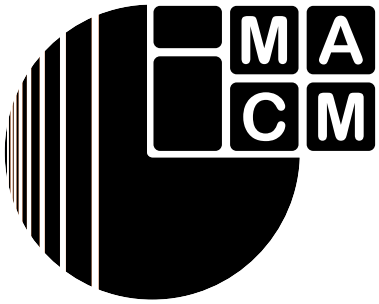

Bergische Universität Wuppertal

Fachbereich Mathematik und Naturwissenschaften

Institute of Mathematical Modelling, Analysis and Computational Mathematics (IMACM)

Preprint BUW-IMACM 12/34

B. Tasić, J.J. Dohmen, E.J.W. ter Maten, T.G.J. Beelen, W.H.A. Schilders, A. de Vries, M. van Beurden

Robust DC and efficient time-domain fast fault simulation

December 2012

http://www-num.math.uni-wuppertal.de

Robust DC and efficient time-domain fast fault simulation

B. Tasić* J.J. Dohmen* E.J.W. ter Maten†,‡ T.G.J. Beelen*
W.H.A. Schilders† A. de Vries§ M. van Beurden*

December 2012

Abstract

Purpose – Imperfactions in manufacturing processes may cause unwanted connections (faults) that are added to the nominal, "golden", design of an electronic circuit. By fault simulation one simulates all situations. Normally this leads to a large list of simulations in which for each defect a steady-state (DC) solution is determined followed by a transient simulation. We improve the robustness and the effciency of these simulations.

Design/methodology/approach – Determining the DC solution can be very hard. For this we present an adaptive time domain source stepping procedure that can deal with controlled sources. The method can easily be combined with existing pseudo-transient procedures. The method is robust and efficient.
In the subsequent transient simulation the solution of a fault is compared to a golden, fault-free, solution. A strategy is developed to efficiently simulate the faulty solutions until their moment of detection.

Finding – We fully exploit the hierarchical structure the circuit in the simulation process to bypass parts of the circuit that appear to be unaffected by the fault. Accurate prediction and efficient solution procedures lead to fast fault simulation.

Originality/value – Our fast fault simulation helps to store a database with detectable deviations for each fault. If such a detectable output "matches" a result of a product that has been returned because of malfunctioning it helps to identify the subcircuit that may contain the real fault. One aims to detect as much as possible candidate faults. Because of the many options the simulations must be very efficient.

Keywords Source stepping, Pseudo transient, Fast fault simulation, Low-rank approximations, Hierarchical simulation, Bypassing, Sensitivity analysis

Paper type Research paper

*NXP Semiconductors, Eindhoven, the Netherlands
†Technische Universiteit Eindhoven, the Netherlands
‡Bergische Universität Wuppertal, Germany
§NewHer Systems, Helmond, the Netherlands

1 Introduction

Imperfactions in manufacturing processes may cause unwanted connections (faults) that are added to the nominal, "golden", design. Here we only consider bridges, linear resistors, as faults. There can be a large variation in location and in resistor value. For each faulty circuit (each with only one fault) one wants to determine if its solution differs from the golden solution. Then the fault is detectable. To reduce the overal time-domain simulation we are interested in fast fault simulation.
The electronic circuit equations can be written as [6, 14]

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q}(\mathbf{x}) + \mathbf{j}(\mathbf{x}) \quad = \quad \mathbf{s}(t, \mathbf{x}). \tag{1}$$

Here $\mathbf{s}(t, \mathbf{x})$ represents the specifications of the sources. The unknown $\mathbf{x} = \mathbf{x}(t)$ consists of nodal voltages and of currents through voltage defined elements. We assume that $\mathbf{q}(\mathbf{0}) = \mathbf{0}$, and $\mathbf{j}(\mathbf{0}) = \mathbf{0}$.
The steady-state solution, which is called DC-solution (Direct Current solution), $\mathbf{x}_{\mathrm{DC}}$, satisfies

$$\mathbf{j}(\mathbf{x}_{\mathrm{DC}}) \quad = \quad \mathbf{s}(0, \mathbf{x}_{\mathrm{DC}}). \tag{2}$$

Usually the DC-solution provides the initial value for the transient problem (1). The importance of the DC-problem lies in the fact that the DC-solution is crucial as starting solution for a number of next analyses (transient analysis, AC analysis, Harmonic Balance analysis, Periodic Steady-State analysis). In general, the problem (2) is non-linear and in several cases it is hard to solve. How to solve this problem will be described in Section 2.
In general, (1) forms a system of Differential-Algebraic Equations (DAEs). We assume that the DAE has at most differentiability-index 1 [6].
In [5, 6] methods for the time integration of the circuit equations (1) are discussed. In Section 3 we will give some more details. However, our paper will restrict mostly to Euler-Backward time integration.
In Section 3 we will consider the problem how to efficiently perform time integration of (1) when we add a resistors between nodes that may be chosen randomly. For each resistor value several values may be taken. Also we may encounter a large sequence of resistors. For each occurring resistor value we have to determine if the corresponding solution defers from the faulty-less solution. That being the case means that the fault is detectable. If such a detectable output "matches" a result of a product that has been returned because of malfunctioning it helps to identify the subcircuit that may contain the real fault. One aims to detect as much as possible candidate faults. Because of the many options the simulations also must be very efficient.

2 Solving the DC-problem

To solve the equations (2) Newton's method, or variants, may be applied [3, 6, 14], which can be combined with $g_{\min}$-stepping, in which linear conductors g are placed parallel

to the non-linear part inside each transistor (device). Iteratively $g \downarrow g_{\min}$, after which the Newton iteration counter is increased. Another approach is Pseudo-Transient [2]. In Pseudo-Transient (PT) one can use relaxed tolerances for the Newton process and for the time step control procedure. Also this can be combined with $g_{\min}$-stepping during each time step. In PT one has to provide a nontrivial initial solution. This has led to a strategy "$A \to B \to C \to D$", to sequentially try a new method when the previous process was not successful. Here the methods are A : Newton without $g_{\min}$-stepping; B : Newton with $g_{\min}$-stepping; C : PT without $g_{\min}$-stepping; D : PT with $g_{\min}$-stepping. In general the next process is more robust than the previous one, but it also needs more cpu.
A new procedure, Source Stepping by Pseudo Transient (SSPT), is decribed in the next section 2.1. It modifies existing Pseudo Transient methods [2], [3, Section 6.4] in the case of controlled elements. Other methods are: temperature stepping, source stepping (SS, the sources are iteratively increased to their final value), homotopy methods, or optimization [1,4,8,13,14]. In [13] a good example is shown for a simple circuit in which default homotopy fails.

2.1 Time-domain Source Stepping

Usually, in Source Stepping (SS) one introduces a parameter λ and considers the problem

$$\mathbf{j}(\mathbf{x}(\lambda)) + \lambda \mathbf{s}(0, \mathbf{x}(\lambda)) \quad = \quad 0. \tag{3}$$

In this case it is assumed that for $\lambda = 0$ the problem (3) is easily solved so that in the end the original problem is solved. The same parameter λ is applied to all sources s in the circuit. In general, for each value of λ a nonlinear problem has to be solved. In principle one can offer a sequence of problems to the simulator to mimic this process. The method is indicated in Fig. 1 (left). One sequentially solves the problem for $\lambda = j/N$, $j = 0, 1, \ldots, N$, continuing from the solution of the previous step.
We introduce a time-domain variant (SSPT, Source Stepping by Pseudo Transient) that offers an automatic continuation process, based on PT and adapting the transient stepsize and the λ stepsize at the same time. We define a time $t = T$ at which we want to have solved the original DC-problem. We also introduce a time $T_\alpha = \alpha T$ (by default $\alpha = 0.5$) at which ordinary PT will start simulation using the sources as in the original DC-problem, i.e., using $\lambda = 1$ and where PT integrates from T_α to T', where $T' \leq T$ is the point where all transient effects have become negligible (see also Fig. 1, right).
On the interval $[0, T_\alpha]$, a special PT integration is performed with the function $\lambda(t) = t/T_\alpha$. Hence, at each time step, also the actual applied source values change. The interval $[0, T_\alpha]$ is the switch-on interval, the interval $[T_\alpha, T]$ is the interval to damp-out transient effects. On both intervals PT uses an automatic time step determination procedure. On the interval $[T_\alpha, T]$ an ordinary PT procedure is executed. Hence, if, at some time point, the Newton iterative process does not converge, a re-integration will be done with a smaller stepsize.

Recursion in controlled sources asks for a modification in (3), because in (3), λ may

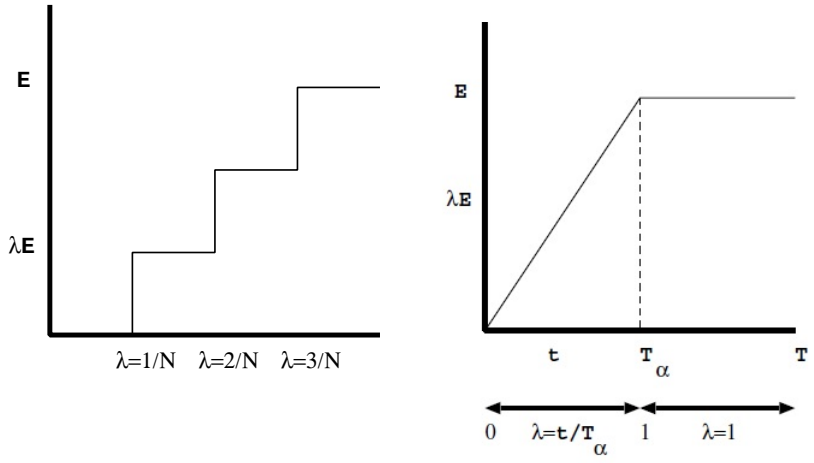

Figure 1: **Left**: Source Stepping (SS): One sequentially solves the problem for $\lambda = j/N$ $(j = 0, \ldots, N)$, continuing from the final solution of the previous step. **Right**: Source Stepping by Pseudo Transiient (SSPT). On $[0, T_\alpha]$ a time-dependent voltage source $\lambda(t)\mathbf{E}$ is used where $\lambda(t) = t/T_\alpha$. On $[T_\alpha, T]$ we have $\lambda \equiv 1$. Here for controlled sources we propose the modification (5)-(6).

affect the Newton-Raphson matrix when solving the non-linear systems. To derive a better suited method we write the expression for an applied source value $V(E_1)$ (say) as

$$V(E_1) \quad = \quad \psi(\mathrm{ev}_1, \mathrm{ev}_2, \ldots, \mathrm{ev}_n). \tag{4}$$

As value during the source stepping at time t on $[0, T_\alpha]$ we propose to take

$$V(E_1) \quad = \quad \tilde{\psi}(\mathrm{ev}_1, \ldots, \mathrm{ev}_n, t), \quad \text{where} \tag{5}$$

$$\tilde{\psi}(\mathrm{ev}_1, \ldots, \mathrm{ev}_n, t) \quad = \quad \psi(\mathrm{ev}_1, \ldots, \mathrm{ev}_n) + (\lambda(t) - 1)\psi(0, \ldots, 0). \tag{6}$$

Note that $\psi(0, \ldots, 0)$ has to be calculated once. Now λ will not show up in the matrix, also in case of recursive ψ definitions. Clearly, for $\lambda = 0$ the applied voltage is zero (i.e., assuming starting from the zero solution, which implies that all ev's are zero), which makes the zero solution the exact solution. When $\lambda = 1$ the original voltage expression is used. Since our equations (1) are DAEs we remark that for all t the generated solution is consistent for the problem at hand. Because of the switch-on and the damp-out phase the whole process mimics a real physical process. For an impression of the test results we refer to Section 4.1. As will be shown there, the SSPT is robust, but also quite efficient.

3 Fast Fault Simulation

For time integration in circuit simulation one usually applies the Backward-Differentiation Formulas (BDF) methods, or the Trapezodial Rule (TR), or a combination of

BDF2 and TR. For a discussion on additional method see [5]. Here, for simplicity, we just apply BDF1, being Euler Backward. We introduce some notation and recall some general approaches in solving and stepsize control. Assuming time points $t_{k+1} = t_k + h_k$ ($k \geq 0$) with stepsizes h_k and approximation $\mathbf{x}^n$ at t_n, BDF1 calculates $\mathbf{x}^{n+1}$ by

$$\frac{\mathbf{q}^{n+1} - \mathbf{q}^n}{h_n} + \mathbf{j}^{n+1} \quad = \quad \mathbf{s}^{n+1}. \tag{7}$$

Here $\mathbf{q}^k = \mathbf{q}(\mathbf{x}^k, t_k)$, $\mathbf{j}^k = \mathbf{q}(\mathbf{j}^k, t_k)$, for $k = n, n+1$. The source we simplified to $\mathbf{s}^{n+1} = \mathbf{s}(t_{n+1})$. The system is solved by a Newton-Raphson procedure. A fixed Jacobian can reduce the number of LU-decompositions, but may increase the number of iterations and thus the number of (costly) evaluations. In these cases the assembly of the matrices is not much more effort. Here one may prefer to make a new LU-decomposition and (in case of an hierarchical linear solver) to profit from hierarchical bypassing [5], which will be exploited in Section 3.

Improvements in the time-domain integrations, after starting with a proper $\mathbf{x}_{\mathrm{DC}}$, have been tuned to fault analysis. To improve the analog and mixed-signal test development defect-oriented testing has been introduced [7,12,16,17]. It consists of three major activities: fault extraction from the layout, transistor-level fault simulation and test generation. We are concerned with the second part. Here we concentrate on linear bridges: resistors that bring in new connections. The list of possible bridges is large. In addition, the resistor values may be taken from a series of values. For the reference fault simulator, called DOTSS (Defect-Oriented Test Simulation System) [16], existing circuit simulators are used. Here every fault is simulated independently without taking advantage of the similarities between faulty and fault-free circuits. Also the entire time interval is simulated for each each fault. As a consequence, analog fault simulation of large mixed-signal circuits is a major bottleneck in analog and mixed-signal testing. At specific measurement time points the outcomes are stored in a database that can be used to determine the fault coverage of the test set. Several researchers have proposed time-efficient fault simulation methods. The work reported in [9] reduces the order of the fault matrix and improves the state prediction for the faults. This is also what we aim for, but the methods we use are very different.

3.1 Simultaneous Fault Simulation

The standard way of performing the Fault Simulation (until now) is performing a series of sequential transient simulation runs. The runs are unrelated, i.e. every fault transient run simulates the entire time interval without any reuse of the knowledge obtained from the golden and previous fault runs. The main idea of the Fast Fault Simulation is the reuse of the already obtained simulation data, especially from the golden run. Topological differences between a fault and the golden circuit description are minor and local (one resistor of difference only). Moreover, there are no additional unknowns introduced (new nodes, currents, etc.), which makes matrix dimensions unchanged. Hence, in the Fault Simulation it is reasonable to assume that most of the signals in the majority of runs, during most of the time interval, are approximately the same. This can be exploited

numerically to reduce the most expensive (in time) procedures for obtaining the solution. To achieve this, we propose a parallel run, i.e. a transient simulation in which large number of fault transient runs (including the golden one) are performed simultaneously, using the same time step mechanism. This would allow us to reuse most of the results and data structures (obtained for the golden run) and significantly improve the overall simulation time. At a particular time point we can reuse generated matrices and even

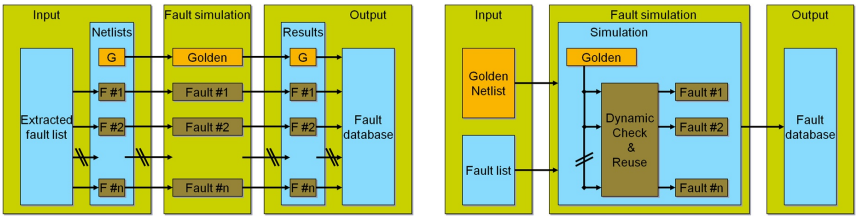

Figure 2: **Left**: Standard sequential approach for the reference fault simulator DOTSS (Defect-Oriented Test Simulation System). **Right**: Parallel approach with reuse of computed data for fast fault simulation.

their decomposed versions for all circuit models which signals do not differ much from the golden run. We can apply the Sherman-Morrison-Woodbury formula [10] to update the inverse matrix, which improves the efficiency of the solver. Ideally, if a particular fault result is almost the same as the golden run, we could obtain the solution almost instantly (almost no additional efforts required). In situations where a fault result differs significantly from the golden one, they can be computed separately, simply removing that particular run from the group computed in parallel. The ingredients we exploit are:

- The submodel that contains a bridge is known.
- An extra port for submodels makes it possible to include bridges between submodels.
- Detection of an error is only done at special time points.
- All simulations use the same maximum time step thats was used by the golden solution.
- We extensively apply hierarchical localization of changes in order to exploit bypassing.
- We re-use the golden solution to provide predictions.
- We apply the Sherman-Morrison-Woodbury formula [10] (in an hierarchical form) to efficiently solve the systems for each faulty solution.
- We abandon a fault simulation when we have detected a significant deviaton from the golden solution.
- We improve the predictions by applying sensitivity techniques.

Here we will only describe the hierarchical steps and the sensitivity in some details.

3.2 Partitioning of the system

We assume that the circuit is defined in a hierarchical way.The linear solver in our circuit simulator[1] fullly exploits the hierarchical structure; for a description see [5]. A

[1]Pstar: inhouse analog circuit simulator of NXP Semiconductors

graphical impression of an hierarchical organization is shown in Fig. 3 (left). At each level the performance of a submodel M_{ijk} depends on the terminal values provided by its parent model M_{ij}. It gives back its contribution to the equatons at the parent model. Thus, normally, terminal voltages are provided and terminal currents are obtained. The whole system is solved by a Newton solver. Hence contributions to the nonlinear residual function as well as to its Jacobian matrix have to be assembled from each model. This is done by a hierarchical recursion. At each model a local solution vector, and a local right-hand side vector and a local matrix is available. These involve (block) parts for the terminal unknowns and for the internal unknowns. Assembly of equations and of the

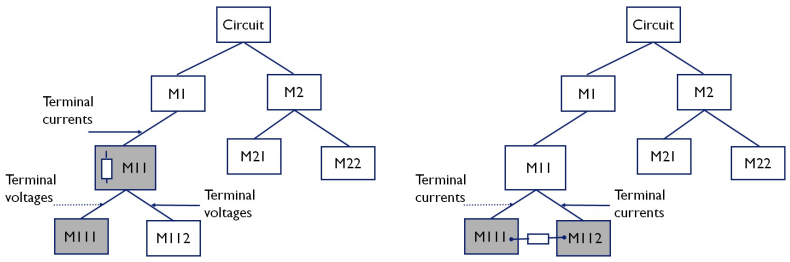

Figure 3: **Left**: Hierarchical structure of a circuit with a linear fault resistor in submodel M_{11}. **Right**: A circuit with a fault resistor bridging the two submodels M_{111} and M_{112}.

Jacobian matrix is done in a bottom-up recursion (that starts from the top). For each model, additionally, partial LU-decomposition of the matrix is done, thus eliminating the internal unknowns from the contributions to the right-hand side vector of the equations for the terminal unknowns at the parent model. Now the contents of the block part of the matrix and right-hand side vector of the child model is added to the proper corresponding places in the parent model. What is left after the full recursion is a linear system with a coefficient matrix that has a triangle form. Starting from the terminal value at the top circuit model (which is the ground value) a top down recursion provides a solution at all local models. Note that the solution vectors become 'consistent': at the terminals the values are equal when seen from the parent models or when seen from the child model. The right-hand side vectors are not consistent. for more details, see [5].
In Fig. 3, at the left a fault resistor in submodel M_{11} is shown, which gives a new, rank-one contribution $\frac{1}{R}\,\mathbf{u}\mathbf{u}^T$, to the matrix, when compared to matrix for the fault free problem. At the right the situation looks a bit more complicated since we now have a bridge between two submodels that are in different sub-trees of the hierarchy. To deal with this we extended the list of termininals with an additional unknown for communication. The local contribution is now $\frac{1}{R}\,\mathbf{u}\mathbf{v}^T$ for suitable vectors $\mathbf{u}$ and $\mathbf{v}$. The hierarchical assembly lifts the local contributions to the right-hand side vector to those at each higher level until the level of the submodel where the two hierarchical sub-trees meet.
In both cases it fits the general hierarchical block structure but it may require new local entries, also in the situation at the left in Fig. 3. For this reason we exploit the

Sherman-Morrison-Woodbury formula.

3.3 Hierarchical bypassing

The signals corresponding to a fault are obtained by simulating the fault-injected representation of the fault-free circuit. The fault injection can cause a faulty circuit's operating region to differ radically from that of a fault-free circuit. However, it often happens that the impact of the fault is just local and large parts of the circuit stay very close to the fault-free circuit. In this case the affected equations have to be solved, but the solution of the other equations can simply be skipped and the golden solution of the fault-free circuit can be reused for these equations.
If the impact of the fault is only local, then only the equations of the affected models have to be solved and the other models can simply be bypassed. The bypassing algorithm is described by the following steps

1. Flag the model containing the fault.
2. The unknowns of the flagged models are solved for.
3. Check the right-hand side contributions (the "terminal currents") of the flagged models to the right-hand side vector of each one's parent model.
4. Flag the parent model if the difference in the terminal currents between the golden solution and those of the faulty solution is larger than a numerical error tolerance of the simulator.
5. Check the terminal voltages of the sub-models in each flagged model.
6. Flag the sub-model if the terminal voltage difference between the golden solution and the faulty solution is larger than a numerical error tolerance of the simulator.
7. Repeat step 2 until all terminal voltages and currents of the bypassed models have smaller difference than the tolerance.

In [9] bypassing is also used in a similar way as described in this document. The main difference is that we use the hierarchy to reduce the faulty MNA matrix, while in [9] the equations are projected onto a smaller subspace to reduce computation cost. As an example consider the circuit at the left of Fig. 3 and we use the colouring. In step 2 the unknowns of the model M_{11} are solved for. In step 3 the terminal current of M_{11} are checked. We assume that the change of this current in step 4 is smaller than the given tolerance, so model M_1 can still be bypassed. In step 5 the terminal voltages of model M_{111} and of model M_{112} are checked. Suppose that the terminal voltages of model M_{111} have changed significantly, but that the changes in the terminal voltages of model M_{112} are still smaller than the given tolerance. Because of this difference in terminal voltages model M_{111} is flagged in step 6 to be solved. In the next iteration the unknowns of both models M_{11} and M_{111} will be solved for. All signals at the terminals of M_{11} and M_{112} are now approximately the same as in the golden run, so the solution is accepted without ever solving the other unknowns.

3.4 Exploiting sensitivity analysis

In our fault analysis we consider linear bridges only, hence for each fault we have a new contribution $p\,\mathbf{u}\mathbf{v}^T\mathbf{x}(t,p)$ as low-rank modification to the system of the golden solution; thus $\mathbf{j}(\mathbf{x}(t,p),p) = \mathbf{j}_0(\mathbf{x}(t,p)) + p\,\mathbf{u}\mathbf{v}^T\mathbf{x}(t,p)$. Here $p = \frac{1}{R}$ is just a scalar, by which the $\mathbf{p}$-sensitivity 'matrix' $\hat{\mathbf{x}}_p(t,p) = \frac{\partial \mathbf{x}(t,p)}{\partial p}$ becomes a vector. We assume Euler Backward time integration between time points t_n and $t_{n+1} = t_n + h_n$ (with time steps h_n). Let $\mathbf{x}_p^k = \mathbf{x}^k(p) \approx \mathbf{x}(t_k,p)$ be the numerical approximations for $k = n, n+1$ and $\hat{\mathbf{x}}_p^k$ be the corresponding sensitivities. Then with $\mathbf{C}_p^k \equiv \frac{\partial \mathbf{q}(\mathbf{x}_p^k)}{\partial \mathbf{x}}$ and $\mathbf{G}_p^k \equiv \frac{\partial \mathbf{j}_0(\mathbf{x}_p^k)}{\partial \mathbf{x}}$ by sensitivity analysis [11, 15] we obtain

$$[\frac{1}{h_n}\mathbf{C}_p^{n+1} + \mathbf{G}_p^{n+1}]\hat{\mathbf{x}}_p^{n+1} \;=\; -\mathbf{u}\mathbf{v}^T\mathbf{x}_p^{n+1} + \frac{1}{h_n}\mathbf{C}_p^n\hat{\mathbf{x}}_p^n. \tag{8}$$

For $p = 0$, (8) gives the limit sensitivity $\hat{\mathbf{x}}^k = \hat{\mathbf{x}}_0^k$ for the golden, fault-free, solution $\mathbf{x}^k = \mathbf{x}_0^k$ $(k = n, n+1)$

$$[\frac{1}{h_n}\mathbf{C}^{n+1} + \mathbf{G}^{n+1}]\hat{\mathbf{x}}^{n+1} \;=\; -\mathbf{u}\mathbf{v}^T\mathbf{x}^{n+1} + \frac{1}{h_n}\mathbf{C}^n\hat{\mathbf{x}}^n, \tag{9}$$

where $\mathbf{C}^k = \mathbf{C}_0^k$ $(k = n, n+1)$ and $\mathbf{G}^{n+1} = \mathbf{G}_0^{n+1}$. By Taylor expansion we additionally have

$$\mathbf{x}_p^k \;=\; \mathbf{x}^k + p\,\hat{\mathbf{x}}^k + \mathcal{O}(p^2) \quad (k = n, n+1). \tag{10}$$

The golden solution satisfies the linearized equations of the fault-free circuit

$$[\frac{1}{h_n}\mathbf{C}^{n+1} + \mathbf{G}^{n+1}]\mathbf{x}^{n+1} \;=\; \mathbf{s}^{n+1} + \frac{1}{h_n}\mathbf{C}^n\mathbf{x}^n. \tag{11}$$

With (10) and (9) this gives

$$\begin{aligned}
[\frac{1}{h_n}\mathbf{C}^{n+1} + \mathbf{G}^{n+1}]\mathbf{x}_p^{n+1} &= p\,[\frac{1}{h_n}\mathbf{C}^{n+1} + \mathbf{G}^{n+1}]\hat{\mathbf{x}}^{n+1} + \mathbf{s}^{n+1} + \frac{1}{h_n}\mathbf{C}^n\mathbf{x}^n + \mathcal{O}(p^2),\\
&= -p\,\mathbf{u}\mathbf{v}^T\mathbf{x}^{n+1} + \frac{1}{h_n}\mathbf{C}^n(p\,\hat{\mathbf{x}}^n + \mathbf{x}^n) + \mathbf{s}^{n+1} + \mathcal{O}(p^2),\\
&= -p\,\mathbf{u}\mathbf{v}^T\mathbf{x}^{n+1} + \mathbf{s}^{n+1} + \frac{1}{h_n}\mathbf{C}^n\mathbf{x}_p^n + \mathcal{O}(p^2),\\
&= -p\,\mathbf{u}\mathbf{v}^T\mathbf{x}_p^{n+1} + \mathbf{s}^{n+1} + \frac{1}{h_n}\mathbf{C}^n\mathbf{x}_p^n + \mathcal{O}(p^2), &(12)\\
&= -p\,\mathbf{u}\mathbf{v}^T\mathbf{x}_p^{n+1} + \mathbf{s}^{n+1} + \frac{1}{h_n}\mathbf{C}^n\mathbf{x}^n + \mathcal{O}(p). &(13)
\end{aligned}$$

We just used (13). Note that (12) may be a more accurate alternative. Hence

$$\{[\frac{1}{h_n}\mathbf{C}^{n+1} + \mathbf{G}^{n+1}] + p\,\mathbf{u}\mathbf{v}^T\}\,\mathbf{x}_p^{n+1} \;=\; \mathbf{s}^{n+1} + \frac{1}{h_n}\mathbf{C}^n\mathbf{x}^n + \mathcal{O}(p), \tag{14}$$

which, after ignoring the $\mathcal{O}(p)$ term at the right-hand side, is similar to the equation (11) for the linearized, golden, circuit equations, except for the left-hand side, which now contains the conductance of the fault. This leads to a rank-1 update of the golden circuit matrix for which we again apply the Sherman-Morrison-Woodbury formula.
The advantage of the right-hand side in (14) is that it is independent of the solution $\mathbf{x}_p^k$ at the previous time steps. Of course, when followed by further Newton-Raphson iterations, $\mathbf{x}_p^n$ is still needed. To judge the accuracy of the linear sensitivity prediction the nonlinear solver evaluates the circuit at the sensitivity solution and updates the solution. The difference in the initial sensitivity solution and the nonlinear update is a measure for the truncation error.
If we just stick to the prediction, we may calculate the prediction of the fault at a few selected time points, which significantly reduces the work load for the fault sensitivity analysis.

4 Results

4.1 Results DC

We tested the SSPT on a set of problems where parameters were swept: temperature (problems 1 and 5), and statistics (problems 2-4 and 6). The options A and B stand for Newton-Raphson (resp., without and with $g_{\min}$ stepping). The options C and D stand for Pseudo Transient (resp., without and with $g_{\min}$ stepping). If an option results in convergence, the next ones are not tried anymore. In Fig. 4 the $A \rightarrow B \rightarrow C \rightarrow D$ (ABCD) process is compared to Source Stepping (SS) and to Source Stepping by Pseudo Transient (SSPT). For the first process and for SS one continues on the solution of the previous case. SS internally uses ABCD for each step. SSPT always starts from 0. SSPT clearly is the most robust method: it could solve all problems. Figure 4 (right) shows the speed-up offered by SSPT when compared to the ABCD process. It was 0.9-13 times faster, in addition to its higher robustness.

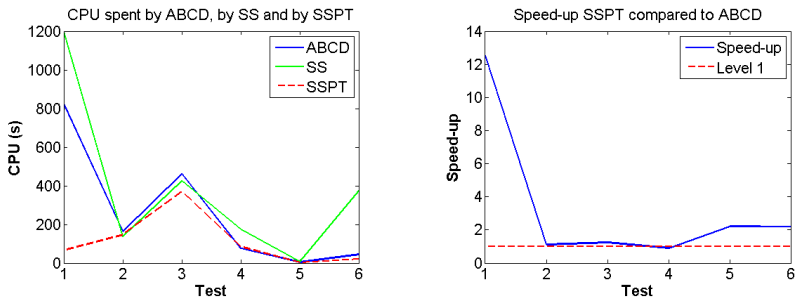

Figure 4: **Left**: CPU outcomes spent by the ABCD process and by SS and by SSPT, respectively. **Right**: Speed-up offered by SSPT when compared to the ABCD process.

4.2 Results FFS with sensitivity prediction

Starting from the solution of the linear sensitivity prediction (14) the nonlinear solver tries to find the faulty solution. Table 1 shows that for a LIN Converter IP Block this new prediction already approximates quite well and that it can be improved by continuing iterations. Here we see significant improvement in the accuracy, while the increase of the CPU time is not as big. The limits used for detectability are determined from the 10σ statistical range. The CPU time of the golden stand-alone simulation is 244s. The CPU time of the Standard AS/DOTSS approach [16] is calculated in the usual way (#faults * CPU time of golden = 412 * 244s = 100437s). The CPU time mostly depends on how many measurement time points are set. This IP Block has 17 in total (divided in two transient runs). The matching seems to converge by the increase of the number of iterations, which gives us a way to judge how good is the sensitivity analysis without running standard AS/Dotss or standard FFS to check. A second circuit that was investigated

Analysis (#faults 412)	**#iter. per Δt**	**#Faults Detected**	**Matching DOTSS**	**CPU [sec]**	**Speed up**
Standard AS/DOTSS	-	220 (53.4%)	-	100437	1
Linear Sensitivity	0	117 (28.4%)	71.6%	458	219
Nonlinear Correction	5	207 (50.2%)	89.1%	2341	43
Nonlinear Correction	30	221 (53.6%)	93.5%	4565	22
Nonlinear Correction	100	218 (52.9%)	93.7%	7088	14

Table 1: Fault coverage results and speed up by including sensitivity prediction for the LIN Converter IP Block.

Analysis (#faults 100)	**#iter. per Δt**	**#Faults Detected**	**Matching DOTSS**	**CPU [sec]**	**Speed** bf up
Standard AS/DOTSS	-	45	-	52513	1
Linear Sensitivity	0	51	88%	916	58
Nonlinear Correction	1	54	89%	4808	11
Nonlinear Correction	2	54	89%	5512	10

Table 2: Fault coverage results and speed up by sensitivity prediction for the Control DAC.

is the Control DAC circuit (a 10-bit Digital-to-Analog Converter). The results for the first case are listed in Table 2. The speedup is lower than in the previous case of the LIN Converter IP Block (but still quite appreciable). This is mainly caused by the much larger number of measurement time points that are used for the Control DAC. The linear sensitivity shows a good fault coverage match with Standard AS/DOTSS, but this match is not improved substantially by the nonlinear corrections.

5 Conclusion and Outlook

We have improved robustness in solving for the DC-solution. The new method SSPT, Source Stepping by Pseudo Transient, treats controlled sources in a novel way that guarantees that good properties of Jacobian matrices are maintained. The method is also remarkably efficient.
For Fast Fault Simulation we considered the fast simulation of a large number of faulty solutions after adding linear bridges to the golden circuit. Several options to obtain efficiency have been implemented. Here especially the power of using sensitivity analysis was demonstrated.
Extensions are now directed to further increase the fault coverage, while maintaining efficiency, to also include linear capacitors as fault and to apply techniques to Monte Carlo simulations.

References

[1] E.L. Allgower, K. Georg (1997), 'Numerical path following'. In: P.G. Ciarlet, J.L. Lions (Eds.), *'Handbook of Numerical Analysis'*, Vol. 5, Elsevier BV, North-Holland, 3–207.

[2] T.S. Coffey, C.T. Kelley, D.E. Keyes (2003), 'Pseudotransient continuation and differential-algebraic equations', *SIAM J. Sci. Comput.*, **25-2**, 553–569.

[3] P. Deuflhard (2004), *'Newton methods for nonlinear problems – Affine invariance and adaptive algorithms'*, Springer-Verlag, Berlin.

[4] A. Dyess, E. Chan, H. Hofmann, W. Horia, L. Trajkovic (1999), 'Simple implementations of homotopy algorithms for finding DC solutions of nonlinear circuits', *Proc. ISCAS 1999 Vol. VI*, 290–293.

[5] J.G. Fijnvandraat, S.H.M.J. Houben, E.J.W. ter Maten, J.M.F. Peters (2006), 'Time domain analog circuit simulation', *J. of Comput. and Applied Maths.*, **185**, 441–459.

[6] M. Günther, U. Feldmann, J. ter Maten (2005), 'Modelling and discretization of circuit problems'. In: W.H.A. Schilders, E.J.W. ter Maten (Eds.): *'Handbook of Numerical Analysis, Vol. XIII, Special Volume on Numerical methods in electromagnetics'*, Elsevier BV, North-Holland, 523–659.

[7] H. Hashempour, J. Dohmen, B. Tasić, B. Kruseman, C. Hora, M. van Beurden, Y. Xing (2010), 'Test time reduction in analogue/mixed-signal devices by defect oriented testing: An industrial example', *Proc. DATE 2010*, 371–376.

[8] M. Honkala, J. Roos, V. Karanko (2006), 'On nonlinear iteration methods for DC analysis of industrial circuits'. In: A. Di Bucchianico, R.M.M. Mattheij, M.A. Peletier (Eds.): *'Progress in industrial mathematics at ECMI 2004'*, Springer-Verlag, Berlin, 144–148.

[9] J. Hou, A. Chatterjee (2003): 'Concurrent transient fault simulation for analog circuits', *IEEE Trans. on Comp.-Aided Design of Integr. Circuits and Systems (TCAD)*, **22-10**, 1385–1398.

[10] A.S. Householder (1957), 'A survey of some closed methods for inverting matrices', *SIAM J. Appl. Math.*, **5-3**, 155–169.

[11] Z. Ilievski, H. Xu, A. Verhoeven, E.J.W. ter Maten, W.H.A. Schilders, R.M.M. Mattheij (2007), 'Adjoint transient sensitivity analysis in circuit simulation'. In: G. Ciuprina, D. Ioan (Eds.): *'Scientific Computing in Electrical Engineering SCEE 2006'*, Series Mathematics in Industry 11, Springer, 183–189.

[12] B. Kruseman, B. Tasić, C. Hora, J. Dohmen, H. Hashempour, M. van Beurden, Y. Xing (2011), 'Defect oriented testing for analog/mixed-signal devices', *Proc. 2011 IEEE Int. Test Conference (ITC 2011)*, 10p.

[13] W. Mathis, L. Trajkovic, M. Koch, U. Feldmann (1995), 'Parameter embedding methods for finding DC operating points of transistor circuits'. *Proc. NDES-1995, Dublin*, 147–150.

[14] J. Ogrodzki (1994), *'Circuit simulation methods and algorithms'*, CRC Press, Boca Raton, FL, USA.

[15] L. Petzold, S.T. Li, Y. Cao, R. Serban (2006), 'Sensitivity analysis of differential-algebraic equations and partial differential equations', *Computers and Chemical Engineering*, **30**, 1553–1559.

[16] Y. Xing (1998), 'Defect-oriented testing of mixed-signal ICs: some industrial experience', *Proc. IEEE International Test Conference ITC'98*, 678–687.

[17] V.A. Zivkovic, F. van der Heyden, G. Gronthoud, F. de Jong (2008), 'Analog test bus infrastructure for RF/AMS modules in core-based design', *13th European Test Symposium, 25-29 May 2008*, 27–32.