Model Order Reduction using SPICE Simulation Traces

Paul Winkler, Henda Aridhi, and Sofiène Tahar

Department of Electrical and Computer Engineering,
Concordia University, Montreal, Canada
pauwink@web.de, h_aridh@ece.concordia.ca, tahar@ece.concordia.ca

Technical Report

December 1, 2013

Abstract

The generation of fast models for device level circuit descriptions is a very active area of research. Model order reduction is an attractive technique for reducing the computational cost of dynamical models simulation. In this work, we propose an approach based on clustering, curve-fitting, linearization and Krylov space projection to build reduced models for nonlinear analog circuits. We demonstrate our model order reduction method for three nonlinear circuits: a voltage controlled oscillator, an operational amplifier and a digital frequency divider. Our experimental results show that the reduced models lead to an improvement in simulation speed while providing the same behavior of the original circuit design.

1 Introduction

Large electronic circuits are very complicated systems. Due to their usual large size and strong nonlinear characteristics, the only way to analyse them is a computer simulation. Simulation programs with integrated circuit emphasis (SPICE) are used widely to analyse or design electronic circuits. The elements of these circuits are usually integrated on a single semiconductor layer of an integrated circuit (IC) or as single devices on a printed circuit board (PCB). The continuously improving technology enables the production of smaller and smaller devices and their implementation on a single IC. Following the Moores law, the number of elements per IC grows exponentially and can reach today 10^9 and more. This trend is predicted to continue in the next decades [1]. Also the numerical solvers, which are used to compute the behaviour of circuits, e.g., in SPICE programs, are very advanced, the high number of elements in a large circuit forces the simulation to slow down. This increases development time and costs for the design of new circuits.

Model order reduction (MOR) techniques, which are able to reduce the size of a dynamical system description, were recently applied to find smaller and faster models of electronic circuits [2, 3]. The most promising MOR approaches for nonlinear circuits use the time traces, resulting from a circuit transient simulation, to find linearization points of an already elaborated mathematical model of the circuit [4, 5, 6]. This report investigates, whether the time traces of a former transient simulation of an electronic circuit can be used to find the mathematical description of the circuit, by using curve-fitting. The elaborated mathematical model can be used, if the parameters of the original system are partially unknown or the device models are to complex to elaborate their mathematical model analytically. The new model, which is elaborated by curve-fitting, is a simplified approximation of the original circuit model in its original state space. It will contain less details than the original description of the circuit, which is enclosed in a SPICE netlist. For the reduction of the system, a Krylov algorithm is used. The simulation of the generated, reduced models should be faster than the simulation of the original models. Also this models are a modification, it is essentially, that they accurately show the behaviour of the original model. It has to be explored, whether the generation of reduced analog circuit models can be automated, by using their time traces, and whether these models are an alternative to conventional circuit models.

Section 3 explains briefly the main techniques to model an electronic circuit mathematically and to compute the time traces of its signals. It also introduces the basics of MOR and explains the projection based MOR technique of nonlinear ODE-systems, which is equal to the methods discussed in [4, 5]. Section 4 shows, how to use simulation time traces to find the mathematical description of linear and nonlinear circuits. The challenges of curve-fitting the simulation results and required additional information to find a proper mathematical model are discussed. Different curve-fitting methods for the curve-fitting are shown. The techniques introduced in Section 3 and 4 are connected together to the final model generation methodology, which is presented in Section 5. In this Section the requirements of a reduced model are discussed. The usefulness of the model generation approach is demonstrated on three applications in Section 6.

The work is concluded and aspects of future work are given in Section 7.

2 Related Work

In the last two decades many researchers realized the possibilities offered by MOR for dynamic systems and tried to improve and apply these methods in practice [2, 3].

Many approaches for linear circuits work on the approximation of the transfer function. Truncation methods as shown in [7] and [8] cut and simplify the transfer function of linear circuits, ensuring the difference between the original and reduced transfer function to stay in a specified range. Therefore the transfer function is developed in polynomial form, using pade approximation [9] or similar techniques, and cut after the most significant terms.

Proper Orthogonal Decomposition (POD) is a method which replaces a correlation matrix, characterizing the system, by a smaller matrix, holding the the same significant eigenvalues [10]. POD can be interpreted as a mean to compute a Krylov subspace. Other ways to find a smaller subspace are iterative methods like Arnoldi and Lanczos algorithm, which are discussed as MOR techniques, e.g., in [4, 5, 11, 12]. These iterative techniques have the advantage of a low computation effort. Working on the reduction of square matrices, they compute the significant eigenvalues and eigenvectors. This most significant vectors build a projection matrix to transform the model to a subspace.

All of these methods work fine for linear systems [3, 7, 8]. They can, for example, describe large RLC-networks, which occur especially in models of wire interconnection in every electronic network. To apply MOR techniques to nonlinear systems, they have to be modified.

A simple approach, which can be used for circuits with only a few nonlinear elements, is introduced in [13]. Dividing a circuit in linear and nonlinear subcircuits, the mathematical model of the linear part of the circuit is decoupled from the nonlinear part, reduced and again linked to the nonlinear part.

A technique to include the nonlinear part into the reduction is the trajectory piecewise MOR introduced in [4, 5]. This approaches transforms the strong nonlinear system to a piecewise linear system and applies a Krylov algorithm on them. A similar technique, which is introdued in [6], uses piecewise polynomial instead of piecewise linear systems, to replace the nonlinear system. Also for this approach Krylov methods are used, as the reduction works on square matrices of the system state space description. All approaches for the reduction of a nonlinear system need a first simulation of the original dynamic system with a training input. This simulation is used to explore the system state space and find proper points for the development of a piecewise system.

System Modelling using Curve-Fitting

Curve-fitting techniques are used in many different scientific disciplines. Mostly experimental data is used to determine some parameters of well known functions, which describe the underlying physical process, e.g., the type and portion of a radio active element in an object can be found by curve-fitting the function of the decay per time.

Also in the field of electrical engineering, curve-fitting is used. A general approach to describe the I-V-characteristic of semiconductor devices is to measure this curves in a laboratory experiment and use this curves to find typical parameters of the devices. This top-down modelling is very efficient and simple, as it uses the physical behaviour of a device to describe it, instead of calculating its parameters from material and geometrical values [14]. Also SPICE programs themselves offer the possibility to calculate device parameters by curve-fitting a set of data [15]. Nevertheless, in all these approaches curve-fitting is used to describe the behaviour of a single device, using a set of data, which may includes around four or five different physical quantities [14]. In this work we investigate to find the mathematical model of a whole circuit, by curve-fitting a high number of different signals.

3 Preliminaries

3.1 Mathematical Modelling and Numerical Simulation

In many disciplines simulation is accepted as the third pillar of science, beside theory and experiment [16]. Especially in the analysis and design of analog electronic circuits simulation it is a required method. It is the most exact and often the only possible way to analyse the dynamical system, which results from the connection of thousands and more nonlinear devices.

SPICE programs, which are the state of the art for analog circuit computation, have two main functionalities. At first they find a mathematical description of a circuit, which is described by a netlist. Therefore, they use very detailed device models and the technique of Modified Nodal Analysis (MNA) [17]. For the transient analysis, the mathematical model can be a pure algebraic equation system (for a pure resistive circuit), a linear ODE or DAE system (circuit just consisting of linear resistances R, capacitances C and inductances L) or a nonlinear ODE or DAE system (circuit includes active devices or nonlinear R, C or L). The second functionality is to solve the generated equation system numerically. Therefore SPICE uses modified Trapezoidal or Backward Euler Algorithm [17, 18], which are introduced in Section 3.1.3.

In Section 3.1.1 is shown, how the three most common active devices, diode, bipolar junction transistor (BJT) and field-effect transistor (FET), are replaced by their equivalent circuit for a transient analysis. In Section 3.1.2 the theory and a practical example of the MNA are explained. The knowledge of this techniques are the basis for the curve-fitting of the time traces. They are required to select the signals, which are the state space variables of the system, and are especially important to generate a function guess for the curve-fitting of the nonlinear systems, described in Chapter 4.

3.1.1 Equivalent Circuits

To prepare the circuit for the MNA, all active devices have to be replaced by their transient equivalent circuit. This will transform the electronic circuit to a network, which is completely drawn by two-port elements. After the replacement, the circuit will just consist of the elements shown in Figure 1.

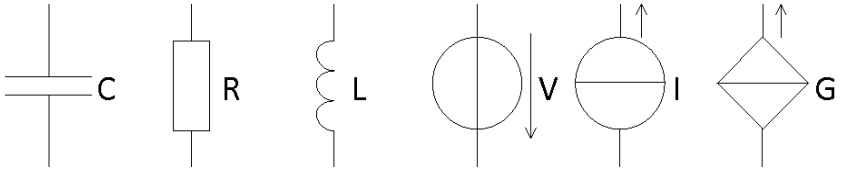


Fig. 1: Two-port Elements for Complete Circuit Modeling at the Device Level

Where the symbols show a capacitance (C), a resistance (R), an inductance (L), an independent voltage source (V), an independent current source (I) and a voltage controlled current source (G). In the following the transient equivalent circuits of the three most common active devices of diode, BJT and FET are shown. The capacitances in the equivalent circuits are junction or diffusion capacitances, depending on the state of the pn-junction. Resistances model the parasitic effect of nonideal conductors.

Diode

The diode semiconductor structure and its transient equivalent circuit are shown in Figure 2, where R_s is the series resistance, C_d the junction capacitance and G_d a VCCS, which follows the I-V-characteristic of the pn-junction shown in Equation (1).

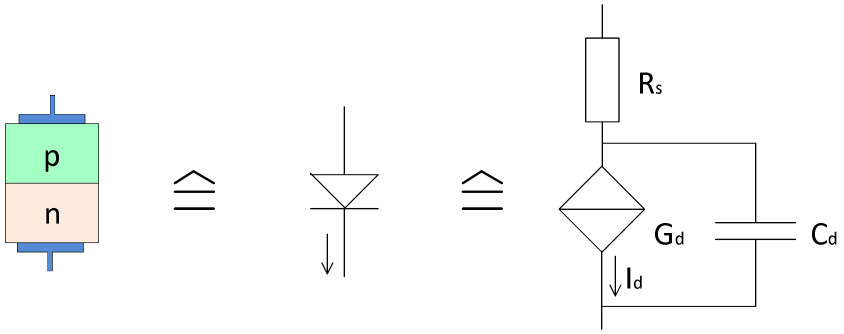


Fig. 2: Diode - Semiconductor Structure, Symbol and Transient Equivalent Circuit

$$I_d = I_s \cdot \left(e^{\frac{V_D}{n \cdot V_T}} \right) \tag{1}$$

Bipolar Junction Transistor

The BJT is implemented by three doped semiconductor zones, which all have a contact terminal. The two pn-junctions behave like diodes and are modelled in the equivalent circuit by VCCSs and capacitances. The transport model of a npn-BJT, which can be used as transient equivalent circuit, is shown in Figure 3 [19]. It is completed by three parasitic resistances to model the ohmic characteristics of the terminals.

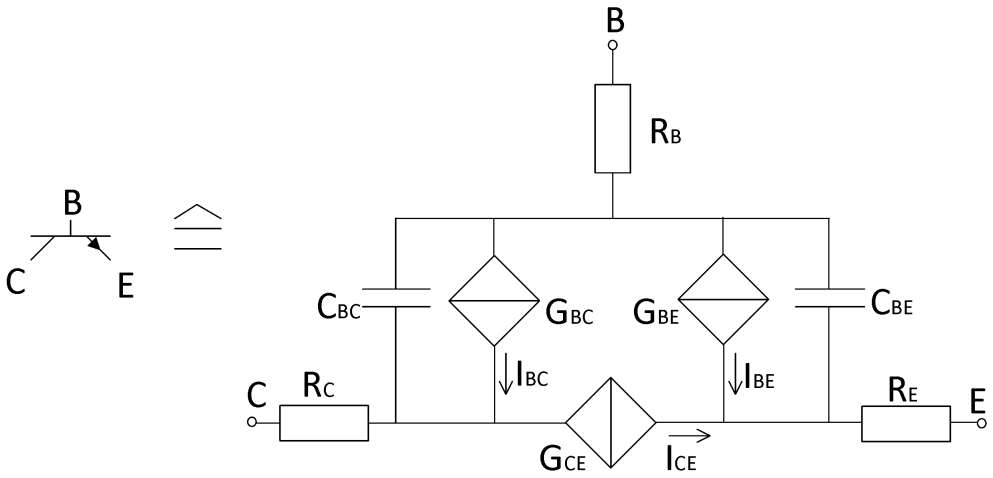


Fig. 3: NPN BJT - Symbol and Transient Equivalent Circuit

The functions of the VCCSs of the equivalent circuit are the shown in the following.

$$
\begin{aligned}
I_{BC} &= \frac{I_s}{B_R} \cdot e^{\frac{V_{BC}}{V_T}} \\
I_{BE} &= \frac{I_s}{B_N} \cdot e^{\frac{V_{BE}}{V_T}} \\
I_{CE} &= I_s \cdot \left(e^{\frac{V_{BE}}{V_T}} - e^{\frac{V_{BC}}{V_T}} \right)
\end{aligned}
\tag{2}
$$

Because of the very small gap between the two pn-junctions, they influence each other and the base-current can control the collector current. The current I_{BC} is close to zero for forward operation. For a pnp-transistor the current direction and the controlling voltages in the VCCS functions are inverted.

Field-Effect Transistor

The FET is, like the BJT, implemented by three semiconductor zones. The controlling node is the gate. Unlike the BJT, the gate is isolated and the current from drain to source is controlled by the gain voltage. As shown in Figure 4 the device has four terminals. The bulk is usually connected to the source node (for n-channel FET) or to drain (for p-channel FET). The transient equivalent circuit is shown in Figure 5. The pn-junctions are again modelled

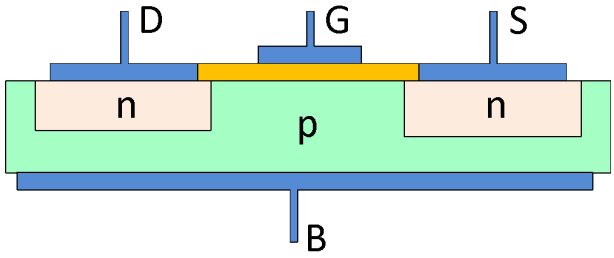


Fig. 4: N-Channel FET - Implementation

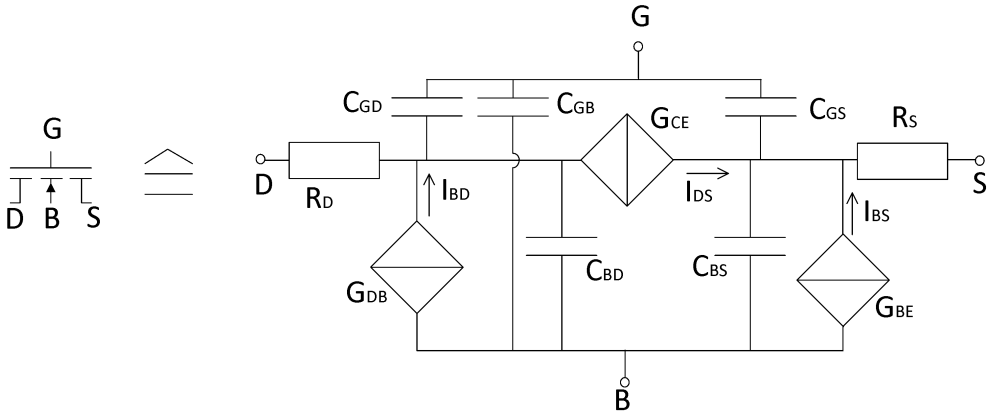


Fig. 5: N-Canal FET - Transient Equivalent Circuit

by a VCCS and a capacitance. Unlike the BJT there is no charge exchange between the controlling gate and the other terminals possible. The currents of the VCCSs for the n-channel FET are determined by the following functions. For the p-channel FET the current

directions and the direction of the controlling voltages in the functions are inverted.

$$
\begin{aligned}
I_{DS} &= \begin{cases} 0 & if\ V_{GS} < V_T \\ \beta \cdot 0.5 \cdot (V_{GS} - V_T) & if\ V_{DS} > V_{GS} - V_T > 0 \\ \beta \cdot V_{DS} \cdot (V_{GS} - V_T - \frac{V_{DS}}{2}) & if\ V_{DS} < V_{GS} - V_T \end{cases} \\
I_{BD} &= I_s \cdot e^{\frac{V_{BD}}{V_T}} \\
I_{BS} &= I_s \cdot e^{\frac{V_{BS}}{V_T}}
\end{aligned}
\tag{3}
$$

where the factor β is a function of geometrical and material parameters of the transistor $\beta = f(w, l, \epsilon_{ox}, d_{ox}, \mu_n)$.

3.1.2 Modified Nodal Analysis

After replacing all active devices by their transient equivalent circuit, the circuit is prepared for MNA, as it is completely drawn by two-port elements. MNA is a method to find the mathematical model of an electronic circuit of any size in a quick, automated manner. The mathematical model is an equation system in matrix-form. The algorithm is based on the Kirchhoff's current law, as it computes the sum of all currents in each node. Different variants of MNA are shown in many sources, e.g., in [17, 20, 21, 22]. At this place a MNA method is explained step by step, which is used to find a frame for the mathematical system, which will be used for the curve-fitting in Chapter 4.

- All branches and nodes of the circuit are labeled, except the ground node, which represents a zero potential reference.
- The resistance, capacitance and inductance matrices R, C, L are developed as diagonal matrices, whose diagonal elements are ordered by the number of the branches.
- Dependent and independent current sources and voltage sources are merged to the row vectors I and V.
- The incidence matrices for all the elements A_R, A_L, A_C, A_I and A_V are built. The rows of the incidence matrices represent the circuit nodes (except ground), the circuit elements correspond to the columns. If a node is the beginning of a branch we will find a '1' in the corresponding matrix element, if it is the end a '-1' and otherwise '0'.

The matrices, which are founded by the circuits structure, are multiplied to get

$$
\begin{aligned}
ACA &= A_C \cdot C \cdot A_C^T \\
AGA &= A_R \cdot R^{-1} \cdot A_R^T.
\end{aligned}
\tag{4}
$$

From this matrices the node voltages, which are imposed voltages, have to be cut. Therefore the rows and columns of the matrix ACA and the rows of the matrices AGA, A_I and A_L, which correspond to the nodes with the imposed voltages, are removed from the matrices to get ACA_{cut}, AGA_{cut}, $A_{I,cut}$ and $A_{L,cut}$. The state variables of the circuit are all voltages e of nodes with a capacitive coupling to the ground and all currents i_L through inductances. The state space model of the electronic circuit is developed as shown in the following equations.

$$
\begin{aligned}
\dot{\vec{e}} &= ACA_{cut}^{-1} \cdot \left(-AGA_{cut} \cdot \begin{bmatrix} V(t) \\ \vec{e} \end{bmatrix} - A_{I,cut} \cdot I(\vec{e}, c) - A_{L,cut} \cdot \vec{i_L} \right) \\
\dot{\vec{i_L}} &= L^{-1} \cdot A_L^T \cdot \begin{bmatrix} V(t) \\ \vec{e} \end{bmatrix}
\end{aligned}
\tag{5}
$$

ODE-System Modelling Using Parasitic Elements

The shown MNA method will result in an ODE or DAE-system, depending on the circuit itself. To ensure the resulting system to be easily manageable with numerical methods, the system should be an ODE-system. A DAE system is not solvable in every case. The circuit can be transformed to a ODE-system, without changing its function or behaviour, by adding very small parasitic elements to it. This is shown in the following example.

Example

At a simple one-transistor amplifier circuit the MNA method is demonstrated. Figure 6 shows its circuit before and after the BJT is replaced by its equivalent circuit.

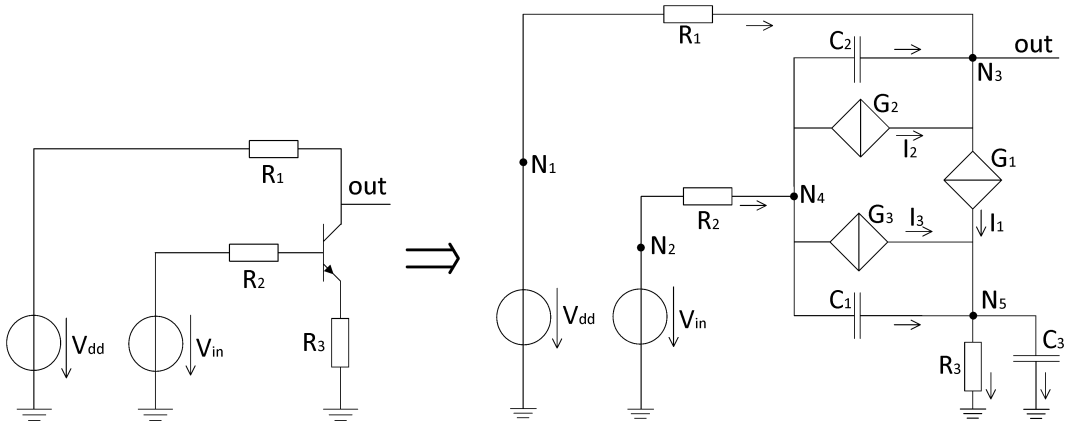


Fig. 6: Simple BJT Amplifier Circuit Before and After Equivalent Circuit Replacement

The branches of the circuit in Figure 6 are labeled with a current direction. Its element and incidence matrices are shown in the following. To illustrate the construction of the incidence matrices A_R, A_I and A_C, the rows and columns are labeled with the corresponding nodes and branches, respectively.

$$R = \begin{bmatrix} R_1 & 0 & 0 \\ 0 & R_2 & 0 \\ 0 & 0 & R_3 \end{bmatrix} \qquad I = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} \qquad V = \begin{bmatrix} V_{vdd} \\ V_{in} \end{bmatrix} \qquad C = \begin{bmatrix} C_1 & 0 & 0 \\ 0 & C_2 & 0 \\ 0 & 0 & C_3 \end{bmatrix}$$

$$A_R = \begin{array}{c} \\ N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \end{array} \begin{array}{c} \begin{array}{ccc} R_1 & R_2 & R_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \qquad A_I = \begin{array}{c} \\ N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \end{array} \begin{array}{c} \begin{array}{ccc} G_1 & G_2 & G_3 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & 1 \\ -1 & 0 & -1 \end{bmatrix} \end{array} \qquad A_C = \begin{array}{c} \\ N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \end{array} \begin{array}{c} \begin{array}{ccc} C_1 & C_2 & C_3 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \\ 1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \end{array}$$

Using these matrices ACA and AGA are calculated using Equation (4):

$$ACA = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_2 & -C_2 & 0 \\ 0 & 0 & -C_2 & C_1 + C_2 & -C_1 \\ 0 & 0 & 0 & -C_1 & C_1 + C_3 \end{bmatrix} \quad AGA = \begin{bmatrix} \frac{1}{R_1} & 0 & \frac{-1}{R_1} & 0 & 0 \\ 0 & \frac{1}{R_2} & 0 & \frac{-1}{R_2} & 0 \\ \frac{-1}{R_1} & 0 & \frac{1}{R_1} & 0 & 0 \\ 0 & \frac{-1}{R_2} & 0 & \frac{1}{R_2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{R_3} \end{bmatrix}$$

The voltages at the first two nodes N_1 and N_2 are imposed by the supply voltage and the input signal, respectively. Therefore the first two rows and columns of the matrix ACA and the first two rows of the matrices AGA and A_I are cut. The final system is equal to Equation (5) and shown in the following.

$$\dot{\vec{e}} = ACA_{cut}^{-1} \cdot \left(-AGA_{cut} \cdot \begin{bmatrix} V(t) \\ \vec{e} \end{bmatrix} - A_{I,cut} \cdot I(\vec{e}) \right) \tag{6}$$

$$\begin{bmatrix} \dot{e_3} \\ \dot{e_4} \\ \dot{e_5} \end{bmatrix} = \begin{bmatrix} C_2 & -C_2 & 0 \\ -C_2 & C_1 + C_2 & -C_1 \\ 0 & -C_1 & C_1 + C_3 \end{bmatrix}^{-1} \cdot \left(- \begin{bmatrix} \frac{-1}{R_1} & 0 & \frac{1}{R_1} & 0 & 0 \\ 0 & \frac{-1}{R_2} & 0 & \frac{1}{R_2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{R_3} \end{bmatrix} \cdot \begin{bmatrix} V_{vdd} \\ V_{in} \\ e_3 \\ e_4 \\ e_5 \end{bmatrix} - \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & 1 \\ -1 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} \right)$$

where $\vec{e}$ is the vector of the voltages of the internal nodes N_{3-5} of the circuit in Figure 6. The terminal voltages $V(t)$ are a function of time and the three currents I of the VCCS are functions of $\vec{e}$.

In this example the matrix ACA is nonsingular. Due to this the equation system consists completely of ODEs. If the capacity C_3 would be missing in the circuit, the matrix ACA_{cut} would be singular and the resulting equation system would be a DAE system of the following structure.

$$ACA_{cut} \cdot \dot{\vec{e}} = -AGA_{cut} \cdot \begin{bmatrix} V(t) \\ \vec{e} \end{bmatrix} - A_{I,cut} \cdot I(\vec{e}, c) \tag{7}$$

The value of the capacity C_3, which is inserted to the circuit, to make it mathematically manageable for an ODE-solver, has to be small in comparison to the other capacities, to not influence the simulation result. This concept of adding small capacitances is also used by professional SPICE programs [19, 23].

3.1.3 Numerical Simulation

After computing the mathematical description of the circuit, the transient solution is calculated by numerical integration. In the scope of this work, we assume the mathematical model of the circuit to be an ODE-description, as described above. Here, the two most common numerical techniques for circuit simulation, the trapezoidal and the gear algorithm, are explained. They are the mean to get the transient time traces, which are the basis for the curve-fitting. Both of them are used to solve problems of the following form, which is a general expression of Equation (5) or (6).

$$\dot{\vec{x}} = f(t, \vec{x}) \qquad \vec{x}(t = 0) = \vec{x}_0 \tag{8}$$

where $\vec{x}_0 \in \mathbb{R}^n$ is an initial condition and $f : \mathbb{R}^{n+1} \Rightarrow \mathbb{R}^n$ a nonlinear function.

Gear Algorithm

The Gear algorithm of order one is known as implicit or backward Euler method. It replaces the exact integral formula by a linear approximation.

$$\vec{x}_{k+1} = \vec{x}_k + \int_{t_k}^{t_{k+1}} f(t, \vec{x}(t)) \cdot dt \approx \vec{x}_k + h \cdot f(t_{k+1}, \vec{x}_{k+1}) \tag{9}$$

where k is the index of the current time step and h is the time step size. As the solution at the next time step $\vec{x}_{k+1}$ is implicit, it has to be calculated iteratively. Therefore an initial guess for the solution at the next time step $\vec{x}_{k+1}^{\,0}$ is used. The formula

$$\vec{x}_{\,k+1}^{\,o+1} = \vec{x}_k + h \cdot f(t_{k+1}, \vec{x}_{k+1}^{\,o}), \tag{10}$$

where o is the index of the current number of iteration, is calculated multiple times, until the change

$$\triangle \vec{x}_{k+1} = \vec{x}_{\,k+1}^{\,o+1} - \vec{x}_{k+1}^{\,o} \tag{11}$$

is smaller than a specified bound. Then the procedure repeats for the next time step k. After the calculation of the second and third time step, some solvers use higher order formulas of the Gear algorithm. In difference to the backward Euler method, shown in Equation (9) and (10), this algorithms considers also the states of former time points. These techniques are explained in detail in [17, 18].

Trapezoidal Algorithm

The Trapezoidal algorithm is the default algorithm used in SPICE [18, 19]. It is also an implicit algorithm. Instead of assuming a rectangle to approximate the integral, it uses a trapezoid, as shown in the following formula.

$$\vec{x}_{k+1} = \vec{x}_k + \int_{t_k}^{t_{k+1}} f(t, \vec{x}(t)) \cdot dt \approx \vec{x}_k + \frac{h}{2} \cdot (f(t_k, \vec{x}_k) + f(t_{k+1}, \vec{x}_{k+1})) = \vec{x}_k + \frac{h}{2} \cdot (\dot{\vec{x}}_k + \dot{\vec{x}}_{k+1}) \tag{12}$$

To get the solution for $\vec{x}_{k+1}$, also for the Trapezoidal algorithm an iterative method is used. The formula to repeat, to find the solution at the next time step, is shown in the following.

$$\vec{x}_{\,k+1}^{\,o+1} = \vec{x}_k + \frac{h}{2} \cdot (f(t_k, \vec{x}_k) + f(t_{k+1}, \vec{x}_{k+1}^{\,o})) \tag{13}$$

where o and k are the indices of the current iteration and the current time step, respectively. The initial guess in each time step is usually calculated by the forward Euler method.

$$\vec{x}_{\,k+1}^{\,0} = \vec{x}_k + h \cdot f(t_k, \vec{x}_k) \tag{14}$$

This combination of forward Euler and Trapezoidal method is known as Heun's method. Advanced numerical ODE-solver, as they are used in SPICE or MATLAB, work with a variable time step h. The control of the time step is explained in detail in [17, 18].

3.2 Model Order Reduction

This section briefly explains Krylov MOR techniques. First, the MOR for linear systems is explained. Then, an approach for the MOR for nonlinear systems, which combines singular value decomposition (SVD) and a Krylov algorithm to elaborate a projection matrix, is shown. This is the approach, which also will be used in the presented model generation methodology in Chapter 5 and for the applications in Chapter 6.

3.2.1 Subspace Methods for Linear Systems

For linear systems, like RC, RL or RLC-circuits, the system description in form of the state space is shown in the following.

$$\begin{aligned} \dot{x} &= A \cdot x + B \cdot u \\ y &= C \cdot x \end{aligned} \tag{15}$$

where $y \in \mathbb{R}^o$, $x \in \mathbb{R}^n$, $u \in \mathbb{R}^p$, $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times p}$, and $C \in \mathbb{R}^{o \times n}$. MOR will replace this dynamical system by a system with less state variables, as shown in the following.

$$\begin{aligned} \dot{\hat{x}} &= \hat{A} \cdot \hat{x} + \hat{B} \cdot u \\ y &= \hat{C} * \hat{x} \end{aligned} \tag{16}$$

where $y \in \mathbb{R}^o$, $\hat{x} \in \mathbb{R}^q$, $u \in \mathbb{R}^p$, $\hat{A} \in \mathbb{R}^{q \times q}$, $\hat{B} \in \mathbb{R}^{q \times p}$, $\hat{C} \in \mathbb{R}^{o \times q}$ and $q < n$.

Also the reduced system is smaller, it must keep the properties of the original system. The input-output behaviour of both systems must be similar, which means that the transfer function of both systems are alike. The system is reduced using its state space matrices. The transfer function can be developed from the state space model, by using Laplace transformation, as shown in the following.

$$\begin{aligned} &1. && \dot{x} = A \cdot x + B \cdot u \\ &2. && y = C \cdot x \\ &Laplace\ Transformation \downarrow \\ &3. && s \cdot X(s) = A \cdot X(s) + B \cdot U(s) \\ &4. && Y(s) = C \cdot X(s) \\ &from\ 3. && s \cdot X(s) - A \cdot X(s) = B \cdot U(s) \\ & && (s \cdot I - A) \cdot X(s) = B \cdot U(s) \\ &5. && X(s) = (s \cdot I - A)^{-1} \cdot (B \cdot U(s)) \\ &from\ 4.\ and\ 5. && Y(s) = C \cdot (s \cdot I - A)^{-1} \cdot (B \cdot U(s)) \\ & && G(s) = Y(s) \cdot U(s)^{-1} = C \cdot (s \cdot I - A)^{-1} \cdot B \end{aligned} \tag{17}$$

where I is the identity matrix. If the system is a single input single output (SISO) system, the transfer function can be transformed to a fraction of two polynomials using Pade approximation [9].

$$G(s) = \frac{Y(s)}{U(s)} = \frac{a_0 + a_1 \cdot s + a_2 \cdot s^2 + ... + a_n \cdot s^n}{1 + b_1 \cdot s + b_2 \cdot s^2 + ... + b_m \cdot s^m} \tag{18}$$

Around an expansion point $s_0 \in \mathbb{C}$ this polynomials can be replaced by an infinite power series [3] of its moments m, as shown in Equation (19). To find a reduced description of the transfer function, its most significant moments have to be computed. The transfer function of the reduced system $\hat{G}(s)$ just consists of the q most significant moments of the original transfer function $G(s)$.

$$\begin{aligned} G(s) &= m_0 + m_1 \cdot (s - s_0) + m_2 \cdot (s - s_0)^2 + m_3 \cdot (s - s_0)^3 + ... \\ \hat{G}(s) &= m_0 + m_1 \cdot (s - s_0) + m_2 \cdot (s - s_0)^2 + ... + m_q \cdot (s - s_0)^q \end{aligned} \tag{19}$$

One method, to find the moments, is the Arnoldi algorithm. It will compute a matrix $V \in \mathbb{R}^{n \times q}$, where q is the number of the moments, and $V \cdot V^T = I$. The matrix V consists of the moment vectors:

$$V(B, \hat{A}) = span\{B, \hat{A} \cdot B, ..., \hat{A}^{q-1} \cdot B\} = [V_1, V_2, ..., V_q].$$

The Arnoldi algorithm, shown in the following, is an iterative algorithm. It computes the eigenvalues and eigenvectors of a matrix, to get an orthogonal basis for a Krylov subspace.

Algorithm Arnoldi: $[A,\ B,\ q] \rightarrow V$

$V_1 = \frac{b}{||b||}$
for $i = 1$ to q **do**
  $v = \langle A, V_i \rangle$
  for $j = 1$ to q **do**
    $H_{j,i} = \langle V, v \rangle$
    $v = v - H_{j,i} \cdot V_j$
  end for
  $H_{i+1,i} = ||v||$
  $V_{i+1} = \frac{v}{H_{i+1,i}}$
end for

The matrix of interest is the system matrix A. The start vector for the iteration is the vector B. q is the number of iterations, which is equal to the number of the columns of the matrix V, which the algorithm will compute. It is proven, e.g., in [24], that the moment matching using Arnoldi algorithm, will keep the q most significant moments of the transfer function, as long as it does not break down. This would be the case, if q reaches the rank of matrix A. That means, that the transfer function of the reduced model, which is calculated as shown in the following, has the same first q moments as the original system. To compute the reduced model in the subspace, the matrix V is used, which is the output of the Arnoldi algorithm.

$$\hat{A} = V^T \cdot A\ , \qquad \hat{B} = V^T \cdot B\ , \qquad \hat{C} = C \cdot V$$

Another well known algorithm to find a Krylov subspace, by computing the eigenvalues and eigenvectors of the matrix A, is the Lanczos algorithm [3]. An approach to make the Arnoldi algorithm working for multi input multi output (MIMO) systems is the global Arnoldi algorithm, which is shown as an extension of the standard Arnoldi algorithm in [25].

3.2.2 Nonlinear Systems

Also, there are a lot of parts in electronic circuits, which can be modelled by a linear system, the most circuits are nonlinear. For those systems a reduction is not as straightforward as for linear systems. A method to find a projection basis for a nonlinear state space model, using Arnoldi algorithm and SVD, is presented in the following. It is similar to the trajectory piecewise MOR, which is presented in [4, 5, 26, 27]. It is shown, how the reduced nonlinear system is simulated, using piecewise linear models.
A nonlinear model has the general description

$$\begin{aligned} \dot{x} &= f(x, u(t)) \\ y &= g(x), \end{aligned} \tag{20}$$

where $x \in \mathbb{R}^n$ are the state variables of the system, $u \in \mathbb{R}^p$ and $y \in \mathbb{R}^o$ the input and output vector of the system and $f : \mathbb{R}^{n+p} \rightarrow \mathbb{R}^n$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}^o$ are nonlinear functions.
This model is transformed to a piecewise model of k linear systems, using the simulation traces of a former simulation of the system. For the linearisation of the system around the k linearisation points x_L, the Jacobian matrices J_x and J_u of f at these points are used.

J_x contains all first derivatives of f regarding to the state variables, while J_u contains the derivatives to the input signals. A method to find the linearization points is shown in detail in Chapter 5.

The function f of Equation (20) is replaced by the following linear expression, which is equal to the Taylor polynomial of order 1 of f.

$$\begin{aligned} \dot{x} &= x_{L,i} + J_{x,i} \cdot (x - x_{L,i}) + J_{u,i} \cdot u \\ x_{L,i} &= \min_{x_{L,i}}(||x_{L,1} - x||, ||x_{L,2} - x||, ..., ||x_{L,k} - x||) \end{aligned} \tag{21}$$

where $J_x \in \mathbb{R}^{n \times n}$, $J_u \in \mathbb{R}^{n \times p}$ and i is the index to identify the corresponding linear system, which is the system, whose development point $x_{L,i}$ is the closest to the current state x. To find a projection matrix, the Arnoldi algorithm is applied to all system matrices A_i to find the k eigenvector-matrices V_i. Again, each matrix V_i contains the q most significant eigenvectors of the corresponding matrix A. The final projection matrix U is a product of the SVD of the combination of all the matrices V_i. The SVD is already implemented in MATLAB. The other two matrices resulting from the SVD are not used.

$$(U, P, O) = SVD\ (V_1 \cup V_2 \cup ... \cup V_k)$$

Matrix U is used as projection matrix. It transforms the model, shown in Equation (21), to the reduced state space model shown in the following.

$$\begin{aligned} \dot{z} &= z_i + A_i \cdot (z - z_i) + B_i \cdot u \\ z_i &= \min_{z_i}(||z_1 - z||, ||z_2 - z||, ..., ||z_k - z||) \end{aligned} \tag{22}$$

where

$$\begin{aligned} A &= U^T \cdot J_x \cdot U \\ B &= U^T \cdot J_u \\ z_i &= U^T \cdot x_{L,i} \end{aligned} \tag{23}$$

For a numerical simulation of the dynamical system, the initial condition is projected to the reduced state space. Then, at each time step t_k of the numerical solution of the reduced system, the current linear system is selected by searching for the closest linearization point z_i to the current state $z(t)$. Once the simulation is finished, the time traces of the reduced state space variables are transformed back to the original state space and provide an approximation of the solution in the original state space.

$$\hat{x}(t) = U \cdot z(t) \tag{24}$$

4 Curve-Fitting of Analog Circuit Simulation Traces

The focus of this chapter is to identify successful methods for the curve-fitting of analog circuits. The function system, which is the desired output of the curve-fitting procedure is an ODE-system. It can be used instead of the original system, to describe the behaviour of the analog circuit. At this point it is necessary to differentiate between linear and nonlinear systems, as the effort for the curve-fitting and the additionally required information are quite different. In case of a linear system the procedure shown in Section 4.1 can be used to find the system description, if the parameters of the original circuit are unknown. For nonlinear systems, the procedure shown in Section 4.2 is helpful if the active device models, which

are used by the original simulation, are to complicated and should be replaced by simpler models.

For both linear and nonlinear systems, the curves, which are under the focus of the curve-fitting, are the time derivatives of the circuits state variables. An approximation of these time derivatives is calculated using the time traces themselves.

$$\dot{\vec{x}} \approx \left(\frac{\triangle \vec{x}}{\triangle t} \right)_{sampl} \tag{25}$$

As the approximation of the time derivatives is the average gradient of the signal between two sample points, the corresponding signal value is computed as the middle between the two samples. By this procedure, all sample points are shifted by half of a time step. This is done for all state variable time traces.

$$x_k = \frac{x_k + x_{k+1}}{2}; \tag{26}$$

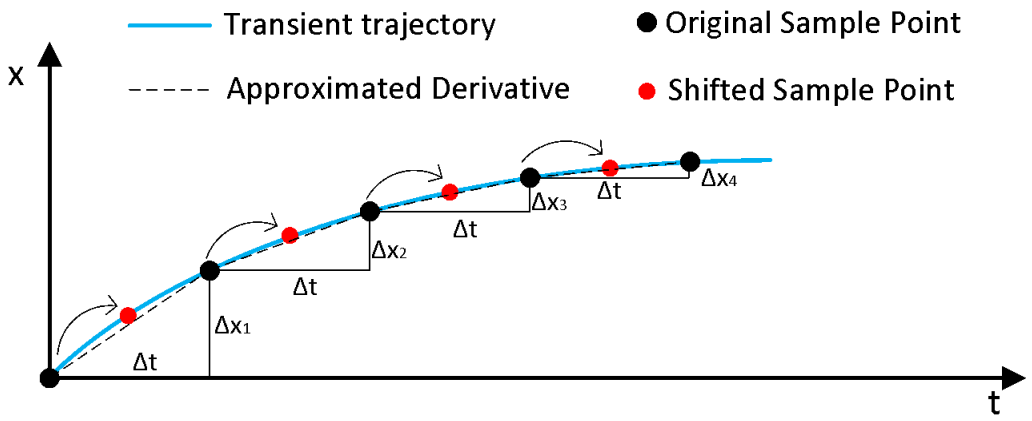


Fig. 7: Sample Points Shift and Derivative Calculation

Holding the time trace of the circuits state variables $\vec{x}(t)$ and the time traces of the input signals $\vec{u}(t)$ the objective of the curve-fitting procedure is to find the function $\hat{g}$, which is an approximation of g.

$$\dot{\vec{x}} = g(\vec{u}(t), \vec{x}) \tag{27}$$

One difficulty of the curve-fitting is the amount of data. The derivative of each state variable can depend on many other state variables, according to the structure of the circuit. Each time trace of a signal can include up to 10^6 sample points. The methods explained in the following sections are able to deal with this huge amount of data.

4.1 Linear Circuits

A linear electronic circuit just consists of linear resistances, capacitances and inductances. In case of a linear circuit, Equation (27) can be transformed to a linear expression of the following form.

$$\dot{\vec{x}} = A \cdot \vec{x} + B \cdot \vec{u}(t) \tag{28}$$

where $\vec{x} \in \mathbb{R}^n$ is the vector of the state variables of the circuit, $\vec{u}(t) \in \mathbb{R}^p$ is the vector of the input signals of the circuit and $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times p}$ are constant matrices. The state variables of the circuit and so the signals, which have to be taken into account, can be

identified using the MNA method, shown in Section 3.1.2. To prepare the curve-fitting for the identification of the matrices A and B, Equation (28) is transcribed to:

$$\dot{\vec{x}} = \begin{bmatrix} B & A \end{bmatrix} \cdot \begin{bmatrix} \vec{u}(t) \\ \vec{x} \end{bmatrix} = J \cdot \begin{bmatrix} \vec{u}(t) \\ \vec{x} \end{bmatrix} \tag{29}$$

where $J \in \mathbb{R}^{n \times (n+p)}$ is the Jacobian matrix of the system. This matrix contains all first derivatives of the state variables to each other and to the input signals.

$$J = \begin{bmatrix} \frac{\partial x_1}{\partial u_1} & \frac{\partial x_1}{\partial u_2} & \cdots & \frac{\partial x_1}{\partial u_p} & \frac{\partial x_1}{\partial x_1} & \frac{\partial x_1}{\partial x_2} & \cdots & \frac{\partial x_1}{\partial x_n} \\ \frac{\partial x_2}{\partial u_1} & \frac{\partial x_2}{\partial u_2} & \cdots & \frac{\partial x_2}{\partial u_p} & \frac{\partial x_2}{\partial x_1} & \frac{\partial x_2}{\partial x_2} & \cdots & \frac{\partial x_2}{\partial x_n} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \frac{\partial x_n}{\partial u_1} & \frac{\partial x_n}{\partial u_2} & \cdots & \frac{\partial x_n}{\partial u_p} & \frac{\partial x_n}{\partial x_1} & \frac{\partial x_n}{\partial x_2} & \cdots & \frac{\partial x_n}{\partial x_n} \end{bmatrix} \tag{30}$$

For a linear system, the Jacobian is constant and independent from the state variables. It is a complete description of the dynamical system. To calculate all elements of J with the mean of the time simulation traces, n linear systems are developed, which each corresponds to one line of the Jacobian matrix and one derivative of a state variable. Using this approach, the elements of J are calculated row by row. The equation system for the nth state variable consists of equations of the following form.

$$\dot{x}_{n,k} = J_{n,1} \cdot u_{1,k} + ... + J_{n,p} \cdot u_{p,k} + J_{n,p+1} \cdot x_{1,k} + ... + J_{n,p+n} \cdot x_{n,k} \tag{31}$$

where k is the number of sample points. Every sample point will add an equation to the linear system. As the number of sample points of a numerical simulation is normally much larger than the number of state variables, the equation system is overdetermined. Here an overdetermined system is necessary for a proper solution, as all equations are just an approximation (Equation 25).

Developing equations (31) for all sample points, the complete system is found.

$$b = A \cdot c \tag{32}$$

$$\text{where} \quad b = \begin{bmatrix} \dot{x}_{n,1} \\ \dot{x}_{n,2} \\ \vdots \\ \dot{x}_{n,k} \end{bmatrix}, \quad A = \begin{bmatrix} \vec{u}_1^T & \vec{x}_1^T \\ \vec{u}_2^T & \vec{x}_2^T \\ \vdots & \vdots \\ \vec{u}_k^T & \vec{x}_k^T \end{bmatrix} \quad \text{and} \quad c = \begin{bmatrix} J_{n,1} \\ J_{n,2} \\ \vdots \\ J_{n,p+n} \end{bmatrix}.$$

The best solution of this overdetermined system is expressed as a least squares problem. The values c_{opt} which minimize the square of residuals between the samples and the linear function for all time steps, must be calculated.

$$c_{opt} = \min_c \sum_{a=1}^{k} (res_a)^2 = \min_c (A \cdot c - b)^2 \tag{33}$$

Two techniques to solve (33) are shown in the following.

4.1.1 Normal Equations

To find the minimum of the least squares, the scalar product is expanded

$$\begin{aligned} (A \cdot c - b)^2 &= (A \cdot c - b)^T \cdot (A \cdot c - b) \\ &= c^T \cdot A^T \cdot A \cdot c - 2 \cdot c^T \cdot A^T \cdot b - b^T \cdot b \end{aligned} \tag{34}$$

and its derivative regarding c is set to zero, to calculate the values of c, which minimize the sum of the squares of the residuals.

$$\begin{aligned} 0 &= 2 \cdot A^T \cdot A \cdot c - 2 \cdot A^T \cdot b \\ c &= (A^T \cdot A)^{-1} \cdot A^T \cdot b \end{aligned} \tag{35}$$

Using the original symbols again, for each line n of the matrix J the following equation will calculate its parameters, by using the k sample points of a former transient simulation.

$$\begin{bmatrix} J_{n,1} \\ J_{n,2} \\ \vdots \\ J_{n,p+n} \end{bmatrix} = \left(\begin{bmatrix} \vec{u}_1 & \vec{u}_2 & ... & \vec{u}_k \\ \vec{x}_1 & \vec{x}_2 & ... & \vec{x}_k \end{bmatrix} \cdot \begin{bmatrix} \vec{u}_1^T & \vec{x}_1^T \\ \vec{u}_2^T & \vec{x}_2^T \\ \vdots & \vdots \\ \vec{u}_k^T & \vec{x}_k^T \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} \vec{u}_1 & \vec{u}_2 & ... & \vec{u}_k \\ \vec{x}_1 & \vec{x}_2 & ... & \vec{x}_k \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_{n,1} \\ \dot{x}_{n,2} \\ \vdots \\ \dot{x}_{n,k} \end{bmatrix} \tag{36}$$

4.1.2 QR Decomposition

Another technique, to solve Equation (33), is to use the QR decomposition of the matrix A. QR decomposition will replace the matrix A by an orthogonal matrix Q and a square, upper triangular matrix R.

$$\begin{aligned} A &= Q \cdot R \\ Q^T \cdot Q &= I \end{aligned} \tag{37}$$

where I is the identity matrix and $A \in \mathbb{R}^{k \times (n+p)}$, $Q \in \mathbb{R}^{k \times (n+p)}$ and $R \in \mathbb{R}^{(n+p) \times (n+p)}$. Using this decomposition we can rewrite Equation (32) to find c.

$$\begin{aligned} b &= Q \cdot R \cdot c \\ Q^T \cdot b &= R \cdot c \\ c &= R^{-1} \cdot Q^T \cdot b \end{aligned} \tag{38}$$

The QR decomposition algorithm is available as a function in MATLAB with the following command.

```
[Q,R]=qr(A);
```

4.1.3 Example

The two different ways to calculate the Jacobian matrix, namely the use of normal equations or QR decomposition, are applied to a linear example circuit. This circuit shows a small network, built by a transmission line and three loads in an IC. The three loads have ohmic-capacitive characteristic and the power consumption of each is $500mW$.

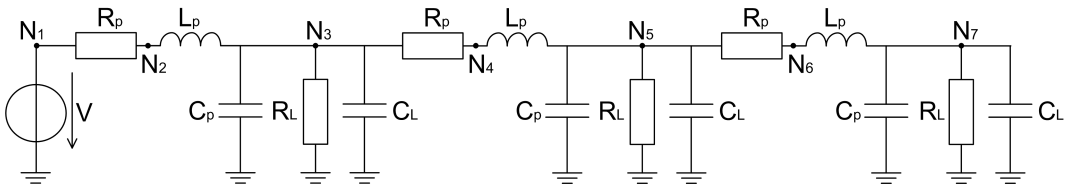


Fig. 8: Unideal IC Transmission Line

The values of the parasitics elements are chosen according to [28]. The circuit parameters are the following.

$$R_P = 1\Omega\ , \quad L_P = 25fH\ , \quad C_P = 1pF\ , \quad R_L = 50\Omega\ , \quad C_L = 0.5nF$$

The state variables vector of this circuit is built by the node voltages with capacitive coupling and the currents through inductances. The only input signal is the supply voltage.

$$\vec{x}^T = \begin{bmatrix} e_3 & e_5 & e_7 & i_{L1} & i_{L2} & i_{L3} \end{bmatrix}\ , \quad u(t) = V(t)$$

Using the MNA approach provided in Section 3.1.2 the circuit equation system is shown by the following linear system.

$$\dot{\vec{x}} = \begin{bmatrix} 0 & \frac{-1}{R_L \cdot (C_P + C_L)} & 0 & 0 & \frac{1}{C_P + C_L} & \frac{-1}{C_P + C_L} & 0 \\ 0 & 0 & \frac{-1}{R_L \cdot (C_P + C_L)} & 0 & 0 & \frac{1}{C_P + C_L} & \frac{-1}{C_P + C_L} \\ 0 & 0 & 0 & \frac{-1}{R_L \cdot (C_P + C_L)} & 0 & 0 & \frac{1}{C_P + C_L} \\ \frac{1}{L_p} & \frac{-1}{L_P} & 0 & 0 & \frac{-R_p}{L_P} & 0 & 0 \\ 0 & \frac{1}{L_p} & \frac{-1}{L_P} & 0 & 0 & \frac{-R_p}{L_P} & 0 \\ 0 & 0 & \frac{1}{L_p} & \frac{-1}{L_P} & 0 & 0 & \frac{-R_p}{L_P} \end{bmatrix} \cdot \begin{bmatrix} \vec{u}(t) \\ \vec{x} \end{bmatrix}$$

The input signal is a DC voltage. The system is simulated in hSPICE. The resulting time traces are used to find the Jacobian matrix of the system.

For the curve-fitting, just the first samples of the signals can be used. After 10 ns the signals do not change anymore, but remain constant, due to the DC input voltage. The time derivative of the state variables becomes zero and the equation system, which would result from the later samples, would be linear dependent and useless for the curve-fitting. The following figure shows the time traces of the state variables. The interesting part of the curves is highlighted.

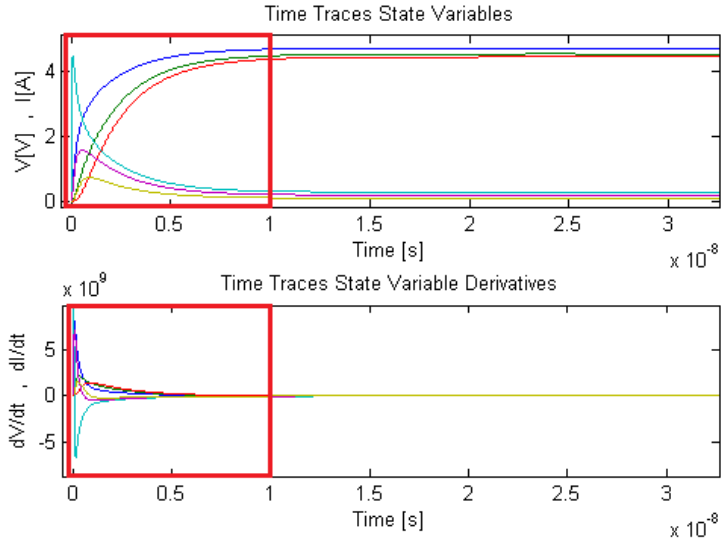


Fig. 9: State Variables Transient Curves

The quality of the matrices, which are calculated by curve-fitting, depends on the time step of the curves and on the method, which is used for the calculation. The following table shows the average error of all matrix elements to the average value of the matrix elements

and the maximum error of all elements.

$$Error_{average} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n+p}\left(\frac{\triangle J_{i,j}}{J_{mean}}\right)}{size(J)} \tag{39}$$

$$Error_{max} = max\left(\frac{\triangle J_{i,j}}{J_{mean}}\right) \tag{40}$$

where $size(J)$ is the total number of elements in the matrix.

Table 1: Jacobian Matrix Quality for Different Methods and Traces

Method	Normal Equation				QR Decomposition			
Traces Time Step Size $[s]$	1e-14	1e-13	1e-12	1e-11	1e-14	1e-13	1e-12	1e-11
$Error_{average}$	1.1e-4	1.7e-3	2.1e-3	0.15	4.3e-8	1.6e-3	2.1e-2	0.15
$Error_{max}$	1.0e-3	2.0e-2	0.25	1.4	1.8e-7	2.0e-2	0.25	1.4

The table shows the strong dependency of the quality of the computed Jacobian to the time step size of the used simulation traces. If the simulation, which is used for the calculation, has a small step size, the resulting Jacobian is very close to the analytical derived matrix. If the time step is bigger, both algorithms fail to compute the zero elements of the matrices properly, what results in an unacceptable $Error_{max}$ of 140% and $Error_{average}$ of 15%. But, in contrast to the big failure in the matrices, the simulation can be redone using the computed Jacobian and the same initial condition and input signal, without any visible difference. This is shown in Figure 11. The system used for the figure is the one with the biggest error (calculated with normal equations and time traces step size 1e-11s).

Based on the presented example, we can summarize that the modelling of a linear

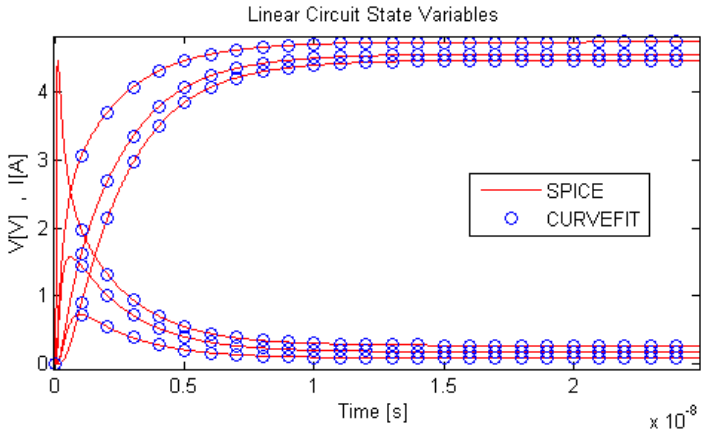


Fig. 10: Comparison - Original Simulation Traces and Curve Fitted System

system with unknown parameters is possible. Depending on the time step size of the SPICE simulation traces, the quality of the approximation model differs. A necessary condition for the curve-fitting is, that the traces are not constant, to avoid a linear dependent system, which is not solvable. They have to show, e.g., the activation operation of the circuit.

4.2 Nonlinear Circuits

After presenting, how a linear system model can be derived, by the use of the transient time traces, the focus of this section is on nonlinear circuits. In the following, two techniques are shown, which approximate the mathematical description of a nonlinear circuit by curve-fitting. The first technique, the Vandermode Interpolation Method, tries to fit a curve with a polynomial function, while the second method, the LevenbergMarquardt algorithm, is a least-squares algorithm, which identifies a set of parameters to minimize the failure between a guess function and a curve.

4.2.1 Vandermode Interpolation Method

The Vandermode Interpolation method used here is described in [29]. It can approximate curves of signals, which depend on more than one other signal, what is required in the case of circuits.

To prepare the curve-fitting procedure, the circuit structure, given in the netlist, has to be used, to identify, which state variables of the circuit influence each other. Therefore, an identification matrix is elaborated, using a procedure similar to the MNA method described in Section 3.1.2. The following figure and equation show an example for an identification matrix.

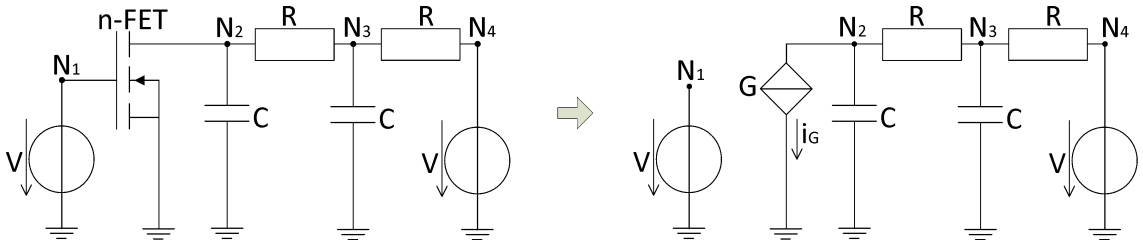


Fig. 11: Example Circuit for Identification Matrix

Assuming that the n-FET is ideal (no parasitic resistances or capacitances) in the circuit in the left, it is replaced by a VCCS between drain and source. Knowing that the FET-current in the circuit is a function of drain, source and gain voltage

$$I_G = f(e_D, e_S, e_G) = f(e_{N1}, 0, e_{N2}),$$

the identification matrix of the circuits state variables is the following.

$$J_I = \begin{array}{c} \\ \dot{e}_{N2} \\ \dot{e}_{N3} \end{array} \begin{array}{c} \begin{array}{cccc} e_{N1} & e_{N2} & e_{N3} & e_{N4} \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Vandermodes method assumes each function to be a polynomial of its arguments, to fit the function values. The original Vandermodes method would use a polynomial of degree $k-1$ for k interpolation points [30]. As the number of points of the transient time traces is too high, just the terms until the second order are taken into account. E.g., if a function value y depends on two arguments x_1 and x_2, the polynomial containing all combination

until the second order, would look like the following.

$$y = c_1 + c_2 \cdot x_1 + c_3 \cdot x_2 + c_4 \cdot x_1^2 + c_5 \cdot x_2^2 + c_6 \cdot x_1 \cdot x_2 + c_7 \cdot x_1^2 \cdot x_2 + c_8 \cdot x_1 \cdot x_2^2 \quad (41)$$

At this point the linear equation system is developed using the k sample points of the transient time traces. The function value y would be the derivative of the state variables, using Equation (25).

$$\begin{aligned}
y_1 = &\quad c_1 + c_2 \cdot x_{1,1} + c_3 \cdot x_{2,1} + c_4 \cdot x_{1,1}^2 + c_5 \cdot x_{2,1}^2 + ... + c_8 \cdot x_{1,1} \cdot x_{2,1}^2 \\
y_2 = &\quad c_1 + c_2 \cdot x_{1,2} + c_3 \cdot x_{2,2} + c_4 \cdot x_{1,2}^2 + c_5 \cdot x_{2,2}^2 + ... + c_8 \cdot x_{1,2} \cdot x_{2,2}^2 \\
&\vdots \\
y_k = &\quad c_1 + c_2 \cdot x_{1,k} + c_3 \cdot x_{2,k} + c_4 \cdot x_{1,k}^2 + c_5 \cdot x_{2,k}^2 + ... + c_8 \cdot x_{1,k} \cdot x_{2,k}^2
\end{aligned}$$

The linear coefficients c in this polynomial are determined by solving the linear equation system, e.g., using the methods explained in the previous section.

This technique has two main drawbacks and is therefore not practical for the curve-fitting of nonlinear circuits. The first reason is that the derivative of all state variables usually depend on more than three other signals. The linear systems derived by the Vandermodes method will become too large and contain too many coefficients c, even, if just the second order polynomial of the arguments is used. The second drawback is the violation of Kirchhoff's current law, which results, if this method is applied to complete circuits. If the method is used as described, the sum of currents in every node would not be zero.

4.2.2 LevenbergMarquardt Algorithm

The Levenberg-Marquardt algorithm is the state of the art for nonlinear curve-fitting tasks [31, 32]. It is implemented in many standard tools like MATLAB, GNU Octave or Mathematica. The algorithm is used to identify a set of parameters $c = (c_1, c_2, ..., c_n)$ of a nonlinear function $y = f(x_1, x_2, ..., x_m, c_1, c_2, ..., c_n)$ for a given set of points $(y_i, x_{1,i}, x_{2,i}, ..., x_{m,i})$, where $i = 1, 2, ..., k$ [32, 33]. In this section, the algorithm itself is explained and then it is shown, how to get the function guess f, by using the nonlinear electronic circuits netlist.

The problem is to minimize the sum of squares of the residuals between the nonlinear function $f(x_i, c)$ and the set of points y_i.

$$c_{opt} = \min_c \sum_{i=1}^{k} (res_i)^2 = \min_c \sum_{i=1}^{k} (y_i - f(x_i, c))^2 \quad (42)$$

where $x_i = (x_{1,i}, x_{2,i}, ..., x_{m,i})$.

To find the set c_{opt}, the sum of squares of all k points is elaborated in vector form

$$S(c) = ||Y - f(x, c)||^2 \quad (43)$$

and starting from an initial set of parameters c_0 the problem is solved iteratively. At each iteration it is tried to find a better set of parameters $c_{new} = c + d$. The parameters in d are determined in each iteration, as it will lead to a new estimation of the parameters set c. The sum of squares for the new parameters is calculated as an approximation, as follows.

$$S(c + d) \approx ||Y - f(x, c) - J \cdot d||^2 \quad (44)$$

where J is the composition of all Jacobian vectors J_i. The Jacobian vectors J_i contain all derivatives of $y = f(x, c)$ to all parameters $c = (c_1, c_2, ..., c_n)$ at the points (y_i, x_i, c). Then the vector d, which minimizes the sum of squares, is calculated by setting the derivative of the sum of squares to d equal to zero.

$$\begin{aligned}
0 &= \frac{\partial S(c+d)}{\partial d} = \frac{\partial}{\partial d}||Y - f(x,c) - J \cdot d||^2 \\
0 &= \frac{\partial}{\partial d}[(Y - f(x,c) - J \cdot d)^T \cdot (Y - f(x,c) - J \cdot d)] \\
0 &= \frac{\partial}{\partial d}[(Y - f(x,c))^T \cdot (Y - f(x,c)) - 2 \cdot (J \cdot d)^T \cdot (Y - f(x,c)) + d^T \cdot J^T \cdot J \cdot d] \\
0 &= -2 \cdot J^T \cdot (Y - f(x,c)) + 2 \cdot J^T \cdot J \cdot d \\
J^T \cdot J \cdot d &= J^T \cdot (Y - f(x,c))
\end{aligned} \tag{45}$$

At this point the vector d could be calculated, by solving the linear system, which is reached. This would have the drawback that the parameters d would be too big, if the initial set of parameters c_0 is too far from the optimal solution c_{opt} [32]. Due to this the iterative algorithm would not converge. The damping factor λ is used to make the algorithm more robust against a bad initial guess. It is calculated depending on the change of the sum of squares S using a trust region method, as explained in detail in [32].

$$(J^T \cdot J + \lambda \cdot I) \cdot d = J^T \cdot (Y - f(x,c)) \tag{46}$$

The vector of the change of parameters d is in addition modified, using a scaling matrix $diag(J^T J)$, what again has a positive effect on the convergence and the speed of the algorithm [32].

$$(J^T \cdot J + \lambda \cdot diag(J^T J)) \cdot d = J^T \cdot (Y - f(x,c)) \tag{47}$$

Using this equation, the change of the parameters d is calculated iteratively, until the size of d or the change in sum of squares of the residuals is smaller than a defined bound. The last set of parameters is a local minimum of the sum of squares.

To elaborate the function f, which will be used as a function guess for the Levenberg-Marquardt algorithm, the MNA method is used, which was presented in Section 3.1.2. Active devices are replaced by simplified equivalent circuits, which include VCCS with some undefined parameters c. The functions of the VCCS will be exponential functions for the VCCS for diodes and BJTs, while for FETs a piecewise function is used, which contains the linear and quadratic terms of the terminal values. This is shown in detail in Section 5.1.

Using the Levenberg-Marquardt algorithm to curve-fit the transient simulation traces of the original circuit, its parameters c are determined. This approach has the advantage that the elaborated function guess is close to the analytical function, as all the passive element values are taken from the netlist, while the active devices are simplified, but still contain a good function guess. The number of parameters to determine is small, compared to the one of the Vandermodes interpolation approach.

5 Reduced Model Generation

This Chapter shows how to combine the techniques introduced in Chapters 3 and 4 to generate reduced circuit models, using the time traces of their SPICE simulation. The

methodology is shown in Figure 12. This methodology was successfully applied to three circuits, which are shown in Chapter 6. The figure shows all steps, which are required to generate a reduced model of a circuit, starting from its netlist and its transient simulation traces.

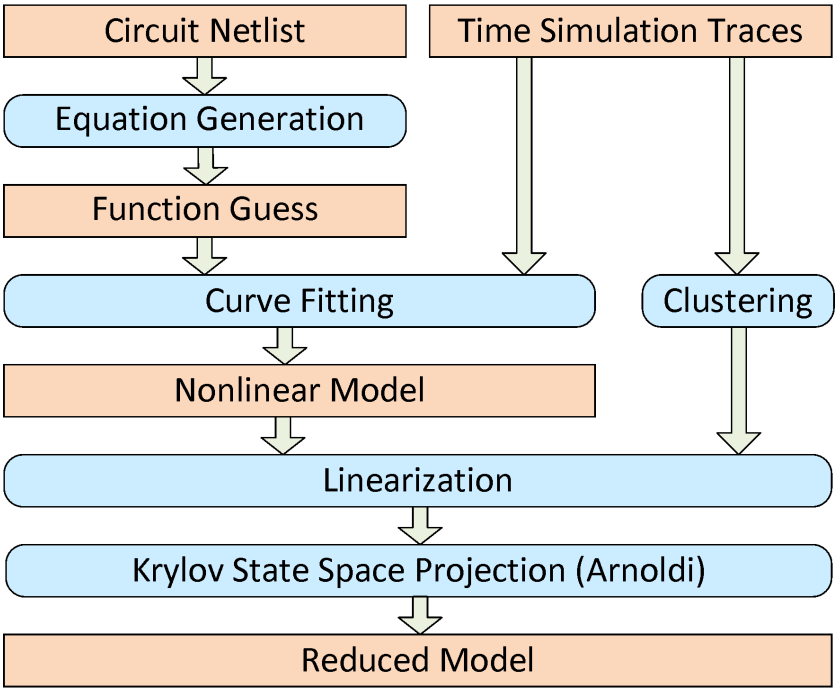


Fig. 12: Proposed Model Order Reduction Methodology

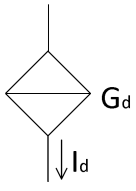
All steps for the model generation are implemented in MATLAB. They are explained in this Chapter. Most of the steps of the generation of the reduced model are automated.

5.1 Equation Generation

The function of the 'Equation Generation'-procedure is to elaborate a mathematical function, which can be used as a guess-function for the Levenberg-Marquardt curve-fitting.

As SPICE active model devices can be very detailed and include a huge number of physical and geometric parameters, they are replaced by an equivalent circuit, which just consists of the VCCS of the elements and assumes its parasitic elements, like connection resistances or junction capacitances, to be zero or constant. Three simplification of the equivalent circuit of a diode, a BJT and a FET and a possible parametrization of the VCCS are shown in the following:

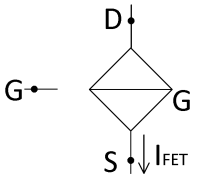
Diode



The diode is replaced by a single VCCS with two parameters c_1 and c_2.

$$I_D = c_1 \cdot e^{\frac{V_D}{c_2}} \tag{48}$$

FET

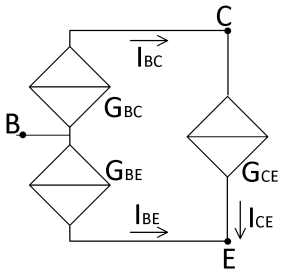


The SPICE level 1 FET model can be approximated by a single VCCS depending on two voltages. It contains two parameters. Here is shown the function for the n-channel FET. Depending on the

operating frequency, the parasitic capacitances have to be added.

$$I_{DS} = \begin{cases} 0 & if\ V_{GS} < c_1 \\ c_2 \cdot 0.5 \cdot (V_{GS} - c_1) & if\ V_{DS} > V_{GS} - c_1 > 0 \\ c_2 \cdot V_{DS} \cdot (V_{GS} - c_1 - \frac{V_{DS}}{2}) & if\ V_{DS} < V_{GS} - c_1 \end{cases} \tag{49}$$

BJT



The BJT can be replaced by three VCCS. For higher frequencies its small parasitic capacitances have to be considered.

$$\begin{aligned} I_{BC} &= \frac{c_1}{c_2} \cdot e^{\frac{V_{BC}}{c_4}} \\ I_{BE} &= \frac{c_1}{c_3} \cdot e^{\frac{V_{BE}}{c_4}} \\ I_{CE} &= c_1 \cdot \left(e^{\frac{V_{BE}}{c_4}} - e^{\frac{V_{BC}}{c_4}} \right) \end{aligned} \tag{50}$$

The shown equivalent circuits and the parameterizations of the VCCS-functions can be used to simplify the SPICE level 1 models of the diode, FET and BJT. For other SPICE models, the simplified equivalent circuits might have to be adapted.

Using this VCCS with undetermined parameters, all active devices are replaced. It is important that for active device models, which occur multiple times in the netlist, the same parameters are used. This will lead to a smaller number of parameters and also will support later the curve-fitting algorithm, to find optimal values for the parameters.

The passive elements of the circuit and the functions of the VCCS are used to elaborate an ODE-system as shown in Section 3.1.2. This ODE-system is the function guess that contains a number of parameters c, which will be determined by the Levenberg-Marquardt algorithm. It has the general form:

$$\dot{x} = f(x, u, c) \tag{51}$$

where $x \in \mathbb{R}^n$ are the circuits state variables, $u \in \mathbb{R}^p$ are the input signals, $c \in \mathbb{R}^m$ is a set of parameters and $f : \mathbb{R}^{n+p+m} \to \mathbb{R}^n$ is a nonlinear function.
The following table shows the necessary MATLAB functions, which will generate a function guess file, starting from the circuits netlist. The functions can be found on the CD in the Appendix.

Table 2: MATLAB Functions for the Equation Generation

Function	Input	Output	Description
replace_subs	Original SPICE netlist	New SPICE netlist with: - no subcicuits - modified active devices	- Read all subcircuit descriptions of the original netlist and generate a subcircuit data structure, which contains all internal nodes and elements. - Replace all subcircuit elements by their description. - Replace all active devices by G-elements (VCCS) with undefined parameters.

read_netlist	New SPICE netlist with: - no subcicuits - modified active devices	Elements Structure in MATLAB	- Read the SPICE netlist and add each element to a structure in MATLAB. - For the linear elements (R, L, C) save their values (in Ω, H, F) and nodes. - For the G-elements save the function, the controlling nodes and the connected nodes.
develop_MNA_matrices	Elements Structure	Nodes List, MNA Matrices: AGA, ACA, A_I	- Elaborate the numerical matrices, by using the information about the linear elements and the connection nodes. - Elaborate a nodes list.
develop_I	Elements Structure	String Array I	- Development of a string array, which consists of the functions of the VCCSs.
develop_para_file	MNA Matrices, Nodes List, String Array I	Function Guess File	- Development of a MATLAB file, which contains the function guess for the whole circuit and which can be called from the curve-fitting algorithm.

5.2 Curve-Fitting

For the curve-fitting, the Levenberg-Marquard algorithm described in Section 4.2.2 is used. This algorithm is available in MATLAB as an option of the function

`lsqcurvefit`.

The SPICE simulation traces of the state space signals are read form the SPICE output file by a MATLAB function. They are arranged, to correspond to the state variables of the guess function. For a successful curve-fitting, the time traces have to show the transient behaviour of the circuit, which means, they have to change over time. If a signal stays constant, its derivative will be zero and these sample points cannot be used for the curve-fitting. To prepare a good curve-fitting, regions in the time traces, where a signal has a high gain or decline, have to be chosen instead of regions, where the signal stays constant.

The Levenberg-Marquard algorithm will return the set of parameters c, which is the closest minimum of the sum of the residuals to the initial guess c_0. To find the best set of parameters c_{opt}, it can be necessary to try systematically different sets of input parameters. As the curve-fitting algorithm will also return the sum of the residuals, which remains for the calculated set c_{opt}, it is easy to identify the best set of parameters.

The output of the curve-fitting is a set of parameters, which will give an approximated ODE-description of the circuits state variables. It will transform Equation (51) to the nonlinear model of the circuit.

$$\dot{x} = f(x, u) \tag{52}$$

where $x \in \mathbb{R}^n$ are the circuits state variables, $u \in \mathbb{R}^p$ are the input signals and $f : \mathbb{R}^{n+p} \rightarrow \mathbb{R}^n$ is a nonlinear function.

5.3 Linearization and State Space Transformation

This section explains how the MOR technique for nonlinear dynamical systems, which was shown in Section 3.2.2, is applied in the generation of reduced analog circuit models, within the methodology.

The time traces of the SPICE transient simulation are not only used to find a simplified ODE-function, but also for the piecewise linearization of the nonlinear system. The sample points of the trajectories are clustered by the kmeans algorithm into a predefined number k of clusters. For n state variables the algorithm will start from an initial set of k points in $\mathbb{R}^n$, which can be random or user-defined and which are used as initial cluster center points. Iteratively the algorithm will combine all points, which are close to each other in one cluster. Therefore, in each iteration the points are assigned to the cluster, whose center is the closest to them and the new cluster center points are calculated from all points into the particular cluster. The algorithm stops, when there is no change in the clusters anymore, or when a defined maximum number of iterations is done [34].

The determination of the number of linearization points is one important issue for the automation of the MOR of nonlinear systems. It is known, that a main factor for the number of linearization points is the number of state variables. Nevertheless, until now the number of necessary linearization points cannot be determined automatically, as it also depends on factors like the operation area of the circuit, the existence of sharp flanks in the input signal and the structure of the circuit. To get a fast reduced model, it is necessary to determine the number of clusters, which is high enough to transform it from a nonlinear to a piecewise linear system as shown in Section 3.2.2. But also the number has to be small enough, to not slow down the simulation, as with a high number of piecewise systems the estimation of the proper system, in each time step of the numerical simulation, would be more time consuming.

After the k linearization points x_L are determined by using the sample points, the nonlinear ODE-system is linearized using a numerical differentiation function in MATLAB [35]. This function will elaborate the Jacobean matrix of the ODE-system at each linearization point.

Holding the piecewise linear approximation, the system state space is reduced using the Arnoldi algorithm and SVD, as described in Sections 3.2.1 and 3.2.2. The number of state variables in the reduced state space is chosen as low as possible, to get a high speedup, but guarantee a good model quality.

5.4 Reduced Model Quality

In this section the requirements on a reduced model are discussed. There are two main expectations on the reduced model.

It is required essentially that the reduced model approximates the input-output behaviour of the original model precisely. The reduced model must keep the input sensitivity of the original model for all possible input signals, or at least for a big range of input signals, to make it useable in practice. To validate the input-output behaviour, both the original and the reduced model, have to be simulated with the same input signals. The error of the output signal, which is given in the next chapter for three example circuit simulations, is

calculated, using the following equation.

$$error = \sum_{i=1}^{n_o} \frac{\int_0^{T_{sim}} x_{o,i}(t)\ dt - \int_0^{T_{sim}} \widehat{x}_{o,i}(t)\ dt}{\int_0^{T_{sim}} x_{o,i}(t)\ dt} \tag{53}$$

where T_{sim} is the simulation time, n_o is the number of output signals, $x_{o,i}(t)$ are the time traces of the output signals of the original model simulation and $\widehat{x}_o(t)$ are the output signal time traces of the simulation of the reduced model transformed back to the original state space.

The second criteria is the speedup, which is reached, if the reduced model is used instead of the original model. The speed of the simulation is mainly influenced by the following effects. The first factor is the numerical algorithm itself. To have comparable results, the same backward differentiation algorithm in MATLAB is used for both, the original and the reduced model simulation. The second factor is the number of calculations per timestep. The reduced model requires less steps than the original, as the number of variables to calculate is smaller. But the determination of the closest linearization point, which is done in every time step in the simulation of the reduced model, is time consuming. The third important factor is the number of timesteps. As all advanced solvers control their internal step size independently from the user, by ensuring, that the local error is in a defined range, the user does not influence the number of steps. For both simulations the same local error bound is used, also this might force the reduced model simulation to slow down, because of a smaller time step size. To measure the speedup in the following section, we compare the computation time of the original model to the one of the reduced model, when they are simulated in MATLAB. The duration is not compared to the SPICE simulation time, as it does not use the same integration algorithm and time step control. The original model in MATLAB is an MNA model, where the VCCSs of the active devices are using detailed parameters, similar to the level 1 models in SPICE.

The speedup given for the applications in Chapter 6 is gained through model simplification, linearization and reduction.

6 Applications

In this Chapter the proposed methodology is applied to three circuits, namely a voltage controlled oscillator (VCO), a two stage operational amplifier (OA) and a frequency divider. It is shown, that the generated reduced models are simulated much faster than the original models, while the error due to the reduction stays small. The speedup, shown for each circuit, compares the simulation time in MATLAB for a circuit model, using a transistor model similar to SPICE level 1 model, and the reduced model, which was generated, using curve-fitting of the SPICE simulation traces, and state space transformation.

The simulations were executed on a machine with an 'Intel core i7' CPU, 2.8 GHz and 24 GB of RAM.

6.1 Voltage Controlled Oscillator

The VCO is a widely used device, which provides a rectangle output voltage, whose frequency is controlled by its analog input voltage. VCOs are utilized wherever an adaptive clock-signal is required, e.g., in PLLs. The shown VCO is realized as a ring oscillator of an uneven number of inverters. The delay time of the inverters and so the oscillating frequency

depends on the analog input signal. Figure 13 shows the VCO implementation. The circuit is implemented in CMOS $90nm$ technology and works with a V_{dd}-level of $1.8V$.

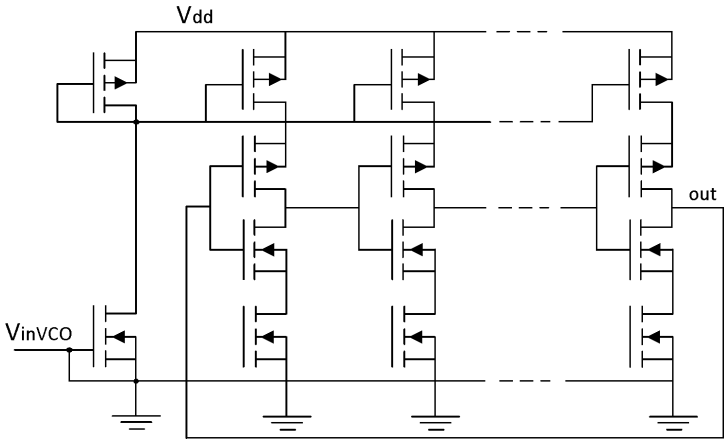


Fig. 13: VCO Circuit

As there are two different transistor models in the circuit, one n-channel and one p-channel FET, the number of parameters to determine by curve-fitting is four, as Equation (49) is used to simplify the FETs. The two transistor models used by the original SPICE simulation are shown in the following.

```
.model nmFET nmos (level=1 L=40u W=80u vto=0.475231 KP=450u PHI=0.576
LAMBDA=0.0023 KF=0 AF=1)
.model pmFET pmos (level=1 L=3u W=15u vto=-0.44922 KP=120u PHI=0.5276
LAMBDA=0.0012 KF=0 AF=1)
```

The curve-fitting procedure will determine the four parameters of the function guess to the following.

$$c_1 = 0.5225 \qquad c_2 = 7.2 \cdot 10^{-4} \qquad c_3 = 0.6416 \qquad c_4 = 10.5 \cdot 10^{-4}$$

Using these parameters, the elaboration of the nonlinear ODE-system is completed. If the system, derived from the curve-fitting, is simulated, it shows the same transient behaviour as the SPICE simulation. This is shown in the following figure, where the time trace of the voltage of the first inverter output is presented.
The ODE-system is linearized at 36 points to a piecewise linear system and then reduced from a state space of 46 variables to a model with 33 state variables, using the Arnoldi algorithm and SVD. The mathematical model description of the nonlinear ODE-system and the reduced system is shown in the following equations and table.

Nonlinear ODE-System derived from MNA

$$\dot{\vec{e}} = ACA_{cut}^{-1} \cdot \left(-AGA_{cut} \cdot \begin{bmatrix} V(t) \\ \vec{e} \end{bmatrix} - A_{I,cut} \cdot I(\vec{e}) \right) \tag{54}$$

Reduced Model

$$\dot{\vec{z}} = \vec{z_i} + A_i \cdot (\vec{z} - \vec{z_i}) + B_i \cdot V(t) \tag{55}$$

Transformation

$$\vec{z} = U^T \cdot \vec{e} \qquad \vec{e} = U \cdot \vec{z}$$

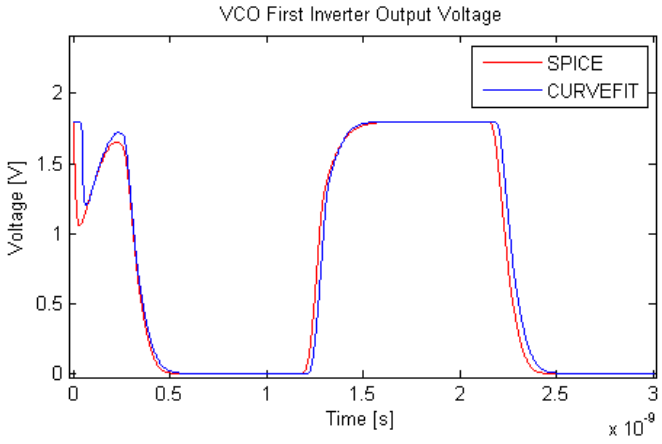


Fig. 14: Results of the VCO Curve-Fitting

Table 3: Dimensions of Original and Reduced Mathematical Models of the VCO

Original model	ACA_{cut}^{-1}	AGA_{cut}	$V(t)$	$\vec{e}$	A_I	$I(\vec{e})$
	48×48	48×50	2×1	48×1	48×62	62×1
Reduced model	R	A	$\vec{z}$	B	$V(t)$	
	33×1	33×33	33×1	33×2	2×1	

To show the input sensitivity of the reduced model, an input signal step from $0.9V$ to $1.1V$ at time $4.0ns$ is simulated.

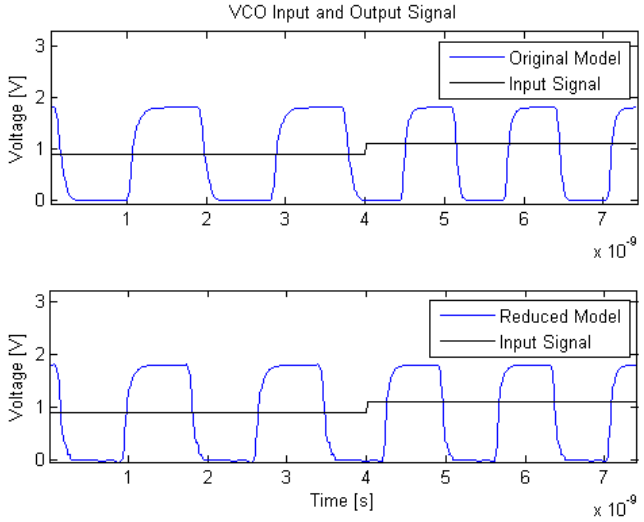


Fig. 15: Original and Reduced VCO Model - Output Signal for Input Signal Step

Both the reduced and original model output frequencies are faster for a higher input signal. The difference between the two output signals is mainly due to a small phase shift of the signals at the beginning of the simulation. The frequencies for the different input signal levels are almost the same.

The next table shows the error of the reduced model output, compared to the original model, calculated using Equation (53), for three different simulations. The three different

simulations are using different reduced models.

Table 4: VCO Simulation Results

Input Signal	Simulated Time	Simulation Time $[s]$		Speedup	Error
$[V]$	$[ns]$	Org.	Red.		$[10^{-2}]$
const. 1.2	3.00	5.57	1.60	3.47	0.97
const. 0.9	3.00	4.05	1.91	2.12	0.75
step 0.9 to 1.1	7.50	11.4	3.78	3.01	0.64

If the constant input voltage of the original model simulation, whose traces were used to linearize the nonlinear model, is applied to the reduced model, there is almost no difference in the simulation result, as shown in the following figure. If the input signal changes, the output signal differs a little bit, but the reduced model shows also the same behaviour as the original model (higher input will force a higher output frequency).

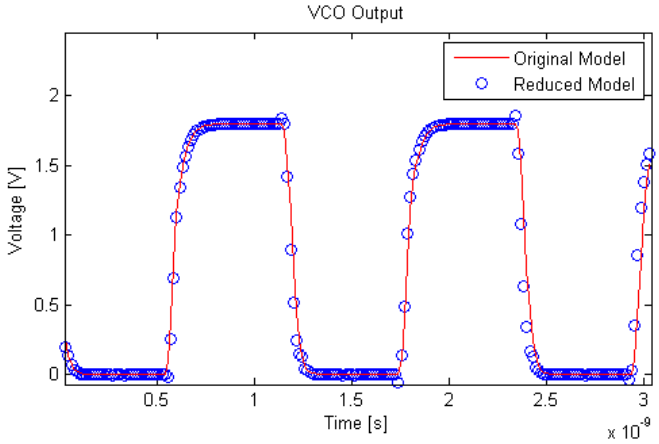


Fig. 16: Output Signal of VCO for Constant Input Voltage of $1.2V$

6.2 Operational Amplifier

The two stage operational amplifier is implemented using four different types of FET transistors (all SPICE level 1), as shown in Figure 19. Its bias current is provided by a constant current source, built by a n-channel FET and a resistance. The open loop gain of the amplifier is approximately 10^5 at $1MHz$. For the simulation, the operation area of the amplifier is between $V_{dd} = 2V$ and $V_{ss} = -2V$.

The connections between the transistors are modeled by resistances to consider their parasitic behaviour. This leads to a state space with 30 state variables, which all are node voltages. The number of parameters, which have to be determined by the least-squares curve-fitting, is eight, as there are four different types of transistors in the circuit.

Problematic is the curve-fitting of the parameters of the transistor of the constant current source. The FET current and its terminal voltages are constant during the whole simulation, independently from the input signal. To curve-fit the parameters of the first transistor only the first $4.5ps$ can be used. If the simulation has a time step bigger than this, a curve-fitting

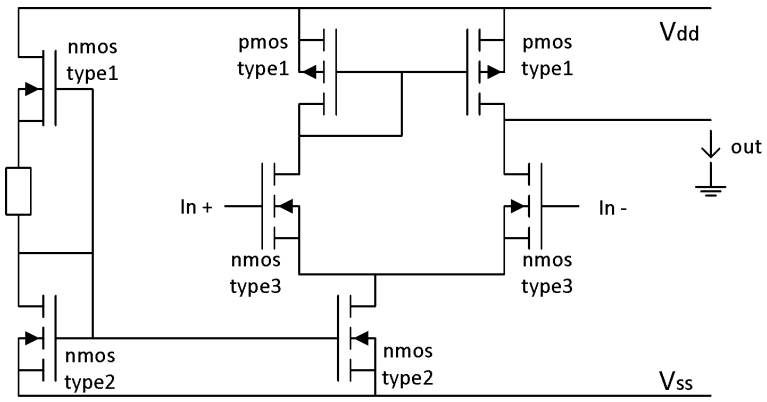


Fig. 17: Two Stage Operational Amplifier Circuit

of the parameters of the first transistor would be impossible. For smaller time steps, a curve-fitting of the complete circuit is possible. If the simulation time step is too big, the model of the first transistor has to be calculated from its netlist parameters.

The parameters, derived from the curve-fitting will give the ODE-system of the circuit. Similar to the voltage controlled oscillator, the nonlinear ODE-model is a very good approximation of the original system.

The following figure shows the output signals of the SPICE simulation and the nonlinear ODE-system, generated by using simplified equivalent circuits. The amplifier is simulated without feedback and load. Its negative input is grounded and its positive input is a sine with an amplitude of $18\mu V$.

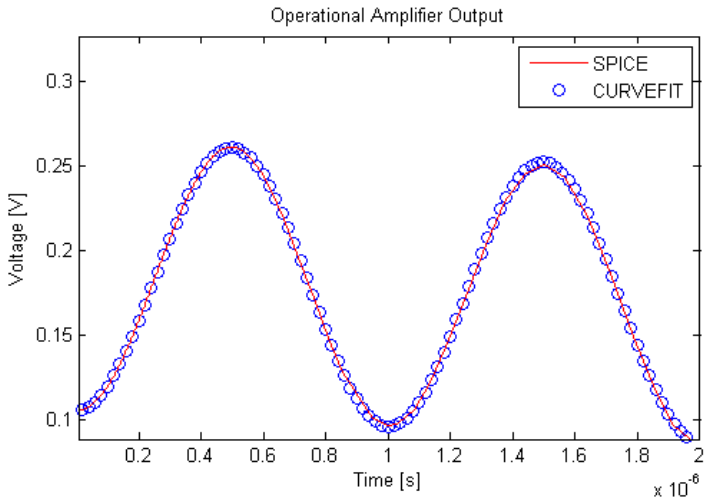


Fig. 18: Curve-fitting Results for Operational Amplifier

A reduced model of the amplifier is generated from the nonlinear ODE-description for two different amplifier circuits. The first case is the pure amplifier with an open loop and a grounded negative input. For the second circuit two resistances are added to the operational amplifier to reach a lower, specified gain. This circuit is shown in the following figure.
To linearize the amplifier circuit, 79 linearization points are required for the open loop circuit, while for the circuit with feedback, just 30 points are enough. Also the open loop circuit cannot be reduced as much as the circuit with the feedback loop. The size of the original model for both circuits is the same, as no capacitive nodes or inductive branches are added

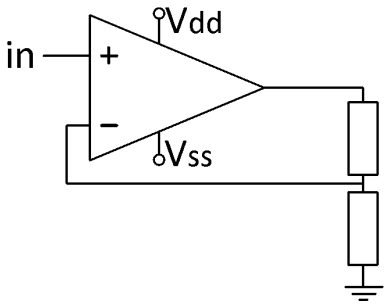


Fig. 19: Operational Amplifier with Feedback

to the circuit. The dimensions of the original model and the reduced models for the circuit with and without feedback loop are shown in the following table. The original and reduced models are the same as the one shown in Equations (54) and (55).

Table 5: Dimensions of Original and Reduced Mathematical Models of the CMOS OA

Original Model	ACA_{cut}^{-1}	AGA_{cut}	$V(t)$	$\vec{e}$	A_I	$I(\vec{e})$
	30×30	30×34	4×1	30×1	30×7	7×1
Red. Model (Open Loop)	R	A	$\vec{z}$	B	$V(t)$	
	27×1	27×27	27×1	27×4	4×1	
Red. Model (Feedback)	R	A	$\vec{z}$	B	$V(t)$	
	25×1	25×25	25×1	25×4	4×1	

The simulation of the reduced models show a significant speedup compared to the simulation of the original model in MATLAB, as shown in Table 6.

Table 6: Operational Amplifier Simulation Results

Circuit	Input Sine		Simulated Time	Simulation Time $[s]$		Speedup	Error
	f [MHz]	$\widehat{V}$ [V]	$[\mu s]$	Original	Reduced		$[10^{-2}]$
Open Loop	1	18μ	6	5.43	2.17	2.51	0.36
Feedback	1	$45m$	6	142.20	4.13	34.43	0.30
	0.5	$45m$	6	40.4	1.27	31.6	0.01

The speedup for the amplifier with the feedback loop is significantly higher then the speedup of the open loop amplifier. This is due to the higher number of linearization points, which corresponds to the number of piecewise systems, from which the closest has to be chosen in every time step during the numerical simulation. The second reason for the higher speedup of the circuit with the feedback, is the size of its reduced model, which has a smaller state space than the reduced model for the open loop amplifier, as shown in Table 5.

The difference between the simulation results of the reduced and original model is very small and not noticeable in the time traces of the output signal.

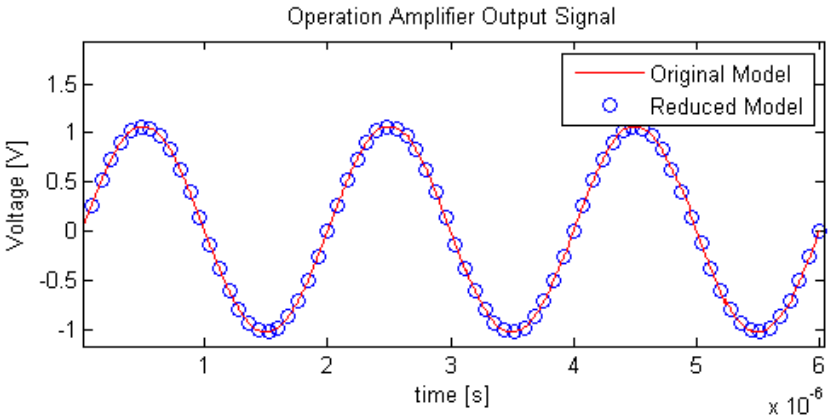


Fig. 20: Output Signal of OA with Feedback for Sine Input ($\hat{V} = 45mV, f = 0.5MHz$)

6.3 Flip-Flop Frequency Divider

The frequency divider is implemented as a data-flip-flop (DFF) in a CMOS 90 nm technology. The DFF is edge triggered and will operate at the high flank of the clock signal. Its inverted output terminal is feed back to the input. The output terminal will provide a signal with half the frequency of the clock input. This circuit is one of the basic circuits in many digital circuits, e.g., synchronous or asynchronous counters.

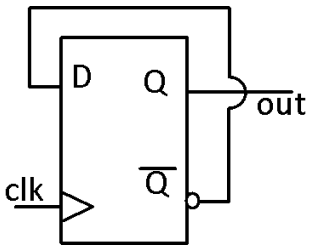


Fig. 21: DFF as Frequency Divider

Internally the DFF consists of eight NAND and two inverter gates. The implementation of the DFF by the logic gates and the device level implementation of the logic gates is shown in Figure 22.

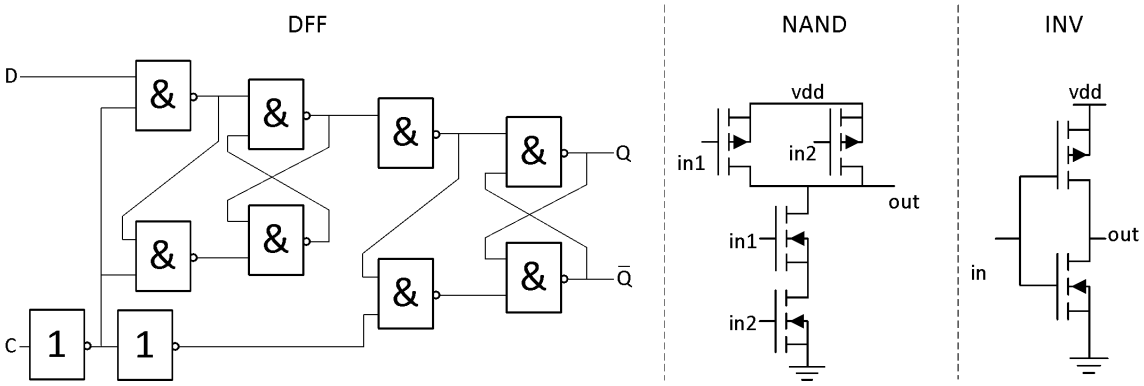


Fig. 22: Edge Triggered DFF - Internal Circuit and Logic Gates Implementation

For the whole implementation of the DFF two different types of transistors are used, one p-channel and one n-channel FET. As the input signal is a rectangle voltage and also all the internal signals change periodically, with sharp flanks, the curve-fitting to find the

parameters of the VCCS is not difficult. The four parameters, resulting from the equation generation of the circuit, using the simplified FET model, are calculated by curve-fitting. The resulting ODE-model is simulated using the MATLAB ODE-solver. The output signals of both simulations are shown in Figure 24.

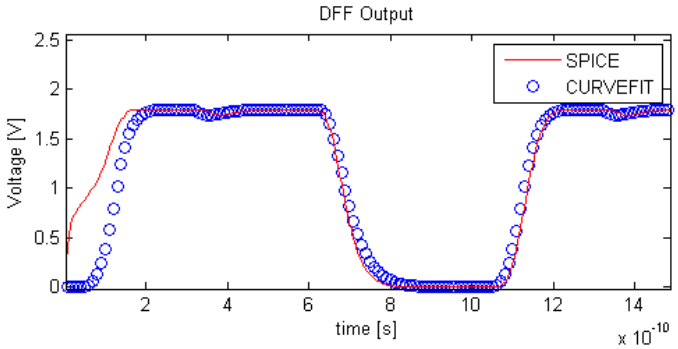


Fig. 23: Curve-Fitting Results for Edge Triggered DFF

Beside the activation operation, the DFF model using the SPICE level 1 FET and the elaborated model, which uses the parameters derived from curve-fitting, show the same behaviour.

To linearize the ODE-model, 44 points are used. The state space transformation will reduce the number of state space variables from 53 to 48. The size of the ODE-description of the circuit in its original and reduced state space are shown in the following table.

Table 7: Dimensions of Original and Reduced Model of the Frequency Divider Circuit

Original model	ACA_{cut}^{-1}	AGA_{cut}	$V(t)$	$\vec{e}$	A_I	$I(\vec{e})$
	53×53	53×55	2×1	53×1	53×36	36×1
Reduced model	R	A	$\vec{z}$	B	$V(t)$	
	48×1	48×48	48×1	48×2	2×1	

If the reduced model is used to simulate the DFF instead of the model in the original state space, the simulation time is reduced as shown in the following table.

Table 8: Frequency Divider Simulation Results

Input Frequency	Simulated Time	Simulation Time $[s]$		Speedup	Error
$[GHz]$	$[ns]$	Original	Reduced		$[10^{-2}]$
2.0	5.00	19.50	6.07	3.21	0.89
1.0	8.00	33.6	10.0	3.36	0.72
0.5	12.00	24.52	7.49	2.90	0.70

Also for this circuit, the reduction has no visible influence on the simulation results. The following figure shows the output of both the original and reduced model, for a clock input signal of $1GHz$.

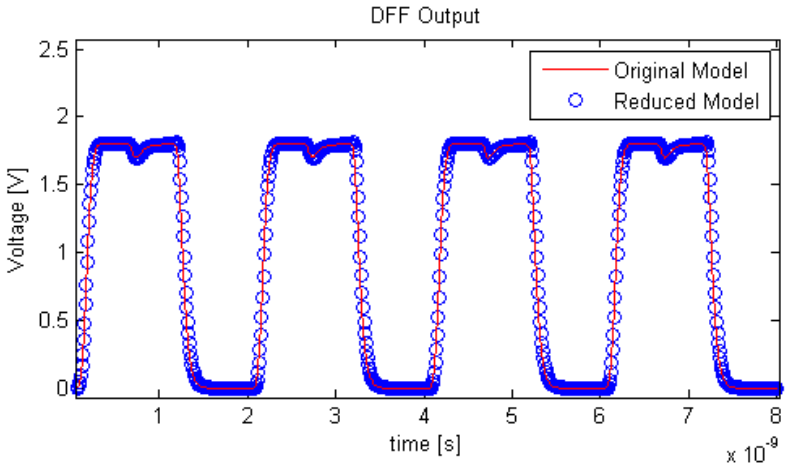


Fig. 24: Output Signal of the Frequency Divider Models for a $1GHz$ Clock Signal

6.4 Discussion

The most important criteria for a model are its accuracy and its simulation time. The generated models, for all three presented circuits, show the same behaviour as the original models. Also the models are generated using transient time traces, resulting from a specific input of the circuit, they are sensitive for a large range of input signals. This is demonstrated especially by the simulations of the operational amplifier and the flip-flop circuit. The errors between the simulations of the original and reduced models are always below 1%. Differences between the simulation results of the original and the reduced model are often not visible. The small errors are mainly due to the numerical techniques, used for the linearization and the reduction of the models. The curve fitting, which is used for the parametrization of the active devices, is very accurate, because of the advanced function guess, which is developed by using MNA and simplified active device equivalent circuits.

The speed of the simulation of the simplified and reduced models is in every case at least two times higher than the one of the original ODE-system, which has a bigger state space and uses detailed active device models, similar to the SPICE level 1 model. In some cases the speedup is very high, up to 34 times. The difference in the speedups is due to the circuits themselves. The effort to linearize the model, reflected by the number of linearization points, is one main criteria for the speedup. The second criteria is the reduction itself. The degree of reduction, calculated with the number of state variables, is between 10% and 31% for the shown circuits.

All files for the curve-fitting processes, the original and reduced circuit models and the simulations-files can be found on the CD in the Appendix.

7 Conclusion

7.1 Summary

The objective of this work was to investigate whether it is possible to automate the reduction of electronic circuit models using their SPICE simulation traces. Therefore, the traces of the state space variables of the circuit are used to elaborate a mathematical model of the circuit. This model, which is an ODE equation system, is used as a basis to elaborate a reduced model using state space transformation.

To find the mathematical description of the circuit, the function of the time derivatives of the state variables, depending on the state variables themselves and the input signals has to be found. Therefore, the derivatives are extracted and the resulting least-squares problem has to be solved. Different methods are discussed, which can be applied to linear and nonlinear systems, to curve-fit the time traces of the derivatives. It is shown that the full determination of a linear system is possible, if the step size of the time traces is small enough. Unfortunately, for nonlinear systems such simple methods are not available. In the case of a nonlinear circuit, a good function guess has to be elaborated to approximate the derivative of the traces. To elaborate the function guess an MNA technique is used, which is automated in MATLAB. It replaces active devices by simplified models, whose parameters are determined by the curve-fitting. The resulting ODE-system shows the same transient behaviour as the original SPICE simulation, as demonstrated on three examples. The use of curve-fitting to find a mathematical description of the circuit, instead of computing it with the SPICE models, is useful, if the whole SPICE model description is not available or it uses to many unknown parameters.

Holding the nonlinear ODE-description of the circuit, the system is linearized and reduced, to reach a model with a low simulation time. Therefore, the Arnoldi algorithm and SVD are used. To linearize the system, the samples of the simulation time traces of the SPICE simulation are clustered. The number of linearization points, needed to transform the nonlinear model to a piecewise linear model, should be as low as possible, to guarantee a fast reduced model, but high enough to get a proper approximation of the nonlinear behaviour. The necessary linearization of the system is a drawback of the shown methodology. While Krylov subspace techniques can deal with linear systems including thousands of state variables[3, 22, 36], for the proposed methodology for nonlinear circuits, it is often not possible, to deal with models of such a dimension. This is due to a high number of piecewise linear systems, which is necessary to replace the strong nonlinear system. If the number of required linearization points is larger than 100, it must be proven, whether this high number still allows a speedup of the simulation.

If the linearization of the circuit is possible, with an adequate number of piecewise systems, the proposed methodology works fine, as demonstrated on three common circuits. The generated models have the same behaviour as the original models, and reach very good improvements regarding simulation time.

The biggest part of the model generation is automated. Just a calculation of the number of linearization points and the state space size of the reduced models have to be done iteratively.

7.2 Future Work

To fully automate the MOR of nonlinear analog electronic circuits, using Krylov subspace methods and SVD two main issues have to be solved.

The first problem is the linearization of the circuit. Different circuits require a different

number linearization points. An analytical calculation of this number or at least an estimation of the number, based on the size of the circuit, its simulation traces and maybe the ratio of linear to nonlinear devices, is required essentially. Without the knowledge of the number of piecewise systems a full automation is not possible.

The second weak point of the proposed methodology is the determination of the size of the subspace. This means that the Arnoldi algorithm has to be modified, to automatically stop if a moment matching of the most important moments is finished. It would be a big step towards the automation of reduced systems, if such a break condition could be applied successfully.

To gain a faster speedup of the reduced models, the determination of the closest model in each time step of the simulation has to be calculated faster. This can be done for example by limiting the systems taken into account, for the calculation of the closest linear system, by grouping them into subgroups, or using weight functions. This means, that not only the closest linearization point, but maybe the closest two or three are used for the derivative calculation. This technique should also reduce the number of necessary linearization points.

References

[1] R. Myslewski, "Moore's law 'alive and well'," *The Register*, 2013.

[2] P. Benner, H. M., and E. J. W. ter Maten, *Model Reduction for Circuit Simulation*. Springer, 2011.

[3] W. Schilders, H. van der Horst, and J. Rommes, *Model Order Reduction: Theory, Research Aspects and Applications*. Springer, 2008.

[4] M. Rewienski and J. White, "A Trajectory Piecewise-Linear Approach to Model Order Reduction and Fast Simulation of Nonlinear Circuits and Micromachined Devices," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 22, no. 2, pp. 155–170, 2003.

[5] H. Aridhi, M. H. Zaki, and S. Tahar, "Towards Improving Simulation of Analog Circuits Using Model Order Reduction," *IEEE/ACM Design Automation and Test in Europe*, pp. 1337–1342, Mar 2012.

[6] N. Dongi and J. Roychowdhury, "General-Purpose Nonlinear Model-Order Reduction Using Piecewise-Polynomial Representations," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 27, no. 2, pp. 649 – 654, 2008.

[7] J. Phillips, L. Daniel, and L. Silveira, "Guaranteed Passive Balancing Transformations for Model Order Reduction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 8, pp. 1027–1041, 2003.

[8] P. Heydari and M. Pedram, "Model-Order Reduction Using Variational Balanced Truncation with Spectral Shaping," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 4, pp. 879–891, 2006.

[9] R. Prasad, "Pade Type Model Order Reduction for Multivariable Systems Using Routh Approximation," *Computers and Electrical Engineering*, vol. 26, no. 6, pp. 445 – 459, 2000.

[10] Y. C. L. et al., "Proper Orthogonal Decomposition and its Applications," *Journal of Sound and Vibration*, 2002.

[11] M. Ahmadloo and A. Dounavis, "Parameterized Model Order Reduction of Electromagnetic Systems Using Multiorder Arnoldi," *IEEE Transactions on Advanced Packaging*, vol. 33, no. 4, pp. 1012–1020, 2010.

[12] R. W. Freund, "Krylov-Subspace Methods for Reduced-Order Modeling in Circuit Simulation," *Journal of Computational and Applied Mathematics*, vol. 123, no. 1-2, pp. 395 – 421, 2000.

[13] A. Steinbrecher and T. Stykel, "Model Order Reduction of Electrical Circuits with Nonlinear Elements," in *Progress in Industrial Mathematics at ECMI 2010* (M. Günther, A. Bartel, M. Brunk, S. Schöps, and M. Striebel, eds.), Mathematics in Industry, pp. 169–177, Springer, 2012.

[14] I. Aris, L. Hulley, N. Mariun, and R. Sahbudin, "Using Curve-Fitting Optimisation Technique to Estimate Power MOSFET Model Parameters for PECT II System," in *1998. Proceedings. ICSE '98. 1998 IEEE International Conference on Semiconductor Electronics*, pp. 157–161, 1998.

[15] *SPICE Model Generator*. Agilent Technologies, 395 Page Mill Road, Palo Alto, CA 94304 USA, 2013.

[16] R. Duraiswami, "Computational Methods, Lecture Notes," tech. rep., University of Maryland, Jan 2013. *http : //www.umiacs.umd.edu/ ramani/cmsc*460/ *Lecture*1_*Introduction*_2013.*pdf*.

[17] L. Nagel, *SPICE2: A Computer Program to Simulate Semiconductor Circuits*. Memorandum, Electronics Research Laboratory, College of Engineering, University of California, USA, 1975.

[18] T. Tuma and A. Burmen, *Circuit Simulation with SPICE OPUS: Theory and Practice*. Birkhäuser, 1st ed., 2009.

[19] *Avant! Star-Hspice Manual*. 46871 Bayside Parkway Fremont, CA 94538, 1998.

[20] C. Desoer and E. Kuh, *Basic Circuit Theory*. McGraw-Hill Education (India), 2009.

[21] F. N. Najm, *Circuit Simulation*. Wiley, 2010.

[22] T. Reis and T. Stykel, "Modellreduktion von RC Schaltungen," tech. rep., Elgersburg Workshop Mathematische Systemtheorie, Elgersburg, 2013.

[23] C. Hymowitz, "Step-by-Step Procedures Help you Solve Spice Convergence Problems," *EDN*, 1994.

[24] L. Trefethen and D. Bau, *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

[25] M.-H. Lai, C.-C. Chu, and W.-S. Feng, "MIMO Interconnects Order Reductions by Using the Global Arnoldi Algorithm," in *IEEE International Symposium on Circuits and Systems*, pp. 4 pp.–1110, 2006.

[26] Z. Yacine, A. Benfdila, and S. Djennoune, "Model Order Reduction of Nonlinear Systems via the Trajectory Piecewise Linear Approach Application to Transmission Lines Circuit," in *New aspects of applied informatics and communications Part II*, pp. 369–375, 1998.

[27] A. C. Antoulas and D. C. Sorensen, "Approximation of Large-Scale Dynamical Systems: An Overview," *Applied Mathematics and Computer Science*, vol. 11, no. 5, pp. 1093–1122, 2001.

[28] P. Sen and G. I. of Technology, *Estimation and Optimization of Layout Parasitics for Silicon-based Millimeter-Wave Integrated Circuits*. Georgia Institute of Technology, 2007.

[29] D. W. Harder, "Numerical Analysis, 5.1 Vandermonde," tech. rep., University of Waterloo, 2013.

[30] E. W. Weisstein, "Vandermonde Matrix," tech. rep., MathWorld, 2013. http://mathworld.wolfram.com/VandermondeMatrix.html.

[31] H. Gavin, "The Levenberg-Marquardt Method for Nonlinear Least Squares Curve-Fitting Problems," Sept. 2013.

[32] M. K. Transtrum and J. P. Sethna, "Improvements to the Levenberg-Marquardt Algorithm for Nonlinear Least-Squares Minimization," *arXiv preprint arXiv:1201.5885*, 2012.

[33] D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *Journal of the Society for Industrial & Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[34] E. W. Weisstein, *K-Means Clustering Algorithm*. MathWorld-A Wolfram Web Resource, http://mathworld.wolfram.com/K-MeansClusteringAlgorithm.html, 2013.

[35] J. DErrico, "Adaptive Robust Numerical Differentiation," tech. rep., MATLAB Central, 2013.

[36] P. Miettinen, M. Honkala, J. Roos, and M. Valtonen, "PartMOR: Partitioning-Based Realizable Model-Order Reduction Method for RLC Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 3, pp. 374–387, 2011.