsymbol{x}'_{n+1} = \delta^{(k-1)}\boldsymbol{x}'_{n+1} - \alpha_{n+1}^{(k-1)}\,\delta^{(k-1)}\boldsymbol{x}'_n \tag{23}$$

—the 2nd term in the right side of (23) contains known sensitivities at t_n. Therefore, only the first term must be unrolled and such decreasing the order can continue until the sensitivity vectors at t_{n+1} and t_n are reached, i.e.,

$$\begin{aligned}\delta^{(k-1)}\boldsymbol{x}'_{n+1} &= \delta^{(k-2)}\boldsymbol{x}'_{n+1} - \alpha_{n+1}^{(k-2)}\,\delta^{(k-2)}\boldsymbol{x}'_n, \\ &\cdots \\ \delta^{(1)}\boldsymbol{x}'_{n+1} &= \underbrace{\delta^{(0)}\boldsymbol{x}'_{n+1}}_{\boldsymbol{x}'_{n+1}} - \alpha_{n+1}^{(0)}\underbrace{\delta^{(0)}\boldsymbol{x}'_n}_{\boldsymbol{x}'_n}.\end{aligned} \tag{24}$$

The equations (23) and (24) give the general formula

$$\delta^{(k)}\boldsymbol{x}'_{n+1} = \boldsymbol{x}'_{n+1} - \sum_{l=0}^{k-1}\alpha_{n+1}^{(l)}\,\delta^{(l)}\boldsymbol{x}'_n, \tag{25}$$

which is an analogy of the last part of (19). Now we can evaluate the terms in the right side of (22) by means of (25). For the first term ($k = 1$, i.e., $l = 0$ in (25)), we obtain

$$\frac{1}{t_{n+1} - t_n} \boldsymbol{x}'_{n+1} - \frac{1}{t_{n+1} - t_n} \alpha^{(0)}_{n+1} \delta^{(0)} \boldsymbol{x}'_n. \quad (26)$$

For the second term ($k = 2$, i.e., $l = 0$ or 1), we obtain

$$\frac{1}{t_{n+1} - t_{n-1}} \boldsymbol{x}'_{n+1} - \frac{1}{t_{n+1} - t_{n-1}} \times \left(\alpha^{(0)}_{n+1} \delta^{(0)} \boldsymbol{x}'_n + \alpha^{(1)}_{n+1} \delta^{(1)} \boldsymbol{x}'_n\right), \quad (27)$$

for the third term ($k = 3$, i.e., $l = 0$, 1, or 2), we obtain

$$\frac{1}{t_{n+1} - t_{n-2}} \boldsymbol{x}'_{n+1} - \frac{1}{t_{n+1} - t_{n-2}} \times \left(\alpha^{(0)}_{n+1} \delta^{(0)} \boldsymbol{x}'_n + \alpha^{(1)}_{n+1} \delta^{(1)} \boldsymbol{x}'_n + \alpha^{(2)}_{n+1} \delta^{(2)} \boldsymbol{x}'_n\right), \quad (28)$$

and, for the last term ($k = k_{n+1}$, i.e., $l = 0, \ldots, k_{n+1} - 1$), we obtain

$$\frac{1}{t_{n+1} - t_{n+1-k_{n+1}}} \boldsymbol{x}'_{n+1} - \frac{1}{t_{n+1} - t_{n+1-k_{n+1}}} \times \left(\alpha^{(0)}_{n+1} \delta^{(0)} \boldsymbol{x}'_n + \cdots + \alpha^{(k_{n+1}-1)}_{n+1} \delta^{(k_{n+1}-1)} \boldsymbol{x}'_n\right). \quad (29)$$

The grouping of the related terms in (26) to (29) gives the compact form of $\dot{\boldsymbol{x}}'_{n+1}$ (i.e., "a sensitivity algebraization"):

$$\begin{aligned}
\dot{\boldsymbol{x}}'_{n+1} = \boldsymbol{x}'_{n+1} & \sum_{k=1}^{k_{n+1}} \frac{1}{t_{n+1} - t_{n+1-k}} \\
& - \alpha^{(0)}_{n+1} \delta^{(0)} \boldsymbol{x}'_n \sum_{k=1}^{k_{n+1}} \frac{1}{t_{n+1} - t_{n+1-k}} \\
& - \alpha^{(1)}_{n+1} \delta^{(1)} \boldsymbol{x}'_n \sum_{k=2}^{k_{n+1}} \frac{1}{t_{n+1} - t_{n+1-k}} - \cdots \\
& - \alpha^{(k_{n+1}-1)}_{n+1} \delta^{(k_{n+1}-1)} \boldsymbol{x}'_n \underbrace{\frac{1}{t_{n+1} - t_{n+1-k_{n+1}}}}_{\sum_{k=k_{n+1}}^{k_{n+1}} \frac{1}{t_{n+1} - t_{n+1-k}}},
\end{aligned}$$

which is an unrolled form of (21)—all the above formulae are more flexible than possible ones that can be derived on the basis of Gear's method [10]. ■

A comparison with the first-order method Several classical CAD tools have used the simplest first-order formula

$$\left[\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)_{n+1} + \frac{1}{\Delta t_{n+1}} \left(\frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{x}}}\right)_{n+1}\right] \boldsymbol{x}'_{n+1} = - \left(\frac{\partial \boldsymbol{f}}{\partial p}\right)_{n+1} + \frac{1}{\Delta t_{n+1}} \left(\frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{x}}}\right)_{n+1} \boldsymbol{x}'_n \quad (30)$$

instead of (14). Comparing the efficiencies of (30) and (14) by means of the C.I.A. program [2], we obtain the CPU times (on Pentium IV/2.6 GHz/Windows 98 SE) in Tab. 1 for the sensitivity analyses of the circuit in Fig. 1—they clearly show the usefulness of the novel formula.

Relative interpolation error		10^{-6}	10^{-7}	10^{-8}
CPU time (s)	Formula (30)	49	64	65
	Formula (14)	34	38	38

Tab. 1. Comparison of the efficiency of the classical formula with the novel one.

References

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

[2] DOBEŠ, J. Reliable CAD analyses of CMOS radio frequency and microwave circuits using smoothed gate capacitance models, *AEÜ—International Journal of Electronics and Communications*, 2003, vol. 57, no. 6, p. 372 – 380.

[3] DOBEŠ, J. A modified Markowitz criterion for the fast modes of the LU factorization. In *Proceedings of the 48th Midwest Symposium on Circuits and Systems*. Cincinnati (Ohio, USA), 2005, in print.

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

[5] SALAMA, M. K., SOLIMAN, A. M. Low-voltage low-power CMOS RF four-quadrant multiplier. *AEÜ—International Journal of Electronics and Communications*, 2003, vol. 57, no. 1, p. 74 – 78.

[6] SHEU, B. J., SCHARFETTER, D. L., KO, P. K., JENG, M.-C. BSIM: Berkeley short-channel IGFET model for MOS transistors. *IEEE Journal of Solid-State Circuits*, 1987, vol. 22, no. 8, p. 558 – 566.

[7] CHENG, Y., HU, C. *MOSFET Modeling & BSIM3 User's Guide.* Boston: Kluwer Academic Publishers, 1999.

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

[9] LIU, W. *MOSFET models for SPICE simulation including BSIM3v3 and BSIM4.* New York: John Wiley & Sons, 2001.

[10] CHUA, L. O., LIN, P.-M. *Computer-Aided Analysis of Electronic Circuits.* Englewood Cliffs, New Jersey: Prentice-Hall, 1975.

About Author ...

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.