Overview of Circuit-Simulation Activities at TKK CTL*

Janne Roos[1]

Helsinki University of Technology, Department of Electrical and Communications Engineering, Circuit Theory Laboratory, P.O.Box 3000, FI-02015 TKK, Finland.
janne@ct.tkk.fi

1 Introduction

This paper summarizes the recent circuit-simulation activities [Roo04]–[Sil06] at Helsinki University of Technology (TKK), Circuit Theory Laboratory (CTL). This paper is mostly based on the results of the national projects Advanced Radio Frequency SImulation and Modeling (ARFSIM 2002–2003) [Roo04], MOdeling and Simulation for Advanced Integrated Circuits and Systems (MOSAICS 2004–2005), and Accurate Models Aim for Zero Errors (AMAZE 2006–2008). All these projects have been funded by the National Technology Agency of Finland, Nokia Corporation, and AWR–APLAC Corporation; the annual volume at TKK CTL has been 4.0–5.5 man years. In these projects, APLAC circuit simulation and design tool [A06] has been used as a common platform for the circuit analysis and modeling methods developed.
This paper is organized as follows. Section 2 very briefly reviews transistor-model development. Section 3 lists our recent work with various analysis methods. In the following two sections, the current research interests of the author are treated in some more detail: Sections 4 and 5 discuss model-order reduction and behavioral modeling, respectively. Finally, Section 6 briefly summarizes that part of our recent research that has been carried out outside the ARFSIM, MOSAICS, and AMAZE projects.

2 Transistor models

During the ARFSIM, MOSAICS, and AMAZE projects, the C-code implementation of several BJT, MESFET [Kal02], and MOSFET semiconductor models has been improved. Also, an attempt has been made to make the transistor-model development more fluent: both a C-code model interface and a Philips SiMKit adapter have been implemented in APLAC. What comes to fundamental research, a new rule for MESFET gate-charge division based on the energy-conservation principle has been presented [KV04].

* Invited Paper at SCEE-2006

3 Analysis methods

During the ARFSIM, MOSAICS, and AMAZE projects, the following analysis methods have been studied and/or developed and/or implemented in APLAC:

- DC
  - speed/convergence improvements based on industrial feedback
  - piecewise-linear solution algorithm [RVV02], [Roo05], [Roo06]
  - nonmonotone norm-reduction method [Hon02a], [Hon02b]
  - nonlinear iteration/optimization methods [HRK06]
  - homotopy methods [Lin06]
  - parallel hierarchical analysis [Hon02c], [HK02], [KH02], [Hon03]
- AC
  - minor improvements
- Transient
  - event-based time-step control
  - truncation-error criteria
  - treatment of transmission lines
  - parallel hierarchical analysis [Hon03]
  - optimization of C-code implementation
- Multi-tone Harmonic Balance (HB)
  - reducing the memory consumption and increasing speed
  - efficient formulation of HB equations [Vir05]
  - inexact-Newton method with GMRES solver [Vir05]
  - nonmonotone norm-reduction method [Hon02b]
  - transient-assisted HB
  - multi-dimensional frequency mappings
  - sampling of nonlinear component-model functions [Vir05]
  - parallelization using threads [KH04]
  - oscillator analysis [Vir05]
  - frequency-divider analysis [Poh06]
- Multi-variate steady state time domain
  - GMRES preconditioners [LVV03], [Leh03]
  - multi-grid approach [Leh03]
- Large-signal–small-signal
  - GMRES preconditioners
  - amplitude/phase noise analysis [Vir05]
- Envelope
  - self-starting polynomial collocation/projection ODE-solver
  - MATLAB–APLAC prototype implementation
- Finite Difference Time Domain (FDTD)
  - FDTD–circuit/system co-simulation [Cos05]
  - optimization of C-code implementation [Cos05], [Cos06]

4 Model-order reduction for EM/circuit simulation

Let us divide the whole Model-Order Reduction (MOR) chain into three steps:

1. *Interconnect modeling*: model, using Electro-Magnetics (EM) simulation or other methods, the interconnect (e.g., layout parasitics) by a large RLC network.

2. *Linear MOR*: reduce the RLC network to obtain a reduced-order frequency-domain interconnect model (e.g., a set of poles and residues).
3. *Macromodel realization*: link the model obtained to transient simulation of the whole nonlinear circuit by generating an appropriate equivalent-circuit representation.

These three steps are treated in Sections 4.1, 4.2, and 4.3, respectively.

4.1 Interconnect modeling

Interconnects can be modeled using RC/RLC networks, (dispersive multi-conductor) transmission lines, (measurement or EM-simulation-based) tabulated frequency-domain scattering parameters, or even 3D full-wave models. The selection of a proper interconnect model depends on operation frequency, desired accuracy, available computational resources, etc.

In [Aal03], a dispersive inhomogenous two-conductor transmission line was treated in conjunction with the MOR method Padé-via-Lanczos (PVL). In [Pal04], in turn, a RLC lumped-element approximation for a dispersive multi-conductor transmission line was implemented in APLAC and treatment of tabulated frequency-domain scattering parameters was considered. Although we have studied interconnect modeling, it has not been the main focus area; in most cases, our starting point for MOR has been a given RLC netlist.

4.2 Linear MOR

During the last 15 years, various MOR methods have been proposed in the electrical-engineering literature. The first MOR methods were able to calculate single-input single-output transfer functions of linear circuits. The current MOR methods, in turn, are able to reduce large RLC networks such that the reduced-order models obtained can be consistently linked to the transient simulation of the whole nonlinear circuit (see Section 4.3).

In [Aal03], the following linear MOR methods were evaluated: Asymptotic Waveform Evaluation (AWE), Complex Frequency Hopping (CFH), Padé-via-Lanczos (PVL), reduction via split congruence transforms, coordinate-transformed Arnoldi algorithm, Passive Reduced-Order Interconnect Macromodeling Algorithm (PRIMA), and the PVL derivatives SyPVL, MPVL, and SyMPVL (see [Aal03] for all the references). According to [Aal03], "*Most of these methods were coded in a combination of C, MATLAB, and APLAC input language. Several test RLC networks were then reduced with the methods. In some cases, the result was a transfer function that was compared with that of the original circuit. In others, the result was a macromodel. Then, APLAC was used to run AC and transient analyses on both the original circuit and the reduced one. ... The best methods found were PRIMA and MPVL.*"

Based on this information, PRIMA [OCP98] was studied in more detail in [Pal04], where it was found that "*PRIMA provided passive reduced-order macromodels for interconnect circuits with excellent accuracy up to microwave frequencies. During this work, an attempt was made to develop a stopping criterion which would allow PRIMA iteration to be stopped right after numerical accuracy has been lost, thus allowing the easy generation of passive reduced-order models with the maximum*

available order. This attempt, however, proved futile: the instability of the reduced-order models could not be predicted from the properties of the matrices available during the iteration. In addition to this, an error estimate for PRIMA, presented in the literature, was evaluated. The results obtained with the error estimate were not always accurate enough, and the computation of the error estimate was too CPU-time consuming in some cases."
In [Aal03], the test RLC networks were quite small. While these small RLC networks were excellent in revealing the shortcomings of the reduction methods, they did not show all the potential of PRIMA (and MPVL). In [Pal04], in turn, much larger RLC networks (having nearly 1000 nodes) were reduced, and the overall impression of PRIMA was much more positive than in [Aal03], yet there were the problems mentioned above.
In [Pal04], PRIMA was implemented in APLAC, where it can be used as an off-line preprocessing tool for a large nonlinear circuit to be simulated. Namely, each large RLC block is treated as an N-port and reduced with PRIMA, the result being a file containing poles common for all the N-port Y-parameters, and the corresponding individual sets of residues for each Y-parameter.
Here, let us emphasize that PRIMA is by no means the last published linear MOR method; there are many newer methods that would be worth studying, too.

4.3 Macromodel realization

Since the reduced-order models are described in the frequency domain (or as differential equations), they have to be linked to the time-domain simulation of the total nonlinear circuit. This can be done by replacing the reduced-order models with appropriate macromodels.
Most of our scientific MOR activity has been just in this area: [Aal03], [Pal04] concentrate partially, and [AR02], [PR03], [PR04] fully on the macromodel realization. In particular, in [PR04] a comprehensive comparison of nine reduced-order interconnect macromodels for time-domain simulation is presented: the macromodels are reviewed, presented in a unified manner, and compared both theoretically and numerically.
The reduced-order macromodels can be divided into two groups:

- *Equivalent-circuit realizations*: a SPICE, APLAC, etc. netlist is synthesized using basic circuit elements. Nearly any time-domain circuit simulator can then be used.
- *Time-varying macromodels*: a macromodel with time explicitly present in the updating equations is generated. For most simulators, this method requires a modification of the simulator's source code.

In [PR04], we found the best macromodels for both categories. The overall fastest macromodel was the time-domain Differential-Equation Macromodel (DEM), which we proposed in [PR03]. In the course of the work of [Pal04], the time-domain DEM was implemented in APLAC. This time-varying macromodel is used as such in transient analysis. In other analysis methods (DC, AC, HB, etc.), another version of DEM is internally invoked; for example, in the case of AC analysis, $s = j\omega$ is inserted into relevant frequency-domain Y-parameter expressions.
The main message of [PR04] is that the macromodel realization has a great impact on the transient-simulation CPU time; in fact, the transient simulation of a poorly

realized macromodel (along with the nonlinear circuit) may last longer than that of the original, unreduced, circuit.

5 Behavioral modeling of components and circuit blocks

In Behavioral Modeling (BM), there is no (fast and accurate) model available, only the input-output data; therefore, BM corresponds to nonlinear system identification. The BM methods can be classified in many ways, e.g., as follows:

- analog ↔ digital
- static ↔ dynamic
- linear ↔ nonlinear
- white box ↔ black box
- single component ↔ circuit block
- measurement/simulation based BM [WR05] ↔ nonlinear MOR [Vos05]

Sections 5.1–5.4 (that are partly based on the ongoing Ph.D. Thesis work of Tuomo Kujanpää) will concentrate on Artificial Neural Network (ANN) based "analog, static, nonlinear, black-box-like" BM of a single component. Then, Section 5.5 will very briefly discuss our most recent research on Dynamic Neural Network (DNN) based "analog, dynamic, nonlinear, black-box-like, simulation-based" BM of a circuit block.

5.1 Motivation

The modeling of RF/microwave components for computer-aided design continuously faces new challenges because of increasing operation frequencies, circuit complexity, integration density, and decreasing time to market. It is often impossible to derive analytical models for new devices. Conventional numerical methods like 3D EM simulation are accurate but CPU-expensive. Empirical models, in turn, are fast but inaccurate over a wide operation range. Recently, it has been shown that ANNs offer benefits to urgent modeling needs; fast and accurate ANN models have been created for a wide range of components [ZG00].

Our goal has been to develop and implement an easy-to-use ANN-model generator for industrial model developers and circuit designers, who are neither ANN experts nor willing to switch between various modeling tools and simulators. In order to reach this goal, we first implemented an ANN-model generator prototype [RSP03] using the flexible input language of APLAC. Later on, we implemented `ANNModelGenerator` in APLAC using C language. Thus, the trained ANN models can be readily used in the same simulation framework.

5.2 Multi-layer perceptron ANNs

The most widely used ANN in RF/microwave component modeling is the Multi-Layer Perceptron MLP [ZG00] (yet we have also studied radial basis function ANNs [Poh03]). In our `ANNModelGenerator`, any number of MLP layers can be specified, but let us, for simplicity and due to the universal approximation theorem [Hay99], concentrate on a three-layer MLP that realizes the following nonlinear mapping:

$$\tilde{y}_l(\mathbf{x},\mathbf{w}) = w_{l0} + \sum_{j=1}^{N_\mathrm{h}} w_{lj} \tanh\left(w_{j0} + \sum_{i=1}^{N_\mathrm{i}} w_{ji} x_i \right), \quad l = 1,2,\ldots,N_\mathrm{o}, \tag{1}$$

where N_i, N_o, and N_h represent the number of inputs, outputs, and hidden-layer neurons, respectively; $\mathbf{x} = (x_1, x_2, \ldots, x_{N_\mathrm{i}})$, $\tilde{\mathbf{y}} = (\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_{N_\mathrm{o}})$, and $\mathbf{w} = (w_{10}, w_{11}, \ldots, w_{N_\mathrm{o}N_\mathrm{h}})$ represent ANN inputs, outputs, and weights, respectively. Let $\mathbf{y} = \mathbf{y}(\mathbf{x})$ be an unknown, nonlinear, multi-dimensional function to be approximated by the MLP mapping (1), that is, $\tilde{\mathbf{y}} = \tilde{\mathbf{y}}(\mathbf{x},\mathbf{w})$. Let $\{(\mathbf{x}^k, \mathbf{y}^k), k = 1,2,\ldots,N_\mathrm{tr}\}$ be an appropriate training set, N_tr being the number of samples, and the training-set inputs and outputs being scaled in the range $[-1,1]$. Furthermore, let us define the normalized ANN training error as [ZG00]

$$E_\mathrm{tr}(\mathbf{w}) = \sqrt{\frac{1}{N_\mathrm{tr} N_\mathrm{o}} \sum_{k=1}^{N_\mathrm{tr}} \sum_{l=1}^{N_\mathrm{o}} \left(\frac{\tilde{y}_l(\mathbf{x}^k,\mathbf{w}) - y_l^k}{2} \right)^2}. \tag{2}$$

The training of the ANN means minimizing of $E_\mathrm{tr}(\mathbf{w})$ with respect to weights, $\mathbf{w}$, by optimization. The generalization capability of the trained ANN is evaluated by applying (2) to an independent test set, $\{(\mathbf{x}^k, \mathbf{y}^k),\ k = 1,2,\ldots,N_\mathrm{te}\}$.

5.3 ANN-model generation

Overview

A block diagram of ANN-model generation (`ANNModelGenerator`) and ANN-model usage (`MODEL_FILE`, `ANNModel`, and `ANNFunc`) is shown in Fig. 1. The obligatory and optional blocks are drawn with solid and dashed lines, respectively. The operation of (most of) these blocks is explained in the following five subsections.

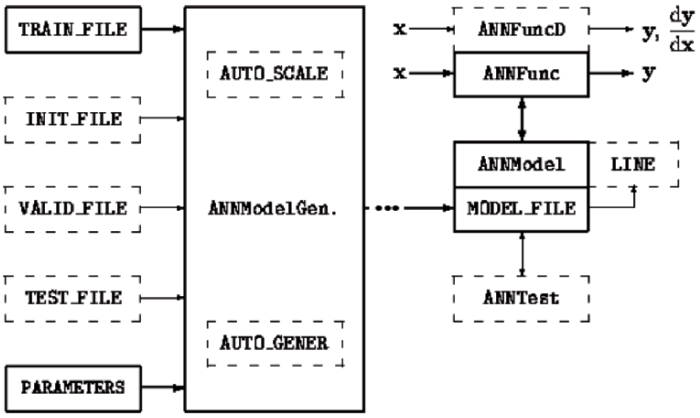

Fig. 1: ANN-model generation and usage.

ANN-structure selection

One challenge in ANN-based modeling is the determination of the number of hidden layers and their neurons; a very simple MLP with few weights may not offer enough degrees of freedom for the approximation problem, while a complex MLP structure results in many weights to be optimized. In `ANNModelGenerator`, the default number of hidden layers is one, which should be enough in many cases. The (optimistic) default number of neurons in that hidden layer is $N_\mathrm{h} = N_\mathrm{i} + N_\mathrm{o}$. Naturally, the user can increase the number of layers and/or neurons (by adjusting '`PARAMETERS`', See Fig. 1), if needed.

Training/validation/test-set generation

The very first step in ANN training is to generate a training-set file. The training-set file (obligatory `ANNModelGenerator` input `TRAIN_FILE` in Fig. 1) includes a collection of data samples, each consisting of relevant inputs (e.g., MOSFET DC-bias voltages V_gs and V_ds) and desired outputs (e.g., MOSFET drain current $I_\mathrm{ds}(V_\mathrm{gs}, V_\mathrm{ds})$), obtained from measurements or simulations.
If ANN training continues for too long, overlearning, or oscillatory overfitting to (noisy) training data, may occur [Hay99]. Therefore, an independent validation set may be used to avoid overlearning by early stopping [Hay99] at the lowest validation error obtained. Moreover, the generalization ability of the ANN should be tested with an independent test set after ANN training. The optional validation-set file (`VALID_FILE`) and test-set file (`TEST_FILE`) should be constructed such that all their inputs, $\mathbf{x}^k = (x_1^k, x_2^k, \ldots, x_{N_\mathrm{i}}^k)$, $k = 1, 2, \ldots, \{N_\mathrm{va}, N_\mathrm{te}\}$, are located inside the region defined by the training-set inputs; otherwise, this data will be also used for validating/testing the (non-guaranteed) extrapolation capability of the ANN.

Data scaling

In the literature, many heuristic methods have been suggested for improving ANN training [Hay99]. One of these methods is scaling: since in typical RF/microwave modeling applications the orders of magnitude of input/output parameter values are very different from one another, scaling of training data is desirable for ANN training [ZG00].
In `ANNModelGenerator`, each ANN input and output, that is, each column of the training (and validation/test) set, is scaled in the range $[-1, 1]$ before the actual ANN training. Linear scaling is used by default, but there is also another option, namely automatic scaling (`AUTO_SCALE`) that is based on our work in [RP04]. The automatic scaling first finds a suitable logarithmic scaling function [ZG00] for each column, after which it optimizes the shape parameter of the function such that the scaled values are spread as equally as possible in the range $[-1, 1]$.

ANN training

APLAC contains 10 optimization methods:

- Global methods:
  - Genetic algorithm

  - Simulated annealing
- Gradient-based methods:
  - Steepest descent
  - Conjugate gradient
- Direct-search methods:
  - Hooke–Jeeves
  - Nelder–Mead
  - Multi-directional search
- Other methods:
  - MinMax
  - Random
  - Exhaustive search

All these methods can be used to optimize virtually any (circuit) variable with respect to any design goal. Thanks to the internal '`ANNModelGenerator` $\leftrightarrow$ optimization methods' C-code interface, all these 10 methods are readily available for ANN training, too. Until now, we have only modified the gradient-based methods for ANN training by adding the Error Back Propagation (EBP) algorithm [Hay99] for fast and accurate gradient evaluation. In `ANNModelGenerator`, the default optimization method is conjugate gradient with EBP and Hestenes–Stiefel search-direction determination [KRH05].
One important factor in ANN training is weight initialization [Hay99]. Currently, we initialize the weights randomly in the range $[-0.25, 0.25]$; this simple scheme will be improved in the future [KR06].
`ANNModelGenerator` calculates the normalized training error, E_{tr}, from (2). If, say, $E_{\mathrm{tr}} \leq 0.5\ \%$, ANN training was succesful.

ANN validation/testing

If `VALID_FILE` is specified, `ANNModelGenerator` calculates the normalized validation error, E_{va}, by applying (2) at every 10th optimization cycle. If E_{va} is smaller than the previous one, the current ANN model (only) is saved (the memory requirement being mainly determined by the, say, $50 \ldots 500$ ANN weights, $w_{10}, w_{11}, \ldots, w_{N_{\mathrm{o}}N_{\mathrm{h}}}$). When ANN training terminates (e.g., at the maximum number of optimization cycles), the best ANN model, with the lowest E_{va} at the early-stopping point, is fetched. If `TEST_FILE` has been specified, `ANNModelGenerator` calculates, at the end of ANN training, the normalized test error, E_{te}. If, say, $E_{\mathrm{te}} \leq 1\ \%$, the generalization capability of the ANN is very good.

5.4 Connection to circuit simulation

The end result of ANN training is the ANN-model file (`MODEL_FILE`) that contains, e.g., the ANN structure, the values of ANN weights, and relevant comment lines for ANN training/validation/test error, ANN-training CPU-time, etc.
The trained ANN model can be connected to APLAC circuit simulation by `ANNModel` (see Fig. 1), which reads in `MODEL_FILE` and stores the parameters. The actual on-line calculation of ANN outputs (during circuit simulation) is done using ANN-evaluation functions `ANNFunc` and `ANNFuncD`, which use the parameters stored by `ANNModel`. In nonlinear modeling applications (e.g., MOSFET drain current

$I_{\mathrm{ds}}(V_{\mathrm{gs}}, V_{\mathrm{ds}})$), the use of `ANNFuncD` is recommended, since it also returns the analytically calculated derivatives [AO00] (e.g., $\partial I_{\mathrm{ds}}/\partial V_{\mathrm{gs}}$ and $\partial I_{\mathrm{ds}}/\partial V_{\mathrm{ds}}$) that are needed in circuit simulation for the Newton–Raphson iteration method.
To summarize, in our approach both ANN-model generation and usage can be seamlessly done inside the same circuit-simulation framework: APLAC.

5.5 Dynamic behavioral modeling using DNNs

Very recently, we have also started to study behavioral modeling of dynamic nonlinear circuit blocks like Power Amplifiers (PAs) using Dynamic Neural Networks (DNNs) [WR05] (Ch. 6 and 7), [Mei96], [PAS01]. Untill now, we have developed and implemented a prototype version of `DNNModelGenerator` using a combination of `ANNModelGenerator` and APLAC input language. We have tested this tool by generating a DNN model for an audio amplifier and for a 5 GHz PA. These DNN models could be used to replace the original circuit block in a HB simulation.

6 Other research activities

Finally, it is worth mentioning that during the years 2002–2006, we have also carried out research outside the ARFSIM, MOSAICS, and AMAZE projects. In [Kuj02], digital components were modeled (for mixed-mode simulation with the in-house development version of APLAC). In [Sun04], APLAC simulation models for striplines with conductor surface roughness were generated. In [Vei06], in turn, several APLAC simulation models for Micro-Electro-Mechanical Systems (MEMS) are documented and the related MEMS publications of TKK CTL are listed. Finally, it is worth mentioning (outside the topic 'circuit simulation') that [Sil06] contains a list of publications on network analyzer calibration methods.

References

[Roo04] Roos, J., Aaltonen, S., Honkala, M., Karanko, V., Kujanpää, T., Lehtovuori, A., Palenius, T., Pohjala, A., Valtonen, M., Virtanen, J.: Advanced radio frequency simulation and modeling of electronic circuits. ECMI Newsletter, 10–11 (2004)

[Kal02] Kallio, A.: MESFET models in the APLAC circuit simulator. M.Sc. Thesis, Helsinki University of Technology, Finland (2002)

[KV04] Kallio, A., Valtonen, M.: A new rule for MESFET gate charge division based on the energy conservation principle. Int. Journal of Circ. Theory and Appl., **32**, 139–165 (2004)

[RVV02] Roos, J., Valtonen, M., Virtanen, J.: Implementation of piecewise-linear DC analysis in APLAC. In: Proc. ICECS 2002, **3**, 1139–1142 (2002)

[Roo05] Roos, J.: On simplex-based piecewise-linear approximations of nonlinear mappings. Int. Journal of Circ. Theory and Appl., **33**, 109–134 (2005)

[Roo06] Roos, J.: Speed-up and performance evaluation of piecewise-linear DC analysis. Int. Journal of Circ. Theory and Appl., accepted to be published (2006)

[Hon02a] Honkala, M.: Nonmonotone norm-reduction method for circuit simulation. IEE Electronics Letters, **38**, 22, 1316–1317 (2002)

[Hon02b] Honkala, M.: Nonmonotone norm-reduction method in numerical circuit analysis. In: TKK CTL report series, **46** (2002)

[HRK06] Honkala, M., Roos, J., Karanko, V.: On nonlinear iteration methods for DC analysis of industrial circuits. In: Di Bucchianico, A., Mattheij, R.M.M., Peletier, M.A. (ed) Mathematics in Industry 8 — Progress in Industrial Mathematics at ECMI 2004. Springer, Berlin Heidelberg. 144–148 (2006)

[Lin06] Linja-aho, V.: Homotopy methods in circuit DC analysis. M.Sc. Thesis, Helsinki University of Technology, Finland (2006)

[Hon02c] Honkala, M.: Parallel hierarchical DC analysis. Lic.Sc. Thesis, Helsinki University of Technology, Finland (2002)

[HK02] Honkala, M., Karanko, V.: Improving the convergence of combined Newton–Raphson and Gauss–Newton multilevel iteration method. In: Proc. ISCAS 2002, **2**, 229–232 (2002)

[KH02] Karanko, V., Honkala, M.: Least squares solution of nearly square overdetermined sparse linear systems. In: Proc. ISCAS 2002, **4**, 830–833 (2002)

[Hon03] Honkala, M.: Parallel Processing in APLAC — Part A: DC and Transient Analyses. In: TKK CTL report series, **47** (2003)

[Vir05] Virtanen, J.: Harmonic balance and phase-noise analysis methods in the APLAC circuit simulator. Lic.Sc. Thesis, Helsinki University of Technology, Finland (2005)

[KH04] Karanko, V., Honkala, M.: A parallel harmonic balance simulator for shared memory multicomputers. In: Proc. EUMW 2004 (2004)

[Poh06] Pohjala, A.: Harmonic balance analysis for frequency-divider circuit (in Finish). Lic.Sc. Thesis, Helsinki University of Technology, Finland (2006)

[LVV03] Lehtovuori, A., Virtanen, J., Valtonen, M.: GMRES preconditioners for multivariate steady-state time-domain method. In: Proc. IMS 2003, 2129–2132 (2003)

[Leh03] Lehtovuori, A.: Multivariate steady-state time-domain analysis method. Lic.Sc. Thesis, Helsinki University of Technology, Finland (2003)

[Cos05] Costa, L.: Incorporating lumped elements into an efficiently implemented FDTD-based electromagnetic simulator. Lic.Sc. Thesis, Helsinki University of Technology, Finland (2005)

[Cos06] Costa, L.: Implementing efficient array traversing for FDTD-lumped element cosimulation. In: Di Bucchianico, A., Mattheij, R.M.M., Peletier, M.A. (ed) Mathematics in Industry 8 — Progress in Industrial Mathematics at ECMI 2004. Springer, Berlin Heidelberg. 149–153 (2006)

[Aal03] Aaltonen, S.: Order reduction of interconnect circuits. Lic.Sc. Thesis, Helsinki University of Technology, Finland (2003)

[Pal04] Palenius, T.: Efficient time-domain simulation of interconnects characterized by large RLC circuits or tabulated S parameters. Lic.Sc. Thesis, Helsinki University of Technology, Finland (2004)

[AR02] Aaltonen, S., Roos, J.: Simple reduced-order macromodels with PRIMA. In: Proc. ICECS 2002, **1**, 367–360 (2002)

[PR03] Palenius, T., Roos, J.: An efficient reduced-order interconnect macromodel for time-domain simulation. In: Proc. ISCAS 2003, **4**, 628–631 (2003)

[PR04] Palenius, T., Roos, J.: Comparison of reduced-order interconnect macromodels for time-domain simulation. IEEE Trans. Microwave Theory and Techniques, **52**, 9, 2240–2250 (2004)

[RSP03] Roos, J., Şengör, N.S., Pohjala, A.: Artificial neural network based RF-model generator — version 0.2. In: TKK CTL report series, **48** (2003)

[Poh03] Pohjala, A.: Generation of simulation models from measurement data using interpolation and radial basis function networks. M.Sc. Thesis, Helsinki University of Technology, Finland (2003)

[RP04] Roos, J., Pohjala, A.: Development and comparison of formulas for scaling ANN inputs and outputs in RF-modeling applications. In: Buikis, A., Ciegis, R., Fitt, A.D. (ed) Mathematics in Industry 5 — Progress in Industrial Mathematics at ECMI 2002. Springer, Berlin Heidelberg. 197–201 (2004)

[KRH05] Kujanpää, T., Roos, J., Honkala, M.: Experimental comparison of optimization methods in ANN training. In: Proc. PRIME 2005, **2**, 430–433 (2005)

[KR06] Kujanpää, T., Roos, J.: Efficient initialization of artificial neural network weights for electrical component models. In: Book of Abstracts of SCEE 2006, 47–48 (2006)

[Kuj02] Kujanpää, T.: Modeling of digital components in the APLAC circuit simulator. M.Sc. Thesis, Helsinki University of Technology, Finland (2002)

[Sun04] Sundström, S.: Stripline simulation models with conductor surface roughness, M.Sc. Thesis, Helsinki University of Technology, Finland (2004)

[Vei06] Veijola, T., *et. al.*: Modeling of micromechanical devices (2006) `http://www.ct.tkk.fi/research/mems/main.html`

[Sil06] Silvonen, K.: Network analyzer calibration methods (2006) `http://www.ct.tkk.fi/research/calibration.html`

[A06] APLAC 8.2 Manuals. AWR–APLAC Corporation, Finland (2006) `http://www.aplac.com`

[OCP98] Odabasioglu, A., Celik, M., Pileggi, L.T.: PRIMA: passive reduced-order interconnect macromodeling algorithm. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, **17**, 88, 645–654 (1998)

[WR05] Wood, J., Root, D.E. (ed): Fundamentals of Nonlinear Behavioral Modeling for RF and Microwave Design. Artech House, Boston London (2005)

[Vos05] Voss, T.: Model reduction for nonlinear differential algebraic equations. M.Sc. Thesis (Philips Nat.Lab. Unclassified Report PR-TN-2005/00919), University of Wuppertal, Germany (2005)

[ZG00] Zhang, Q.J., Gupta, K.C.: Neural Networks for RF and Microwave Design. Artech House, Boston London (2000)

[Hay99] Haykin, S.: Neural Networks — A Comprehensive Foundation. Prentice Hall, New Jersey (1999)

[AO00] Antonini, G., Orlandi, A.: Gradient evaluation for neural-networks-based electromagnetic optimization procedures. IEEE Trans. Microwave Theory and Techniques, **48**, 5, 874–876 (2000)

[Mei96] Meijer, P.B.L.: Neural network applications in device and circuit modelling for circuit simulation, Ph.D. Thesis, Eindhoven University of Technology, The Netherlands (1996)

[PAS01] Plebe, A., Anile, A.M., Rinaudo, S.: Sub-micrometer bipolar transistor modeling using neural networks. In: van Riene, U., Günther, M., Hecht, D. (ed): Scientific Computing in Electrical Engineering, Lect. Notes in Comput. Science and Engineering, **18**, Springer, Berlin Heidelberg. 259–266 (2001)