Mobile communication is rapidly growing. Increasing demands on capacity and bandwidth have to be addressed by future developments. This means higher signal requirements and bandwidth for transceivers in mobile basestations. Transceivers are the component with highest power consumption in a basestation. Especially analog components show different impairments and nonideal behavior with negative effects on energy efficiency and signal integrity. These effects can be analyzed and mathematically described to build a specific digital signal processing algorithm, which mitigates certain effects. This work treats impairments from machine learning perspective. IQ imbalance of modulators as well as power amplifier nonlinearities are representative impairments with significant influence on the signal quality. These effects are trained to artificial neural networks (ANNs) for digital impairment mitigation. Furthermore it is shown that the ANNs are able to model different impairment effects with a single network and can be simply enhanced by further input parameters to mitigate dynamic effects. Physically inspired modeling of long term memory effects like thermal memory and charge trapping are a special focus of this work.
Patrick Jüschke

Physically Inspired Predistortion of RF Power Amplifiers with Artificial Neural Networks
FAU Studien aus der Elektrotechnik

Band 21

Herausgeber der Reihe:
Prof. Dr.-Ing. Bernhard Schmauß
Patrick Jüschke

Physically Inspired Predistortion of RF Power Amplifiers with Artificial Neural Networks

Erlangen
FAU University Press
2023
Physically Inspired Predistortion of RF Power Amplifiers with Artificial Neural Networks

Physikalisch inspirierte Vorverzerrung von Hochfrequenzleistungsverstärkern mit künstlichen neuronalen Netzen

Der Technischen Fakultät
der Friedrich-Alexander-Universität
Erlangen-Nürnberg

zur
Erlangung des Doktorgrades Dr.-Ing.

vorgelegt von

Patrick Jueschke

aus Ludwigsburg
Als Dissertation genehmigt
von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen
Prüfung: 15.11.2022

Vorsitzender des
Promotionsorgans: Prof. Dr.-Ing. Knut Graichen

Gutachter: Prof. Dr.-Ing. Georg Fischer
Prof. Dr.-Ing. Matthias Rudolph
Author’s Statement

Parts of this work, mainly measurement and test results have been previously published in four publications. All of the publications have the same main author as this work and are co-authored by the supervising professor who gave feedback on conducted work and written papers. Andreas Pascht was the department head during this time and supervisor of the work within the RF transceivers and amplifiers department at Bell Labs.

Parts of Sections 3.3, 5.4, 5.5, 5.6, 5.7 and I have been published in following publications.


3. P. Jueschke, G. Fischer, Neural Network Based Linearization of RF Power Amplifiers using In-Situ Device Temperature Measurement, IEEE International Conference on Ubiquitous Wireless Broadband (ICUWB), 2015

4. P. Jueschke, J. Brendel, G. Fischer, A. Pascht, Feed-Forward Neural Networks for Analog Impairment Mitigation in High Power RF Transceivers, IEEE Asia Pacific Microwave Conference (APMC), 2011

Johannes Brendel is a co-author of publication (4). He worked within an internship of three months on the software implementation of the predistortion experiments. A complete list of author’s publications is provided in the appendix.
Abstract

Artificial intelligence (AI) is a barely defined phrase which enjoys high interest these days. Technical systems which are neither rule based nor structurally programmed are usually seen behind AI. Abstract models which are biologically inspired and trained to fit on certain problems build the platform for machine learning (ML) and AI. Artificial neural network (ANN) are specific models for ML.

Mobile communication is rapidly growing. Increasing demands on capacity and bandwidth have to be addressed by future developments. This means higher signal requirements and bandwidth for transceiver (TRX) in mobile basestations. Beside this fact, TRXs are the component with highest power consumption in a basestation. Widely used TRX architectures consist of a digital frontend (DFE) for signal processing, A/D and D/A conversion, power amplifiers and band filters while algorithms in the DFE are used to preprocess the baseband signal and to mitigate impairments which leads to a higher energy efficiency of the system.

Especially analog components show different impairments and nonideal behavior with negative effects on energy efficiency and signal integrity. These effects can be analyzed and mathematically described to build a specific digital signal processing (DSP) algorithm, which mitigates certain effects. This work treats impairments from ML perspective. IQ Imbalance of modulators as well as power amplifier (PA) nonlinearities are representative impairments with significant influence on the signal quality. These effects are trained to ANNs for digital impairment mitigation. Simulations and measurements with real hardware show how it works up to neurocomputing of ANNs in field programmable gate arrays (FPGAs). All ANNs are based on the same neurons (perceptrons) with different transfer functions and use the same training algorithm. Furthermore it is shown that the ANNs are able to model different impairment effects with a single network and can be simply enhanced by further input parameters to mitigate dynamic effects. Physically inspired modeling of long term memory effects (LTM) like thermal memory and charge trapping with ANNs are a special focus of this work.
Zusammenfassung


Mobilfunk ist ein schnell wachsender Bereich der Kommunikation. Um den Kapazitätsanforderungen in Zukunft gerecht zu werden, muss die Technologie ständig weiterentwickelt werden. Für die Transceiver in Mobilfunkbasisstationen bedeutet das vor allem höhere Signalanforderungen und mehr Bandbreite. Gleichzeitig verbrauchen die Transceiver in Basisstationen die meiste Energie. Um die Energieeffizienz zu steigern und die Signalqualität zu verbessern bestehen heutige Systeme aus einer digitalen Signalverarbeitung und analogen Komponenten wie Verstärkerkette und Filtern.

Acknowledgments

I would like to express my sincere gratitude to my advisor Prof. Dr.-Ing. Georg Fischer for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing this thesis. I can't imagine to have a better advisor and mentor for my Ph.D study.

I thank my fellow colleagues Thomas Bohn, Dr. Christoph Haslach, Dirk Wiegner, Gerhard Luz, Dr. Andreas Pascht, Dr. Wolfgang Templ, Dr. Daniel Markert, Dieter Ferling, Heinz Schlesinger, Dr. Xin Yu, Dr. George-Roberto Hotopan, Carsten Haase, Silvia Krönert and Dr. Junqing Guan for the stimulating discussions and all the fun we had in the last years during the studies. Especially my supervisors Dr. Andreas Pascht and Dr. Wolfgang Templ gave me a lot of motivation and I’m thankful for their patience.

All the best to my closest friends, Dennis Jöst, Florian Schwarz, Nils Larcher, Sandra Maier, Laura Lilienthal and Jonathan Krebs (JK). Dennis gave me the initial inspiration for this research and has always been a great fellow with his outstanding experience in machine learning and AI. I’m thankful to Florian, Nils and Sandra for all the motivation and support especially in hard and difficult times. Further I want to thank Laura and JK for their rewinding impulses and great patience in the hot phase of this research.

Special thanks to Fraunhofer IAF in Freiburg for their great semiconductors and power amplifiers. Further to mention is the Bell Labs Modelshop in Stuttgart for their great job in mechanics and electronics for demonstrator and test platform development.

Last but not least, I can’t find the right words to express my gratitude, love and respect to my family. They have always been my inspiration and give me the strength for any accomplishment. Special thanks to my aunt Silvia Staiber for her great support at my defence party.

Parts of this research work have received funding within IntelliSpektrum, a German federal funded project of Federal Ministry for Economic Affairs and Climate Action (BMWi) in IT2Green program.
Contents

Author’s Statement .......................................................... iii
List of Symbols and Abbreviations ................................. xv

1 Introduction ............................................................... 1

2 Tools for Performance Assessment ............................... 5
  2.1 Simulation Environment ........................................... 5
  2.1.1 Circuit Envelope Simulation ............................... 6
  2.2 Hardware Test Platform ........................................... 8
  2.2.1 Class-AB Power Amplifier .................................. 10
  2.2.2 Class-ABJ Power Amplifier .................................. 10

3 Power Amplifier Modeling ........................................... 13
  3.1 Static Nonlinearities ............................................. 14
  3.1.1 Polynomial Model ............................................ 16
  3.1.2 Saleh’s Model .................................................. 16
  3.2 Memory Effects .................................................. 17
  3.2.1 Short Term Memory Effects ................................. 17
  3.2.2 Long Term Memory Effects ................................. 21
  3.3 Thermal Memory .................................................. 22
  3.3.1 Performance of Temperature Sensors ..................... 23
  3.3.2 Thermal Time Constant Evaluation ......................... 24
  3.3.3 Characterization of Thermal Memory ....................... 24
  3.3.4 Modeling Thermal Memory on Measurement Data ........... 25
  3.4 Charge Trapping in GaN Devices ......................... 28
  3.4.1 Measuring Intrinsic Voltages .............................. 29
  3.4.2 Peak Voltage Detector ..................................... 29

4 Predistortion Techniques ........................................... 33
  4.1 Feedback Linearization ......................................... 34
  4.2 Feedforward Linearization .................................... 34
  4.3 Predistortion .................................................... 35
  4.3.1 Direct and Indirect Learning .............................. 38
  4.4 Digital Predistortion (DPD) ................................. 38
  4.4.1 Memoryless DPD ............................................ 39
  4.4.2 Mitigation of Memory Effects ............................. 39
## 5 Artificial Neural Networks for Predistortion of RF Power Amplifiers

5.1 Neuron Model

5.1.1 Initialization of Neuron Weights

5.1.2 Transfer Functions

5.2 Network Architectures

5.2.1 Time Delay Lines for Memory Effect Mitigation

5.3 Training

5.3.1 Learning Rules

5.3.2 Advanced Training Algorithms

5.3.3 Assessment on Training Performance

5.4 IQ Imbalance Mitigation

5.4.1 Neural Network Design

5.4.2 Simulation Results

5.4.3 Measurement Results

5.5 Power Amplifier Linearization

5.5.1 Neural Network Design

5.5.2 Linearization of Saleh Model

5.5.3 Linearization of Polynomial Model

5.5.4 Linearization of Memory Polynomial Model

5.5.5 Measurement Results

5.6 IQ Imbalance Mitigation and PA Linearization

5.6.1 Neural Network Design

5.6.2 Simulation Results

5.6.3 Measurement Results

5.7 Linearization of Thermal Memory

5.7.1 Neural Network Design

5.7.2 Results on Linearization of Thermal Memory

5.7.3 Linearization Performance for TDD

5.8 Linearization of Thermal Memory and IQ Imbalance

5.8.1 Neural Network Design

5.8.2 Results on Thermal Memory and IQ Imbalance

5.9 Linearization of Charge Trapping Memory

5.9.1 Neural Network Design

5.9.2 Results on Linearization of Charge Trapping

5.10 Comparison of Results

5.10.1 Simulation Results

5.10.2 Measurement Results

5.10.3 Comparison with Volterra DPD

5.10.4 Comparison with Memory Polynomial DPD
6 FPGA Neurocomputing ................................................. 91
   6.1 System Design .................................................. 91
   6.2 Implementation of Transfer Functions ......................... 92
      6.2.1 Piecewise Linear Approximation of Sigmoid Function .... 93
      6.2.2 Assessment on Approximation Error ....................... 94
   6.3 Hardware Neural Network ...................................... 94
      6.3.1 Absolute Value Calculation ................................. 96
      6.3.2 HWNN Layers ............................................... 96
      6.3.3 Digital Design of Neurons ................................. 97
      6.3.4 Parallel Computation ..................................... 97
   6.4 Co-Processing .................................................. 98
      6.4.1 Software Neural Network .................................. 98
      6.4.2 Training Algorithm ....................................... 99
      6.4.3 Performance Monitor ...................................... 99
      6.4.4 Parameter Update Logic .................................. 99

7 Conclusion .......................................................... 101

Bibliography .......................................................... 105

Appendix .............................................................. 111
   A List of Publications by the Author .............................. 111
   B Levenberg-Marquardt Training .................................. 113
   C Piecewise Cubic Interpolation Algorithm ...................... 119
   D Error Vector Magnitude Estimation ............................. 120
   E ACP Calculations ................................................ 121
   F Circuit Envelope Simulation Subcircuits ....................... 122
   G Characterization Results of Class-AB PA ..................... 124
   H Characterization Results of Class-ABJ PA .................... 126
   I Measurement Results with 3G UMTS Signal .................... 127
List of Figures ....................................................... 128
List of Tables ....................................................... 132
List of Symbols and Abbreviations

List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACLR</td>
<td>adjacent channel leakage ratio</td>
</tr>
<tr>
<td>ACP</td>
<td>adjacent channel power</td>
</tr>
<tr>
<td>ADC</td>
<td>analog to digital converter</td>
</tr>
<tr>
<td>ADS</td>
<td>Advanced Design System</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial intelligence</td>
</tr>
<tr>
<td>AM-AM</td>
<td>input amplitude to output amplitude distortion</td>
</tr>
<tr>
<td>AM-PM</td>
<td>input amplitude to output phase distortion</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial neural network</td>
</tr>
<tr>
<td>BW</td>
<td>bandwidth</td>
</tr>
<tr>
<td>CORDIC</td>
<td>coordinate rotation digital computer</td>
</tr>
<tr>
<td>CTM</td>
<td>charge trapping model</td>
</tr>
<tr>
<td>CW</td>
<td>continuous wave</td>
</tr>
<tr>
<td>DAC</td>
<td>digital to analog converter</td>
</tr>
<tr>
<td>DC</td>
<td>direct current</td>
</tr>
<tr>
<td>DFE</td>
<td>digital frontend</td>
</tr>
<tr>
<td>DL</td>
<td>downlink</td>
</tr>
<tr>
<td>DPD</td>
<td>digital predistortion</td>
</tr>
<tr>
<td>DSP</td>
<td>digital signal processing</td>
</tr>
<tr>
<td>EBP</td>
<td>Error back-propagation</td>
</tr>
<tr>
<td>EPA</td>
<td>error power amplifier</td>
</tr>
<tr>
<td>EVM</td>
<td>error vector magnitude</td>
</tr>
<tr>
<td>FB</td>
<td>feedback</td>
</tr>
<tr>
<td>FD</td>
<td>frequency domain</td>
</tr>
<tr>
<td>FFNN</td>
<td>feed forward neural network</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>FPGA</td>
<td>field programmable gate array</td>
</tr>
<tr>
<td>FWA</td>
<td>fixed wireless access</td>
</tr>
<tr>
<td>GaN</td>
<td>Galliumnitride</td>
</tr>
<tr>
<td>HEMT</td>
<td>high electron mobility transistor</td>
</tr>
<tr>
<td>HTP</td>
<td>hardware test platform</td>
</tr>
<tr>
<td>HWNN</td>
<td>hardware neural network</td>
</tr>
<tr>
<td>IDE</td>
<td>integrated development environment</td>
</tr>
<tr>
<td>IF</td>
<td>intermediate frequency</td>
</tr>
<tr>
<td>IMD</td>
<td>intermodulation distortions</td>
</tr>
<tr>
<td>IMN</td>
<td>input matching network</td>
</tr>
<tr>
<td>LMT</td>
<td>Levenberg-Marquardt training</td>
</tr>
<tr>
<td>LO</td>
<td>local oscillator</td>
</tr>
<tr>
<td>LTE</td>
<td>Long Term Evolution</td>
</tr>
<tr>
<td>LTI</td>
<td>linear time invariant</td>
</tr>
<tr>
<td>LTM</td>
<td>long term memory effects</td>
</tr>
<tr>
<td>LUT</td>
<td>look-up table</td>
</tr>
<tr>
<td>MIMO</td>
<td>multiple-input multiple-output</td>
</tr>
<tr>
<td>ML</td>
<td>machine learning</td>
</tr>
<tr>
<td>MPM</td>
<td>memory polynomial model</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean squared error</td>
</tr>
<tr>
<td>NLS</td>
<td>nonlinear system</td>
</tr>
<tr>
<td>NLTV</td>
<td>nonlinear time-variant</td>
</tr>
<tr>
<td>OMN</td>
<td>output matching network</td>
</tr>
<tr>
<td>OOM</td>
<td>object oriented model</td>
</tr>
<tr>
<td>PA</td>
<td>power amplifier</td>
</tr>
<tr>
<td>PAE</td>
<td>power-added efficiency</td>
</tr>
<tr>
<td>PAPR</td>
<td>peak-to-average power ratio</td>
</tr>
<tr>
<td>PC</td>
<td>personal computer</td>
</tr>
<tr>
<td>PCHIP</td>
<td>piecewise cubic Hermite interpolation polynomial</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>PM</td>
<td>phase modulation</td>
</tr>
<tr>
<td>PUL</td>
<td>parameter update logic</td>
</tr>
<tr>
<td>RBW</td>
<td>resolution bandwidth</td>
</tr>
<tr>
<td>ReLU</td>
<td>rectified linear unit</td>
</tr>
<tr>
<td>RF</td>
<td>radio frequency</td>
</tr>
<tr>
<td>RNN</td>
<td>recurrent neural network</td>
</tr>
<tr>
<td>ROM</td>
<td>read only memory</td>
</tr>
<tr>
<td>RTOS</td>
<td>real time operating system</td>
</tr>
<tr>
<td>SISO</td>
<td>single-input single-output</td>
</tr>
<tr>
<td>SNR</td>
<td>signal to noise ratio</td>
</tr>
<tr>
<td>SOM</td>
<td>self organizing map</td>
</tr>
<tr>
<td>SR</td>
<td>sampling-rate</td>
</tr>
<tr>
<td>STM</td>
<td>short term memory effects</td>
</tr>
<tr>
<td>SWNN</td>
<td>software neural network</td>
</tr>
<tr>
<td>TA</td>
<td>training algorithm</td>
</tr>
<tr>
<td>TD</td>
<td>time domain</td>
</tr>
<tr>
<td>TDD</td>
<td>Time division duplex</td>
</tr>
<tr>
<td>TDL</td>
<td>Time delay line</td>
</tr>
<tr>
<td>TRX</td>
<td>transceiver</td>
</tr>
<tr>
<td>TWTA</td>
<td>travelling wave tube amplifier</td>
</tr>
<tr>
<td>UL</td>
<td>uplink</td>
</tr>
<tr>
<td>UML</td>
<td>unified modeling language</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunications System</td>
</tr>
<tr>
<td>VBW</td>
<td>video bandwidth</td>
</tr>
<tr>
<td>VHDL</td>
<td>very high speed integrated circuit hardware description language</td>
</tr>
</tbody>
</table>
### List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_{in}(t)$</td>
<td>V</td>
<td>continuous input voltage</td>
</tr>
<tr>
<td>$v_{out}(t)$</td>
<td>V</td>
<td>continuous output voltage</td>
</tr>
<tr>
<td>$v(t)$</td>
<td>V</td>
<td>continuous voltage</td>
</tr>
<tr>
<td>$t$</td>
<td>s</td>
<td>continuous time</td>
</tr>
<tr>
<td>$V_{max}$</td>
<td>V</td>
<td>peak or maximum voltage</td>
</tr>
<tr>
<td>$x[n]$</td>
<td></td>
<td>complex-valued discrete input signal</td>
</tr>
<tr>
<td>$y[n]$</td>
<td></td>
<td>complex-valued discrete output signal</td>
</tr>
<tr>
<td>$\varphi[n]$</td>
<td>$^\circ$</td>
<td>phase of discrete input signal</td>
</tr>
<tr>
<td>$A[n]$</td>
<td></td>
<td>magnitude of discrete input signal</td>
</tr>
<tr>
<td>$a$</td>
<td></td>
<td>scalar coefficient factor</td>
</tr>
<tr>
<td>$\alpha_k$</td>
<td></td>
<td>complex-valued coefficient factor</td>
</tr>
<tr>
<td>$F_{AM}$</td>
<td></td>
<td>AM-AM function</td>
</tr>
<tr>
<td>$F_{PM}$</td>
<td></td>
<td>AM-PM function</td>
</tr>
<tr>
<td>$\gamma_q$</td>
<td></td>
<td>impulse response of IMN as LTI system</td>
</tr>
<tr>
<td>$\gamma_p$</td>
<td></td>
<td>impulse response of OMN as LTI system</td>
</tr>
<tr>
<td>$h[n]$</td>
<td></td>
<td>transfer function of matching network as LTI system</td>
</tr>
<tr>
<td>$l_q, l_p$</td>
<td></td>
<td>delay of matching network as LTI system</td>
</tr>
<tr>
<td>$l_r$</td>
<td></td>
<td>delay of NLS</td>
</tr>
<tr>
<td>$u[n]$</td>
<td></td>
<td>complex-valued discrete output of IMN</td>
</tr>
<tr>
<td>$v[n]$</td>
<td></td>
<td>complex-valued discrete output of NLS</td>
</tr>
<tr>
<td>$h_i$</td>
<td></td>
<td>Volterra kernel</td>
</tr>
<tr>
<td>$\tau^{th}$</td>
<td></td>
<td>thermal time constant</td>
</tr>
<tr>
<td>$\tau^{gl}$</td>
<td></td>
<td>gate lag time constant</td>
</tr>
<tr>
<td>$\tau^{dl}$</td>
<td></td>
<td>drain lag time constant</td>
</tr>
<tr>
<td>$T[n]$</td>
<td></td>
<td>discrete thermal function</td>
</tr>
<tr>
<td>$V_{gs}$</td>
<td>V</td>
<td>gate source voltage of FET</td>
</tr>
</tbody>
</table>
## List of Symbols and Abbreviations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_{ds}$</td>
<td>A</td>
<td>intrinsic drain source current</td>
</tr>
<tr>
<td>$V_{field}$</td>
<td>V</td>
<td>voltage of electric field</td>
</tr>
<tr>
<td>$V_{ds,peak}$</td>
<td>V</td>
<td>intrinsic drain source peak voltage</td>
</tr>
<tr>
<td>$V_{gs,peak}$</td>
<td>V</td>
<td>intrinsic gate source peak voltage</td>
</tr>
<tr>
<td>$\pi$</td>
<td></td>
<td>partition of PCHIP</td>
</tr>
<tr>
<td>$f_i$</td>
<td></td>
<td>set of monotone data values at partition points</td>
</tr>
<tr>
<td>$H_k$</td>
<td></td>
<td>cubic Hermite base functions</td>
</tr>
<tr>
<td>$p(x)$</td>
<td></td>
<td>cubic polynomial</td>
</tr>
<tr>
<td>$\delta_n$</td>
<td></td>
<td>derivates to construct the piecewise interpolant</td>
</tr>
<tr>
<td>$G, A$</td>
<td></td>
<td>feedback loop transfer functions</td>
</tr>
<tr>
<td>$K$</td>
<td>dB</td>
<td>gain factor of a PA</td>
</tr>
<tr>
<td>$x_{DPD}[n]$</td>
<td></td>
<td>complex-valued discrete DPD output</td>
</tr>
<tr>
<td>$g(x[n]) = f^{-1}(x[n])$</td>
<td></td>
<td>complex-valued DPD output</td>
</tr>
<tr>
<td>$f(x)$</td>
<td></td>
<td>transfer function of PA</td>
</tr>
<tr>
<td>$e(t)$</td>
<td></td>
<td>error function for linearization</td>
</tr>
<tr>
<td>$\tau_{EPA}$</td>
<td></td>
<td>propagation delay caused by EPA</td>
</tr>
<tr>
<td>$x_n$</td>
<td></td>
<td>neuron input</td>
</tr>
<tr>
<td>$w_n$</td>
<td></td>
<td>weight factor</td>
</tr>
<tr>
<td>$b$</td>
<td></td>
<td>neuron bias</td>
</tr>
<tr>
<td>$W$</td>
<td></td>
<td>neuron weight matrix</td>
</tr>
<tr>
<td>$x$</td>
<td></td>
<td>neuron input vector</td>
</tr>
<tr>
<td>$H$</td>
<td></td>
<td>hidden layer units</td>
</tr>
<tr>
<td>$e$</td>
<td></td>
<td>neuron error</td>
</tr>
<tr>
<td>$e$</td>
<td></td>
<td>neuron error vector</td>
</tr>
<tr>
<td>$g$</td>
<td></td>
<td>gradient vector</td>
</tr>
<tr>
<td>$t$</td>
<td></td>
<td>neuron target value</td>
</tr>
<tr>
<td>$H$</td>
<td></td>
<td>Hebbian matrix</td>
</tr>
<tr>
<td>$J$</td>
<td></td>
<td>Jacobian matrix</td>
</tr>
<tr>
<td>$F_{i,p,q}(z)$</td>
<td></td>
<td>FIR filter of IQ model</td>
</tr>
</tbody>
</table>
### List of Symbols and Abbreviations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td></td>
<td>variable training factor</td>
</tr>
<tr>
<td>$d_{c_i,q}$</td>
<td></td>
<td>DC component of IQ model</td>
</tr>
</tbody>
</table>
1 Introduction

Telecommunication has rapidly evolved since telegraphy in the 19th century. Especially wireless communication has become a dominating part of telecommunication. "In the future, every human will be connected to every other human on the planet by a wireless network. But that’s just the beginning,” [62] A phrase that is obviously turning into reality. A mobile data forecast from 2021 predicts a growth of mobile traffic per month from 80 EB in 2021 to 370 EB in 2027 which is a factor of around 4.6 in just 6 years [60]. Fig. 1 shows the exponential increase in mobile data traffic including fixed wireless access (FWA). A challenge for the evolution of the wireless access networks is to serve this ever increasing demand in the near future. Having a look into the fundamentals of information theory, the Shannon-Hartley law defines the maximum data rate in symbols (baud) of a channel in relation to bandwidth and signal to noise ratio (SNR). Based on that law there are two main screws to increase wireless data rate, enhancing the bandwidth and signal quality. Bandwidth is a very limited resource in the shared medium air, therefore a high SNR is beneficial for efficient symbol encoding leading to higher modulation schemes with increased throughput. These circumstances tighten the requirements for the network equipment.

![Figure 1: Global mobile traffic growth forecast [60]](image)

Basestations are a key element of mobile networks with their wireless transceivers which have the most significant impact on the overall performance and power consumption. Especially the physical limitations and impairments of the analog components like PAs are a bottleneck regarding power efficiency and bandwidth. The increasing peak-to-average power ratio (PAPR) for signals
with highly complex modulation schemes make the design of PA difficult and degrades their energy efficiency. Digital preprocessing of the signals to lower PAPR and linearize the PA in order to enhance its energy efficiency are essential in transceivers especially for high output power.

The current change in mobile networks from single-input single-output (SISO) systems with low amounts of TRX to multiple-input multiple-output (MIMO) systems with a massive amount of TRXs increase the complexity of the systems enourmously. This leads to the question if systems with a certain level of AI can have a beneficial impact on the future of wireless networks and radio frequency (RF) TRXs.

Research objective

The research objective of this work is to analyze ANNs for digital predistortion (DPD) algorithms to mitigate impairments and linearize PAs in RF TRXs. The main goal is to find a highly flexible, resource efficient and device independent solution for DPD. The suitability of ANNs as a very abstract and generalized method to linearize PAs as a dynamic function fitting problem will be investigated. Furthermore, the benefits of the underlying paradigm of ANNs will play a role when it comes to solve memory effects of PA devices. Physically inspired modeling of the DPD by ANNs for LTM effects are of special interest. Studies on the implementability of the proposed ANN based solutions will show the feasibility realization in digital hardware for physical real-time signal processing systems which are widely used for DFEs in TRXs.

Scope of Work

The scope of this work is to show the benefits and flexibility of ANNs for predistortion of PAs in TRXs. The strength of ANNs as a generalized model to mitigate impairments and to fit transfer functions of physical RF components like a PA device as a black box are shown in different examples. Enhancing the black box by taking more impairments, non-idealities and dynamic effects such as memory behavior of PAs into account becomes a real challenge for ANNs and show their limits. Facing the requirements of physical real-time processing such as stability and uninterruptible operation of transmission under low latency condition are a challenge for any predistortion and impairment mitigation system.
Previous Work

The existing work in the field of PA linearization and predistortion is mainly based on analytical approaches. A recent overview on the analytics for modeling and linearization of PAs is given in [70]. DPD algorithms break down to a function fitting problem which can be basically solved by polynomial approximations [44]. This technique has been further enhanced by orthogonal polynomials [50, 51, 64]. Volterra series representations of the DPD are another widely used analytical approach [3, 16, 37, 75]. Conventional and orthogonal polynomials have been used in enhanced predistortion systems to treat memory effects of PAs [2, 14, 27]. In advance, certain efforts have been spent to reduce complexity and computational effort of the growing polynomial field or volterra systems [72].

ANNs have been used in RF systems and applications. Most of the work focus on modeling and design of components or systems [46, 58]. A summary of ANNs for RF design and modeling is given in [74] and [70]. An early presentation of an ANN approach for linearization based on feed forward neural networks (FFNNs) with perceptrons is given by [67]. This ANN model has been further used and applied for predistortion of travelling wave tube amplifier (TWTA) in satellite communication by [73] and for RF PAs by [43]. Later work further enhanced the use of FFNNs [41] or tried different ANN models [32].

Thesis Outline

The tools for performance assessment are described in Chapter 2 which is the simulation environment and hardware test platform (HTP).

Chapter 3 is about PA modeling. A short introduction in PA nonlinearities and memory effects is given and a novel data-driven model for thermal memory effects of a PA based on intrinsic device temperature measurements is developed. Extracting the intrinsic voltages to determine the electric field between gate and drain of a Galliumnitride (GaN) transistor finalizes this chapter.

The following Chapter 4 gives a deeper insight on the nonlinearities with memory effects of PAs and DPD, feedforward- and feedback linearization techniques to mitigate such effects.

Chapter 5 is about ANNs for PA predistortion. The basic concept and the current state of research is part of this chapter. IQ imbalance mitigation using a simple FFNN with only two perceptrons is an example to show the basic
1 Introduction

principle. This is followed by a presentation of different FFNN designs for PA linearization. The FFNNs are enhanced to novel methods that mitigate memory effects caused by temperature drifts and charge trapping especially in high power devices using the intrinsic device temperature and voltages as input parameters for predistortion algorithms. This chapter presents how to mitigate IQ imbalance and PA nonlinearities with memory effects using a single FFNN.

Neurocomputing on FPGAs is part of Chapter 6. Implementation aspects and digital design of ANNs for a real-time hardware predistortion system based on the proposed FFNNs are part of this chapter.
2 Tools for Performance Assessment

The performance of the experiments and ANNs has been evaluated in simulation and on a HTP. The simulation environment is implemented in MATLAB® and a direct upconversion TRX is used to evaluate the linearization performance with different PAs.

Simulation and hardware tests have been performed with 4G Long Term Evolution (LTE) baseband signals of 5 MHz or 20 MHz bandwidth, which are clipped to reduce PAPR. The parameters of the test signals can be found in Table 1. Fig. 2 shows the spectrum of the test signals.

Table 1: LTE baseband signals for performance evaluation

<table>
<thead>
<tr>
<th></th>
<th>5 MHz LTE</th>
<th>20 MHz LTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3GPP Test Model</td>
<td>TM1.1</td>
<td>LTD5033</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>5 MHz</td>
<td>20 MHz</td>
</tr>
<tr>
<td>Length</td>
<td>1 frame (10 ms)</td>
<td>1 frame (10 ms)</td>
</tr>
<tr>
<td>Sampling Rate</td>
<td>30.72 MHz</td>
<td>122.88 MHz</td>
</tr>
<tr>
<td>PAPR</td>
<td>8 dB</td>
<td>9.5 dB</td>
</tr>
</tbody>
</table>

2.1 Simulation Environment

The ANNs have been designed within MATLAB® Neural Networking Toolbox version 7 [13]. For a better usability the models and ANNs are structured in a simple object oriented model (OOM). An abstract base class defines the general function interfaces inherited by the subclasses of which each represents a behavioral baseband model or ANN. Simulation scripts are used to construct models and ANNs to process and analyze test signals. The software structure in unified modeling language (UML) is shown in Fig. 3. The test signals are converted into an unified structure of type signal which is used for signal processing through different functions of the objects.

- **Scripts**: Simulation scripts can be individual for different test scenarios or parameter settings. Test signals are loaded and processed through instantiated PA models and ANNs of the desired and specified type. Functions of the SigAnalysis class are used to evaluate the performance and plot diagrams.
2 Tools for Performance Assessment

- **PA Models**: Each PA model is inherited from the abstract base class "PA_Model" which defines the structure and functions of the model.

- **Artificial Neural Networks**: Each ANN is an own class which is inherited from the abstract base class "Neural_Network". The ANN is built on construction of the object and stored in a private property. The train function executes ANN training on given input signal and target signal.

- **SigAnalysis**: Library of different signal analysis and plot functions.

### 2.1.1 Circuit Envelope Simulation

A test PA has been designed in an analog circuit simulation environment within Keysight® Advanced Design System (ADS) based on a bare die Wolfspeed® GaN high electron mobility transistor (HEMT) model with 120 W peak power. This physical device model has virtual output ports for intrinsic currents and voltages. Load-Pull simulations have been performed to determine the device’s optimal source and load impedance for 2.14 GHz carrier frequency. LC matching networks have been designed and tuned to optimize performance in respect to energy efficiency and linearity. A resistor and capacitor in parallel in the input matching network (IMN) ensure device’s stability according to the datasheet [8]. Impedance transformation by this components have to be considered in the input matching network. A complex modulated baseband
signal can be processed with this analog circuit design by using envelope circuit simulation. An IQ modulator at the PA input processes the sample based IQ values. This is used in a co-simulation environment with MATLAB® to test the predistortion performance. Fig. 4 shows the circuit design of the PA in circuit envelope simulation.
2.2 Hardware Test Platform

The basic principle of the HTP is depicted in Fig. 5. The signal source is a direct upconversion architecture with a DSP unit controlled by MATLAB®. The DSP is realized on a Xilinx Virtex 6 FPGA evaluation board [65]. Test signals can be transferred into the memory of the FPGA which sends and cyclic repeats the signal over a high speed serial interface through a digital to analog converter (DAC) which converts the digital IQ signal to analog before it is directly upconverted to RF in a direct upconversion IQ modulator. A high precision local oscillator (LO) generates the carrier frequency of the IQ modulator. The measurements are performed at a carrier frequency of 2.14 GHz if not otherwise noticed. A highly linear driver line-up drives the PA. The test signal is stored in an on-board memory and cyclic repeated.

A picture of the lab setup is shown in Fig. 6. The PA is placed in a climate chamber to evaluate the thermal memory performance by sweeping the environmental temperature. An oscilloscope measures the voltage of the thermal sensors.
2.2 Hardware Test Platform

A detailed block diagram of the HTP is depicted in Fig. 7. The FPGA is composed of two main building blocks, the signal path realized in hardware and an embedded processor for communication with a personal computer (PC) to control digital and analog hardware. The test signal can be transferred to
the memory of the signal path and the feedback signal can be read from the FPGA over Ethernet. Training of the ANNs and predistortion of the signal is performed offline based on the measured data in MATLAB®. The linearized signal is sent to the platform for predistortion performance evaluation. Time delay is adjusted by correlating input and feedback signals in MATLAB®.

![Figure 7: Block diagram of HTP](image)

### 2.2.1 Class-AB Power Amplifier

An in-house designed class-AB PA based on GaN technology with almost 42 dBm peak output power is used to evaluate the performance of the proposed ANN in hardware. Fig. 8 shows the PA and Fig. 9 shows the PA performance and characteristic of a continuous wave (CW) power sweep at 2.15 GHz. The blue curve indicates the transfer characteristic and the red curve shows the power-added efficiency (PAE). Small signal characterization results and further measurements are part of Chapter G.

### 2.2.2 Class-ABJ Power Amplifier

The RF PA for thermal memory predistortion performance evaluation is a GaN class-ABJ PA with 20 W peak power designed by Fraunhofer IAF in Freiburg. The test fixture is depicted in Fig. 10. The GaN power bar has four cells with four fingers in each cell. Special features of the device are three temperature sensors, located close to the transistor channel (see Fig. 21). One sensor on each side and one in the center of the device. The gap between temperature sensors and device fingers is 45 μm. Overall size of the powerbar is 2.75 mm x 0.75 mm. The PA has been designed for two different frequency bands at 2.14 GHz and 2.5 GHz. Peak output power, measured with a CW test signal, is between 42.4 dBm at 2.14 GHz and 44.6 dBm at 2.5 GHz with a drain efficiency
2.2 Hardware Test Platform

Figure 8: Picture of class-AB PA

Figure 9: CW power sweep of class-AB PA at 2.15 GHz

greater than 55%. Gain is between 10 dB and 11 dB and PAE is greater than 50% [6]. Results from small signal characterization of this PA are part of Chapter H.
Figure 10: Picture of class-ABJ PA [25]
3 Power Amplifier Modeling

PA modeling can be done in several ways. Physically, equivalent circuit based, behavioral or data-driven models are some of the most important to mention. This work will focus on behavioral and data-driven models. Behavioral models describe the relationship between input and output of the system in an abstract manner by an analytical approach. These models use polynomials, Taylor or Volterra series and even ANNs are considered to model the transfer characteristic or input to output relationship of a PA. The process of modeling the behavior of a PA breaks basically down to a function fitting problem. The combination of a generalized mathematical function like a polynomial function or Volterra series with a training algorithm to derive the coefficients of the function forms a set of data that represents the input and output relationship of the system to be modeled. Calculating the coefficients of the functions or mathematical models is done by iterative algorithms based on Newton's method up to more complex algorithms like Gauss-Seidel. Which training method to choose mainly depends on the desired mathematical representation of the system, the features the model should represent and the data set. Data-driven modeling is directly related to behavioral modeling as the data sets for deriving the coefficients of the behavioral models are usually based on physical parameters or measurements [10, 70].

When it comes to PA modeling, nonlinearities of the system are of main interest. Those nonlinearities can be described in a static and dynamic behavior. Static nonlinearities are easier to model and can be represented by a function that describes the input to output relationship as gain compression and its related phase shift (input amplitude to output amplitude distortion (AM-AM) and input amplitude to output phase distortion (AM-PM)). Dynamic nonlinearities are trickier to model. It is more difficult to derive these so called memory effects from measurements and data sets. Memory effects can usually be found in the uncertainties of the transfer characteristics and they are hard to be separated from each other and other uncertainties like noise. The spectral shaping of static and dynamic distortions is shown in Fig. 11 while Fig. 12 depicts a simple system block that indicates input, output and distortions as different ports and inputs of the black box.
3 Power Amplifier Modeling

Figure 11: Spectral shape of amplified signals for linear amplification (a), static distortions (b), dynamic distortions (c) and both (d) [17]

Figure 12: Basic system model

3.1 Static Nonlinearities

Static nonlinearities can be simply characterized with a single-tone CW test that sweeps the power at the input. Fig. 13 shows the typical AM-AM and AM-PM characteristic of a PA. As a single-tone has zero bandwidth this
3.1 Static Nonlinearities

Figure 13: AM-AM and AM-PM characteristic of a class-AB PA [72]

method does not take bandwidth dependencies of the transfer characteristic into account. The nonlinear behavior leads to harmonic and intermodulation tones at the output. Fig. 14 shows the symbol of a PA with $v_{in}(t)$ as its continuous input voltage function and $v_{out}(t)$ as the respective continuous output voltage function in time domain (TD).

$$v_{out}(t) = a_1 v_{in}(t) + a_2 v_{in}^2(t) + ... + a_N v_{in}^K(t) = \sum_{k=1}^{K} a_k v_{in}^k(t) \quad (1)$$

Within this work, only baseband discrete-time complex valued models are

Figure 14: PA symbol

used for simulations, tests and measurements. This leads to the following representation of the AM-AM and AM-PM behavior as depicted in Fig. 13.
3 Power Amplifier Modeling

\[ y[n] = F_{AM}(|x[n]|)e^{j(\varphi[n]+F_{PM}(|x[n]|))} = F_{AM}(A[n])e^{j(\varphi[n]+F_{PM}(A[n]))} \]  

(2)

where \( F_{AM}(x) \) is representing the AM-AM and \( F_{PM}(x) \) the AM-PM function, which is also known as Saleh’s model.

### 3.1.1 Polynomial Model

The polynomial model is one of the basic models and often used to describe the input to output relationship of a PA in terms of AM-AM and AM-PM behavior. This model expresses the static (memoryless) nonlinear behavior of a PA. Equation 3 describes the polynomial model.

\[ y[n] = \sum_{k=1}^{K} \alpha_k x[n]|x[n]|^{k-1} \]  

(3)

\( \alpha_k \) denotes the complex-valued coefficient and \( K \) the order of the polynomial function.

### 3.1.2 Saleh’s Model

A further enhanced model that needs four parameters to describe a PA’s behavior is Saleh’s model [54]. This model expresses AM-AM (equation 4) and AM-PM (equation 5) behavior in two separate equations.

\[ F_{AM}(x[n]) = \frac{\alpha_{AM}|x[n]|}{1 + \beta_{AM}|x[n]|^2} \]  

(4)

\[ F_{PM}(x[n]) = \frac{\alpha_{PM}|x[n]|^2}{1 + \beta_{PM}|x[n]|^2} \]  

(5)

In relation to the output \( y[n] \) equations 4 and 5 can be rewritten as follows:

\[ y[n] = F_{AM}(x[n])e^{j(\varphi[n]+F_{PM}(x[n]))} \]  

(6)
3.2 Memory Effects

Memory effects describe the influence of the signal history to the current output of the system. This means the output at \( t_0 \) of the PA is not only a product of the input at \( t_0 \), it is dependent on inputs from time \( t_m \) to \( t_0 \) while the interval \([t_m, t_0]\) includes the samples contributing to the output at \( t_0 \) and is known as memory depth. This energy storage effect causes that the same input to the PA does not have the same output at a different time with another history of the signal which is depicted in Fig. 15. The spectral shape or spreaded AM-AM and AM-PM characteristic curves of a measured PA show these effects, too. Another clear sign of memory effects are asymmetrical intermodulation distortions (IMD) in the spectrum of the output signal. Memory effects have been extensively studied to understand, characterize and model these effects [21, 49]. Due to the law of this topic it is still very challenging. Memory effects can be divided into short and long term effects in respect to their effective time constant.

![Figure 15: History of signal influencing present output of a PA](image)

3.2.1 Short Term Memory Effects

The time constant of short term memory effects (STM) is roughly on the order of the period of the RF carrier frequency. STM are caused by matching networks and device capacitances. Dynamic properties of capacitors and inductances influence the memory behavior of a PA [70].

3.2.1.1 Wiener Model

STM caused by matching networks can be represented as an linear time invariant (LTI) system. A cascade of the LTI system (equation 7) representing
the IMN and a static nonlinear system (NLS) (equation 8) describes the Wiener model depicted in Fig. 16 [57].

$$u[n] = \sum_{q=0}^{Q-1} \gamma_q x[n - l_q]$$  \hspace{1cm} (7)

$$y[n] = \sum_{k=1}^{K} \alpha_k u[n] |u[n]|^{k-1}$$  \hspace{1cm} (8)

Substituting equation 8 in 7 gives equation 9. The impulse response of the IMN is described by $\gamma_q$ and $l_q$ while $\alpha_k$ denotes the complex-valued coefficients of the NLS.

$$y[n] = \sum_{k=1}^{K} \alpha_k \left[ \sum_{q=0}^{Q-1} \gamma_q x[n - l_q] \right] \left[ \sum_{q=0}^{Q-1} \gamma_q x[n - l_q] \right]^{k-1}$$  \hspace{1cm} (9)

### 3.2.1.2 Hammerstein Model

According to the Wiener model, the Hammerstein model takes the effects of the output matching network (OMN) into account. Therefore, the Hammerstein model is a cascade of a NLS (equation 10) and an LTI system (equation 11) as shown in Fig. 17 [15].

$$y[n] = \sum_{k=1}^{K} \alpha_k \left[ \sum_{q=0}^{Q-1} \gamma_q x[n - l_q] \right]$$

Figure 16: Wiener model [57]

Figure 17: Hammerstein model [72]
3.2 Memory Effects

\[ v[n] = \sum_{k=0}^{K} \alpha_k |x[n]|^{k-1} \]  
(10)

\[ y[n] = \sum_{p=0}^{P-1} \gamma_p v[n] v[n - l_p] \]  
(11)

Substituting equation 10 in 11 delivers equation 12. In this case \( \gamma_p \) and \( l_p \) describe the impulse response of the OMN.

\[ y[n] = \sum_{p=0}^{P-1} \gamma_p \sum_{k=0}^{K} \alpha_k |x[n]|^{k-1} \]  
(12)

3.2.1.3 Wiener-Hammerstein Model

The Wiener-Hammerstein model takes both matching networks (IMN and OMN) into account [55]. Fig. 18 depicts the cascade of both LTI systems and the NLS in between. The substitution of equations 7, 8 and 11 can be written as equation 13.

\[ y[n] = \sum_{p=0}^{P-1} \gamma_p \sum_{k=0}^{K} \alpha_k |x[n]|^{k-1} \left( \sum_{q=0}^{Q-1} \gamma_q x[n - l_p - l_q] \right)^{k-1} \]  
(13)

3.2.1.4 Memory Polynomial Model

The STM of a PA can be modeled as a summation of parallel static nonlinear polynomials. Each polynomial represents the nonlinear behavior of a discrete time in the history of the signal (memory tap) [29]. Equation 14 defines the input to output relationship of the memory polynomial model.
3 Power Amplifier Modeling

Figure 19: Memory polynomial model [29]

\[
y[n] = \sum_{q=0}^{Q-1} \sum_{k=0}^{K} \alpha_{k,q} x[n-l_q] |x[n-l_q]|^{k-1}
\]

(14)

\(l_r\) is the individual delay for each static nonlinear subsystem.

3.2.1.5 Volterra Models

The Volterra theory of functionals is used to model the dynamic nonlinear behavior of a PA. Modeling the behavior as a Volterra series presumes that the system is weakly nonlinear, this means the system response is continuous and therefore representable by a finite series of contributing terms. A discontinuity would mean a strong nonlinearity and is not covered by this model [70]. Equation 15 describes the Volterra series model. \(h_i\) is the Volterra kernel of \(k\)-th order and \(L\) is the finite memory depth.

\[
y[n] = \sum_{l_1=0}^{L-1} h_1[l_1] x[n-l_1] + \sum_{k=2}^{K} \sum_{l_1=0}^{L-1} \ldots \sum_{l_k=0}^{L-1} h_i[l_1, \ldots, l_k] x[n-l_1] \ldots x[n-l_k]
\]

(15)

The accuracy of the model depends on the memory depth or order of the kernels. A higher order results in a better accuracy of the model but in an
exponentially increasing computational effort and decreasing stability of the system. This means that a proper design of a Volterra series model is needed in applications to ensure stability and computability.

A Volterra series can be developed of a Taylor series, a linear system or a cascade of linear and nonlinear components like the before mentioned Wiener-Hammerstein model. This means in reverse, that the memory polynomial and Wiener-Hammerstein models are special cases of the Volterra series [72].

### 3.2.1.6 Separable Function Model

The separable function model is basically an alternative representation of a Volterra model. It shows a similar performance and lower computational cost than the Volterra model [39]. Equation 16 describes this model.

\[
y[n] = \sum_{q=0}^{Q-1} \sum_{p=0}^{P-1} \sum_{k=1}^{K} \alpha_{k,q,p} x[n-l_q] x[n-l_p]^{k-1}
\]  

(16)

### 3.2.2 Long Term Memory Effects

LTM have a time constant which is much longer than the period of the RF carrier frequency. Thermal effects of the transistor, charge trapping and bias circuit effects have an influence on the LTM. Control circuitries such as bias and gain control or bias point adaption are another part of enhanced PA designs causing LTM. A PA model with LTM is shown in Fig. 20 [56].

![PA model with LTM](image)

Figure 20: PA model with LTM [56]

Equation 17 describes this model mathematically. \( f_{NLS} \) is the static, memoryless nonlinearity.
\[ y[n] = f_{NLS}(x[n]) + \alpha_{\text{long}} x[n] \sum_{q=0}^{Q-1} x^2[n - l_q] h[l_q] \] (17)

### 3.3 Thermal Memory

![Picture of GaN power bar and illustration with sensors](image)

Figure 21: Picture of GaN power bar (a) and illustration with sensors (b) [25]

Thermal memory effects and the related transfer characteristic depend on the channel temperature [69]. Therefore, it is desirable to measure the intrinsic temperature of the transistor channel. Measuring the temperature in the channel is not simply possible for some physical, electrical and RF reasons. Measuring the temperature close to the channel is a compromise that does not affect the RF behavior of the device and can give sufficient information about the intrinsic temperature of the device. The sensors are designed as PTC resistors with a resistance of 1.7 k\(\Omega\) at 20 °C. The distance between the sensors and transistor channel results in a heat flow which can be treated as a thermal low-pass filter. Fig. 21 shows the GaN device and depicts the geometrical structure of the power bar with the position of the sensors. In order to enhance the accuracy of measurements, three sensors are placed in the device, two on each side and one in the center.

A simple voltage divider has been built to measure the thermal dependent resistance of the sensors. The sensor signal is filtered to reduce RF noise in the filter circuit caused by coupling effects. Therefore, it is desirable to design the circuit of the filter close to the sensors. The bandwidth of the filter must be greater than the one of the before mentioned thermal low-pass between channel and sensors to avoid information loss. Suppression of the unwanted RF coupling has been achieved by designing the measurement circuit as depicted in Fig. 22. Two low ESR capacitors are used as a filter and the 1.4 k\(\Omega\) resistor form a voltage divider together with the sensor.

The PA has been placed in a climate chamber. The sensors have been powered with 4 V and an oscilloscope has been used to measure the voltage of the
3.3 Thermal Memory

Figure 22: Filter circuit and voltage divider of temperature sensors [25]

sensors in TD. The current at each sensor has been lower than 1 mA to keep the heat dissipation of the sensors themselves low. The 5 MHz LTE test signal described in Chapter 2 has been used to perform tests on RF operation [25].

3.3.1 Performance of Temperature Sensors

In a first step, the thermal sensors have been calibrated over a temperature range from -20 °C up to 160 °C in steps of 10 °C. The PA is powered off and disconnected from the signal source for sensor calibration. Only the temperature sensors are in operation. The characteristic curve of the voltage to temperature relation of the sensors is shown in Fig. 23. The characteristic shows an almost linear behavior as expected [24]. The voltage divider halves the voltage at 80 °C.

Figure 23: Performance of temperature sensor [24]
### 3.3.2 Thermal Time Constant Evaluation

In the next step, the thermal time constant has been measured. This is done by measuring the step response of the sensor by switching on the PA. The response behavior of the sensors, in this case the one in the center, is measured as shown in Fig. 24. Parameters derived from this measurement are used in equation 18 to calculate a thermal time constant \( \tau_{th} \) of 1.9 ms. The thermal low-pass characteristic can be determined with this value [25].

\[
v(t) = V_{\text{max}} \left( 1 - e^{-\frac{t}{\tau_{th}}} \right)
\]  

(18)

![Figure 24: Time constant measurement of temperature sensor [25]](image)

### 3.3.3 Characterization of Thermal Memory

The class-ABJ PA described in Section 2.2.2 has been driven into saturation using the 5 MHz LTE test signal and the environmental temperature in the climate chamber has been swept from 0 °C to 100 °C in steps of 10 °C to characterize thermal memory effects. The gain compression (AM-AM) and phase shift (AM-PM) characteristic of each temperature step have been measured. The measurement results of the discrete temperature steps have been fitted with a 7th order polynomial to eliminate noise and unwanted distortions. The
characteristic curves related to the in situ device temperature are shown in Fig. 25. Input and output power have been normalized [25].

Figure 25: AM-AM (a) and AM-PM (b) of class-ABJ PA over temperature [25]

3.3.4 Modeling Thermal Memory on Measurement Data

Thermal memory is relatively slow compared to the envelope or RF carrier frequency of the signal. It is one of the strongest LTM that impacts the dynamic behavior of the PA. Thermal energy is produced within the PA device for physical reasons. Most of the heat is dissipated by the current flow in the transistor channel and dissipated through the semiconductor material to a heat sink. It is clear that the heat distribution in the device is inhomogeneous and changes with the current input power, the history of the signal and environmental conditions. Modeling this effect is a very challenging topic. Volterra or memory polynomial models would heavily increase in complexity due to the relatively long time constant of this effect.

Within this work, a novel data-driven method to model thermal memory is proposed based on measurements of the intrinsic or in situ device temperature of a class-ABJ PA leading to an enhancement of the polynomial or Saleh model. The in situ temperature is measured close to the transistor channel and the AM-AM and AM-PM performance of the PA is characterized to build a data set which is used to determine the coefficients of the proposed model. Further details about the PA design and measurement method can be found in Section 2.2.2 and [6]. The basic principle of the model is depicted in Fig. 26 where the intrinsic temperature is another input value of the model.

The extended polynomial model can be formulated by rewriting (3) as follows:
3 Power Amplifier Modeling

\[y[n] = \sum_{k=1}^{K} a_k(T[n])x[n]|x[n]|^{k-1}\]  

(19)

\(a_k(T[n])\) are the complex-valued coefficients as a function of temperature.

According to this formulation, the Saleh model for a temperature input can be written as follows:

\[F_{AM}(x[n], T[n]) = \frac{\alpha_{AM}(T[n])|x[n]|}{1 + \beta_{AM}(T[n])|x[n]|^2}\]  

(20)

\[F_{PM}(x[n], T[n]) = \frac{\alpha_{PM}(T[n])|x[n]|^2}{1 + \beta_{PM}(T[n])|x[n]|^2}\]  

(21)

The AM-AM characteristic \(F_{AM}\) and AM-PM characteristic \(F_{PM}\) become a function of temperature. Rewriting equation 20 and 21 gives:

\[y[n] = F_{AM}(x[n], T[n])e^{j(\varphi[n] + F_{PM}(x[n], T[n]))}\]  

(22)

The additional information of the device temperature adds a further dimension to the static nonlinear model described in Section 3.3. Fitting the discrete AM-AM and AM-PM characteristics of the PA by a 3rd order polynomial in the dimension of temperature results in a 3-dimensional behavioral model with thermal memory. A surface plot of the AM-AM characteristic is shown in Fig. 27 and another one of the AM-PM characteristic is shown in Fig. 28 [24].
3.3 Thermal Memory

Figure 27: Surface plot of AM-AM characteristic over in situ temperature [24]

Figure 28: Surface plot of AM-PM characteristic over in situ temperature [24]
3.4 Charge Trapping in GaN Devices

Electron traps and holes occur in all kinds of semiconductors while they have a strong impact on the behavior of GaN and GaAs devices. The device’s current response shows a delay depending on the state of trapped charges or trapping condition. There are several root causes for trapping like native defects that may be present in bulk GaN due to its crystalline structure, impurities due to foreign atoms, extended defects created by more than a single atom or surface defects located at the outermost layer of the device. The energy level of the trap lies within the bandgap of the semiconductor and depends on its root cause, location, device temperature and terminal voltages which makes it difficult to determine [35]. Fig. 29 shows the gate lag (gl) effect which describes the degradation of the drain current due to a voltage change at the gate. Drain lag (dl) refers to the delayed current response for changing drain voltages as depicted in Fig. 30. The trapping condition depends on the voltages at the gate and ports or in other words the intrinsic electric field between gate and drain [9]. The time constants of charge trapping are much larger than the period of the RF carrier or even period of the baseband frequency of the signal. The positive peak voltage is considered for the drain while the negative peak voltage is taken at the gate because GaN transistors are normally on and need a negative voltage at the gate to adjust their bias point.

Time constants for gate lag seem to be the same for trapping and detrapping while drain lag shows different time constants for the two effects. Trapping and detrapping processes show short and long term behavior for gate as well as drain lag. Typical time constants for gate lag on short term are around several microseconds and for long term around several seconds. Drain lag shows short term time constants for trapping of around ten microseconds and long term of several hundred milliseconds. Detrapping for drain lag takes
3.4 Charge Trapping in GaN Devices

Figure 30: $V_{ds}$ and $I_{ds}$ response for drain lag with trapping effects [9]

several milliseconds for short term and several seconds long term which is much longer than for trapping [9].

3.4.1 Measuring Intrinsic Voltages

Measuring the intrinsic voltages at the gate and drain is essential to determine the charge trapping condition for modeling and predistortion. Fig. 31 shows the equivalent circuit of a GaN HEMT device with gate and drain lag circuit and ports for intrinsic gate and drain voltage. The opposite potential should be ideally taken at the intrinsic source before the parasitic source inductance and capacitance. If the opposite potential is taken at the common ground, $L_s$ and $R_s$ will influence the measurement and have a low pass effect on the intrinsic voltages.

3.4.2 Peak Voltage Detector

Peak voltages are relevant because of the relatively long time constants of the charge trapping effect of at least several microseconds compared to the period of the baseband or even RF carrier frequency of the signal. The lower frequency of the intrinsic voltages relaxes the sensors and A/D converters for digital signal processing. Therefore, detecting the peak from intrinsic voltages is the next step to determine $V_{field}$. This can be achieved by a peak voltage detector as depicted in Fig. 32.

Another method to derive the peak voltages are piecewise cubic Hermite interpolation polynomials (PCHIPs). This method can be easier to apply for data-driven transistor models without a dedicated port for intrinsic voltages.
like in simulations. PCHIP is an algorithm based on monotone cubic Hermite polynomials [18].

\[ \pi : a = x_1 < x_2 < ... < x_n = b \]  

(23)

\( \pi \) is a partition of interval \( I = [a, b] \) while \( f_i : i = 1, 2, ..., n \) is a set of monotone data values at the partition points. It is assumed that \( f_i \leq f_{i+1} (i = 1, 2, ..., n-1) \) or \( f_i \geq f_{i+1} (1, 2, ..., n-1) \). The goal is to construct on \( \pi \) a piecewise cubic function \( p(x) \in \zeta^1 [I] \) such that
3.4 Charge Trapping in GaN Devices

\[ p(x_i) = f_i \quad i = 1, 2, \ldots n \]  \hspace{1cm} (24)

\( p(x) \) is monotone. In each subinterval \([x_i, x_{i+1}]\), \( p(x) \) is a cubic polynomial represented as follows:

\[ p(x) = f_i H_i(x) + f_{i+1} H_2(x) + \delta_i H_3(x) + \delta_{i+1} H_4(x) \]  \hspace{1cm} (25)

where \( \delta_i = p'(x_i), j = i, i + 1 \), and \( H_k(x) \) are the usual cubic Hermite base functions for the interval \( l_i \):

\[
\begin{align*}
H_1(x) &= \phi\left(\frac{x_{i+1} - x}{h_i}\right) \\
H_2(x) &= \phi\left(\frac{x - x_1}{h_i}\right) \\
H_3(x) &= -h_i \psi\left(\frac{x_{i+1} - x}{h_i}\right) \\
H_4(x) &= -h_i \psi\left(\frac{x - x_i}{h_i}\right)
\end{align*}
\]

where

\[
\begin{align*}
h_i &= x_{i+1} - x_i \\
\phi(t) &= 3t^2 - 2t^2 \\
\psi(t) &= t^3 - t^2
\end{align*}
\]

The algorithm for calculating the derivatives \( \delta_1, \delta_2, \ldots, \delta_n \) to construct the piecewise interpolant to \((x_i, f_i) : i = 1, 2, \ldots, n\) is part of Chapter C. Fig. 33 depicts the intrinsic electric field derived from the envelope simulation described in Section 2.1.1 fitted with PCHIP.
Figure 33: Peak voltage detection with PCHIP
4 Predistortion Techniques

Predistortion for PAs is a widely used technology because of their typically nonlinear transfer characteristic. Especially close to the saturation region a PA gets highly nonlinear. Outside the saturation region, linearity of the PA depends on the design and device technology [10]. The drain efficiency and respectively the PAE are important for the energy efficiency and grow with higher input power and is highest in saturation as shown in Fig. 34. High energy efficiency and low power consumption is one of the main goals in PA design. Another aspect which has to be taken into account for PA designs are signal requirements. Higher order modulation schemes enhance the data throughput of communication systems but lead to a high PAPR in the transmit signal, a fact that tightens the requirements for PA designs resulting in high back off and increasing range of linear operation region of the PA. Standard requirements define the level of in band and out of band distortions. In order to meet the requirements of nowadays standards like LTE, 5G or upcoming 6G a predistortion is an essential part of TRX designs [70]. Furthermore, the implementation of predistortion depends on the trade-off between the power spending for the predistortion algorithm and the power saving in the PA gained by using predistortion. If the power consumed by the predistortion is higher than the power savings by higher efficiency values, e.g. in low power devices, a highly linear PA is preferred over a linearized transmitter [40, 70].

![Figure 34: Power amplifier drain efficiency and linearity [40]](image-url)
4 Predistortion Techniques

4.1 Feedback Linearization

Feedback linearization is an approach based on algebraic transformation of nonlinear into linear systems. The principle behind this method is to mitigate the nonlinear system output by a feedback loop. In this way, the current output signal is used to refine the future input signals. This is realized by comparing the actual output signal to a desired output. Then, an error signal is deduced and substituted from the input signal. Fig. 35 illustrates a typical feedback linearizer in a block diagram. Its transfer function is described by equation 26, where $x(t)$ and $y(t)$ are the feedback system input and output. $G$ and $A$ are PA and feedback loop transfer functions and $e(t)$ is the error signal.

$$y(t) = \frac{G}{1+GA} \cdot x(t)$$  \hspace{1cm} (26)

Figure 35: Feedback linearization block diagram [40]

Feedback linearization is relatively simple to implement. Nevertheless, this method suffers from several drawbacks, especially for wideband signals. Indeed, the feedback linearizer decreases the PA gain and suffers from potential instability. Improved linearity comes at the expense of gain reduction, as shown by equation 26. Additionally, it is generally challenging to keep $1+GA$ stable over a wide frequency range. In fact, the feedback system stability is determined by the feedback loop gain variation with frequency. This is difficult to control over a wide signal bandwidth [40].

4.2 Feedforward Linearization

The feedforward method, which is depicted in Fig. 36 is based on the generation of an error signal $e(t)$ [40].

The error signal is amplified to its original magnitude with an error power amplifier (EPA). After amplification, it is subtracted from the PA output.
4.3 Predistortion

In contrast to feedback and feedforward linearization outlined before, predistortion techniques add distortion components to the input signal to cancel nonlinearities of the PA so that the overall system of predistortion and PA becomes linear. The predistorter forms the inverse nonlinear transfer characteristic of the PA \((PA^{-1})\) [70]. The principle of this method is shown in Fig. 37.

\[ g(x) \] is the transfer function of the predistorter and \( f(x) \) is the transfer function of the PA. The output \( x_{DPP}[n] \) of the predistorter is given in equation 27. The relationship between \( g(x) \) and \( f(x) \) is defined by equation 28.


4 Predistortion Techniques

\[ x_{DPD}[n] = g(x[n]) \]  

(27)

\[ g(x[n]) = f^{-1}(x[n]) \]  

(28)

The entire system can be written as equation 29. \( K \) is the PA gain and \( f(x) \) the transfer function of the PA.

\[ y[n] = f(x_{DPD}[n]) = f(g(x[n])) = K \cdot x[n] \]  

(29)

The relationship between input and output in respect to the transfer characteristic of the predistorter and PA can be seen in Fig. 38.

The predistortion system depicted in Fig. 37 is only suitable for static nonlinearities and STM. A common method to mitigate LTM is to enhance this system by a feedback control loop. The information from PA feedback is used in an adaptation algorithm to adjust the coefficients of the predistorter (see Fig. 39).

Generally, a predistortion system can be build in several ways. If it comes to the design of a predistorter one has to choose between an analog or digital system. An analog predistorter is difficult to adapt on runtime while a DPD requires a digital signal processing block in the signal path. This requires analog to digital converters (ADCs) and DACs and is a question of computational cost and effort. Flexibility and adaptability of a DPD is higher compared to an
4.3 Predistortion

A predistorter can basically be built according to PA models using nonlinear base functions. As the predistorter function is the inverse of the PA transfer characteristic same models with a different set of coefficients can be used to linearize PAs. PA modeling is part of Chapter 3.

Figure 38: Input-output characteristic of predistortion system

Figure 39: Adaptive predistortion system [72]

analog design. Another aspect in the design of a predistorter is where to apply it. One can apply the predistorter in the baseband digital signal processing, on RF and even applying it on intermediate frequency (IF) is possible. Last option is not used in practice and IF TRX architectures for RF are anyway outdated. This work focuses on baseband DPD technique for linearization of PA which will be described in more detail in the following chapters.
4.3.1 Direct and Indirect Learning

The principle of direct and indirect learning is depicted in Fig. 39. Adapting the predistorter is done by comparing the input signal $x[n]$ to the normalized feedback signal $y[n]$ (direct learning). Another way is to derive the inverse characteristic of the PA from the predistorted signal $x_{DPD}[n]$ and feedback signal $y[n]$ (indirect learning). Both learning architectures show a similar performance [70].

4.4 Digital Predistortion (DPD)

An adaptive baseband DPD is built by enhancing the system proposed in Fig. 38 with up- and down-converters as well as an ADC and DAC (Fig. 40). In this system the other analog components are considered to be linear and thus add no distortions or impairments to the signal. Polynomials and Volterra series are the most widely used models for predistortion in practical applications [70].

Another very common way besides nonlinear base functions to build a DPD are look-up table (LUT) based methods. The gain and phase information of the predistorter is stored in a small amplitude interval in a LUT. The interval is selected according to the current input of the system and multiplied with the PA input as described by equation 30 [71].

$$x_{DPD}[n] = LUT(|x[n]|) \cdot x[n]$$  (30)
4.4 Digital Predistortion (DPD)

4.4.1 Memoryless DPD

A static predistortion system is based on the principle shown in Fig. 37. For baseband DPD a DAC and up-conversion part must be added between the predistorter and PA. The predistorter can be modeled with polynomials or Saleh’s model described in Section 3.1. The PA is usually characterized and the response of this characterization is used to compute the coefficients for the predistorter function or to fill a LUT. A memoryless DPD doesn’t need a feedback control loop. This is called open loop operation [70].

4.4.2 Mitigation of Memory Effects

A dynamic predistortion system for STM can be modeled based on the same open loop system as the memoryless DPD described in Section 4.4.1. For the predistorter function a memory PA model like Wiener-Hammerstein (see Section 3.2.1.3), memory polynomials (see Section 3.2.1.4) or Volterra series (see Section 3.2.1.5) is chosen. A LUT based approach requires a LUT for every memory tap.

The predistorter is usually built with a feedback control loop in combination with an adaptation algorithm to mitigate LTM. Dynamic effects with a relatively long time constant are mitigated by re-adjusting the coefficients of the predistorter function. A common way to implement an adaptive predistorter is to build a system as shown in Fig. 41. The proposed structure is based on direct learning. A training predistorter is used to ensure that the predistorter in the signal path can work without interruption. The performance of the training predistorter is compared to the real predistorter in the signal path. If the training predistorter reaches a state which has a better performance than the actual predistorter setting, the parameters or coefficients are transferred to the predistorter in the signal path [70, 71].

![Figure 41: Direct learning architecture with DPD parameter transfer](image-url)
Artificial Neural Networks for Predistortion of RF Power Amplifiers

ANNs are a discipline of ML. ML is considered to be a branch of AI. Actually, there is no clear definition for AI and even ML at all. One can say ML systems can learn from data in the manner of being trained. Abstract and generalized structures, methods or algorithms are often inspired by biology, nature and organic processes to solve problems with technical machines like computers [4].

ANNs are a famous ML approach inspired by neurobiology. An ANN usually consists of interconnected neurons structured in layers which are designed in a mathematical and computable manner [28]. There exist many different ANN models which are more or less close to neurobiology. FFNNs, recurrent neural networks (RNNs) or self organizing maps (SOMs) are some of the fundamental and most famous ANN models. Which model to choose for a certain problem depends primarily on its problem class. Common problem classes are function fitting, pattern recognition or classification. Linearization is a function fitting problem. Other important aspects influencing the decision of an ANN model are stability and computability. FFNNs based on perceptrons are a fundamental ANN model for function fitting problems. Their feedforward structure make them very stable [20]. This work uses FFNNs to mitigate impairments and linearize PAs.

5.1 Neuron Model

The neuron model or simply neuron is the fundamental building block of an ANN. In case of an FFNN, perceptrons are the most commonly used neuron model. A neuron can have one to many inputs which are individually multiplied by an input weight and summed up. An optional bias value can be added. The result is used as an input of a transfer function which gives the output of the whole neuron. Fig. 42 depicts the structure of a neuron while equation 31 gives the mathematical description of the neuron model [20]. The weight \( w \) and bias \( b \) values of a neuron are the variable parameters which are adjusted by training algorithms and are comparable to the coefficients in a Volterra series or polynomial function.
Equation 31 can be written in matrix form as follows, where $\mathbf{W}$ represents the weight matrix.

$$ y = f(\mathbf{Wx} + b) $$  \hspace{1cm} (32) 

### 5.1.1 Initialization of Neuron Weights

The initialization of neuron weights plays a significant role for the mathematical convergence and performance of the training algorithm. The simplest way to initialize neuron weights is to use random numbers or normally distributed random numbers. This leads to an initial state of the ANN that can be trained with a training algorithm like Levenberg-Marquardt (see Section 5.3). Nevertheless, a random initialization doesn’t lead to the best training performance. A brief assessment on the training performance for different ANN initialization methods is given by [48]. It turns out that the Nguyen-Widrow algorithm [45] leads to the best training performance for FFNNs. Own experiments have also shown that any other initialization method doesn’t lead to a convergent state of the ANN for complex-modulated signals at all.

The Nguyen-Widrow algorithm divides the initial weight settings into intervals over the output region of the neurons which depends on the transfer function. The hyperbolic tangent sigmoid function has a region of $(-1, 1)$ which means a length 2. With $H$ hidden units each unit will be responsible for an interval
of length \( \frac{2}{H} \). The interval is described as follows, where \( w_n \) is the weight at the respective input of the neuron and \( b \) the neuron’s bias value.

\[
-\frac{1}{w_n} - b \leq x \leq \frac{1}{w_n} - b
\]  

(33)

The interval has the length \( \frac{2}{w_n} \). Therefore

\[
\frac{2}{w_n} = \frac{2}{H} \quad w_n = H
\]

It is desirable to have the intervals overlap slightly, so \( w_n = 0.7H \). Bias value \( b \) will be picked that the intervals are located randomly between \(-1 \leq x \leq 1\).

\[
x = -\frac{b}{w_n} = \text{rand}(-1, 1) \\
\]

\[
 b = \text{rand}(-|w_n|, |w_n|)
\]

### 5.1.2 Transfer Functions

The transfer function is an essential part of the neuron model. Table 2 lists some of the most important transfer functions. Transfer functions are asymmetrical or symmetrical functions and most of them deliver fractional normalized output values in the range of \([0, 1]\) or \([-1, 1]\). In case of function fitting under the universal approximation theorem [11] the hyperbolic tangent sigmoid depicted in Fig. 43 is the most relevant function and is used in most of the ANN designs in this work besides the linear function [20].

### 5.2 Network Architectures

A single neuron is ready to use for function fitting and is able to fit a variety of functions which can be represented according to equation 31. For a larger degree of freedom and to enhance the flexibility and possibilities of function fitting, neurons are arranged in a network. An FFNN consists of at least one layer of which each neuron represents one output of the system. The number of inputs is flexible and each input is connected to any neuron. This means
Table 2: Neuron transfer functions [20]

<table>
<thead>
<tr>
<th>Name</th>
<th>Input/Output Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hard Limit</td>
<td>( f(x) = 0 ) ( x &lt; 0 )</td>
</tr>
<tr>
<td></td>
<td>( f(x) = 1 ) ( x \geq 0 )</td>
</tr>
<tr>
<td>Symmetrical Hard Limit</td>
<td>( f(x) = -1 ) ( x &lt; 0 )</td>
</tr>
<tr>
<td></td>
<td>( f(x) = 1 ) ( x \geq 0 )</td>
</tr>
<tr>
<td>Linear</td>
<td>( f(x) = x )</td>
</tr>
<tr>
<td>Saturating Linear</td>
<td>( f(x) = 0 ) ( x &lt; 0 )</td>
</tr>
<tr>
<td></td>
<td>( f(x) = x ) ( 0 \leq x \leq 1 )</td>
</tr>
<tr>
<td></td>
<td>( f(x) = 1 ) ( x &gt; 1 )</td>
</tr>
<tr>
<td></td>
<td>( f(x) = -1 ) ( x &lt; -1 )</td>
</tr>
<tr>
<td>Symmetric Saturating Linear</td>
<td>( f(x) = x ) ( -1 \leq x \leq 1 )</td>
</tr>
<tr>
<td></td>
<td>( f(x) = 1 ) ( x &gt; 1 )</td>
</tr>
<tr>
<td>Log-Sigmoid</td>
<td>( f(x) = \frac{1}{1+e^{-x}} )</td>
</tr>
<tr>
<td>Hyperbolic Tangent Sigmoid</td>
<td>( f(x) = \frac{e^x-e^{-x}}{e^x+e^{-x}} )</td>
</tr>
<tr>
<td>Positive Linear (ReLU)</td>
<td>( f(x) = 0 ) ( x &lt; 0 )</td>
</tr>
<tr>
<td></td>
<td>( f(x) = x ) ( 0 \leq x )</td>
</tr>
<tr>
<td>Competitive</td>
<td>( f(x) = 1 ) neuron with max ( x )</td>
</tr>
<tr>
<td></td>
<td>( f(x) = 0 ) all other neurons</td>
</tr>
</tbody>
</table>

that the number of neurons of a single layer FFNN depends on the amount of system outputs. Fig. 44 depicts the structure of a single layer FFNN.

Equation 34 describes the single layer FFNN in matrix form.

\[
y = f(Wx + b)
\]  

(34)

If the FFNN has to fit a function of higher orders, a single layer FFNN might not be sufficient. In this case, a multilayer FFNN can be used to extend the order of the FFNN. Hidden layers are added before the output layer and the neurons are interconnected between layers. Each output of the layer usually forms the input for each neuron of the next layer. While the number of
neurons in the output layer is dependent on the number of system outputs and therefore clearly to determine, the amount of neurons and even the number of hidden layers cannot be simply determined from the problem statement or the function to fit. There is actually no real measure or equation which gives a clear answer on this part of the network architecture for a certain problem. It has been proven by the universal approximation theorem that an FFNN with a single hidden layer can approximate any nonlinear function with any desired
error \([11]\). This theorem provides the mathematical proof that a single hidden layer is sufficient to fit any nonlinear and monotone-increasing function but does not prove that a single hidden layer is the optimal choice in case of a given overall number of neurons needed, learning speed or generalization capability. Using two hidden layers can provide a more stable and general solution \([59]\). Other sources state that more than two hidden layers do not add any further advantage in case of function approximation, learning speed or generalization without mathematical proof \([20]\). This leads to the assumption that either one or two hidden layers is the best choice for a higher order FFNN architecture.

An FFNN with two hidden layers is shown in Fig. 45. Equation 35 describes the three layer FFNN in matrix form.

\[
y^3 = f^3(W^3 f^2(W^2 f^1(W^1 x + b^1) + b^2) + b^2) + b^3
\]  

(35)

The FFNN has better generalization capabilities with a smaller number of neurons. This means it will respond correctly for patterns not used for training. With too many neurons, the network can be overfitted on the training patterns but it will fail on patterns never used in training \([68]\).

### 5.2.1 Time Delay Lines for Memory Effect Mitigation

Time delay lines (TDLs) can be added to the inputs of an ANN. This means that the input space of the network is expanded with each time delay by the number of inputs. An ANN with three inputs turns into an ANN with nine
inputs for three TDLs which means that each time step is represented by its own set of input values to each neuron in the first layer [20]. This principle is depicted in Fig. 46.

\[
\begin{align*}
\sum & \quad w_1 \\
x[n] & \quad w_2 \\
x[n-1] & \quad w_k \\
\ldots & \\
x[n-k] & \\
& \quad b \\
\rightarrow & \quad f(x) \\
& \quad y[n-k]
\end{align*}
\]

Figure 46: Neuron model with TDLs

TDLs in ANNs are comparable to memory taps of a Volterra series or polynomial predistorter. TDLs can be used to compensate short term memory effects. The output of the ANN is delayed by the amount of TDLs at the input.

5.3 Training

Training an ANN means to calculate or compute the weight and bias values of the neurons while optimizing an error function. A learning rule or a further advanced training algorithm is needed to train an ANN on a certain set of input and target values representing the problem or function to be trained. This kind of training is called supervised learning [28]. If the problem can be represented by a set of values like in the case of PA predistortion where the input to output behavior of the PA is a set of measured or simulated samples, supervised learning is usually the desired style to train the ANN. In order to find the inverse function of a PA the input signal samples are used as training targets while the distorted output samples are used as training inputs. Beside supervised learning there is reinforcement and unsupervised learning [20].

- **Supervised Learning**: The ANN outputs for certain inputs are compared to the target values. A learning rule or training algorithm is used iteratively to adjust the weight and bias values while optimizing the error function of the ANN output values and target values.

- **Reinforcement Learning**: This learning type is similar to supervised learning but instead of using the correct target for each ANN input the algorithm uses only a grade which is a measure of the network performance over some sequence of inputs.
Unsupervised Learning: Only input values are necessary for this type of learning. This seems to be convenient on the first thought but is impracticable in the case of function fitting. This type of learning is usually used for clustering applications.

For most training algorithms, the training data is split into three parts. Each part is either used for training, validation or test of the ANN. Most of the data is spent for training, about 70-80%. Training an ANN is an iterative process and the set of training data is used in each iteration to train, validate and test the performance of the ANN. The training algorithm has the goal to reduce the error between the output and target values. Mean squared error (MSE) measures, gradients or derivatives can be used to compare the performance in each iteration to a specified goal and as a breaking condition for training [20].

5.3.1 Learning Rules

Learning rules are the fundamental principle to calculate the weight and bias values of a single neuron. The perceptron learning rule is expressed by equation 36. The product of the neuron error $e$ is multiplied by the input vector $\mathbf{x}$ to calculate the next weight vector $\mathbf{w}_{k+1}$.

$$\mathbf{w}_{k+1} = \mathbf{w}_k + e\mathbf{x} \quad (36)$$

The error of the neuron is given by equation 37 where $t$ is the target value and $y$ the output of the neuron.

$$e = t - y \quad (37)$$

This rule can be simply extended to determine the bias value by assuming that the bias is a weight with input 1.

$$\mathbf{b}_{k+1} = \mathbf{b}_k + e \quad (38)$$

Other famous, further enhanced learning rules are Hebbian Learning, Linear Regression or Delta Learning [71]. These rules advance the before mentioned principle but are still only applicable to a single neuron.
5.3 Training

5.3.2 Advanced Training Algorithms

Training a multilayer ANN with many neurons requires more than just a learning rule and is difficult because the data propagates through different nonlinear elements (neurons) and paths through the network [71]. Error back-propagation (EBP) generalizes the delta learning for multilayer ANNs. EBP was the first algorithm for training multilayer networks. Fig. 47 shows EBP for an ANN with one output. The weight update rule is given by equation 39 [53].

\[ w_{k+1} = w_k - \alpha g_k \] (39)

Where \( \alpha \) is the learning constant and \( g_k \) the gradient vector.

An advanced training algorithm is required, especially if the training data contains distortions and uncertainties like in case of measured data. Computational performance and stability are important factors for training algorithms.

Levenberg-Marquardt training is such an advanced training algorithm which is commonly used to train FFNNs for complex function fitting problems. It is an enhancement of the Newton’s method, fast in computation and relatively stable [71, 73]. Equation 40 describes the weight update rule for Levenberg-Marquardt training.

\[ w_{k+1} = w_k - (J_k^T J_k + \mu I)^{-1} J_k e_k \] (40)

The parameter \( \mu \) is introduced by Levenberg-Marquardt and changed during training. If \( \mu = 0 \), the algorithm works as Gauss-Newton. An overview of different training algorithms, especially the Newton algorithm and its enhancements can be found in Table 3.

A brief description of LMT is part of Chapter B.
Table 3: Advanced training algorithms for FFNN [71]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Update Rule</th>
<th>Convergence</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBP</td>
<td>$w_{k+1} = w_k - \alpha g_k$</td>
<td>Stable, slow</td>
<td>Gradient</td>
</tr>
<tr>
<td>Newton</td>
<td>$w_{k+1} = w_k - H_k^{-1}g_k$</td>
<td>Unstable, fast</td>
<td>Grad. and Hessian</td>
</tr>
<tr>
<td>Gauss-Newt.</td>
<td>$w_{k+1} = w_k - (J_k^T J_k)^{-1} J_k e_k$</td>
<td>Stable, fast</td>
<td>Jacobian</td>
</tr>
<tr>
<td>LMT</td>
<td>$w_{k+1} = w_k - (J_k^T J_k + \mu I)^{-1} J_k e_k$</td>
<td>Stable, fast</td>
<td>Jacobian</td>
</tr>
</tbody>
</table>

5.3.3 Assessment on Training Performance

Training an FFNN is a mathematical optimization problem. An optimization algorithm optimizes the MSE of the FFNN output compared to the expected value in case of supervised learning. Weight and bias values are updated by the optimization algorithm after every training iteration. Another significant role plays the gradient of the MSE function over the training iterations. Both values also act as a breaking condition to end the training. This work uses Levenberg-Marquardt training. This algorithm needs to compute the Jacobian Matrix which consists of differential equations (see equation 40). In order to evaluate the training performance of FFNNs for DPD different configurations with varying amounts of hidden layer neurons and TDLs are trained on the data based on the circuit envelope simulation (see Section 2.1.1). As this model contains memory effects, TDLs increase the performance of the network significantly as seen in Fig. 48. This test uses the same FFNN with 20 neurons in each hidden layer.

Another test is performed with different hidden layer configurations. This FFNN has 5 TDLs at the input and compares the performance for 10, 15, 20 and 30 neurons in each of both hidden layers. Training performance of this test is depicted in Fig. 49.
5.3 Training

Figure 48: Training performance for different TDL configurations

Figure 49: Training performance for different hidden layer configurations


5.4 IQ Imbalance Mitigation

IQ imbalance is a common problem in TRX caused by the analog IQ modulator. Non ideal behavior of the IQ modulator results in gain imbalance, direct current (DC) offset in IQ components, orthogonality errors caused by different delays or non exact 90° phase shift between I and Q path. A model of IQ imbalance is shown in Fig. 50. Gain imbalance is modeled by factor $k$, $p$ influences phase imbalance and $dc_i$, $dc_q$ add a DC offset to each path. Memory effects and frequency response can be added by introducing filters $F_i$, $F_p$, and $F_q$ to the model [5].

\[
\begin{align*}
I \text{ path} &\rightarrow F_i(z) \\
Q \text{ path} &\rightarrow k \\
F_p(z) &\rightarrow -\sin(p) \\
F_q(z) &\rightarrow \cos(p) \\
&\downarrow dc_i \\
&\downarrow dc_q \\
&\text{I path} \\
&\text{Q path}
\end{align*}
\]

Figure 50: IQ imbalance model [5]

IQ imbalance behavior of a TRX depends on the used mixer components and their memory effects like temperature, aging and other effects. For this reasons, an FFNN is applied to mitigate the distortions of IQ modulators.

5.4.1 Neural Network Design

The ANN for IQ imbalance mitigation is described in Fig. 51. It consists of only an output layer containing two neurons with linear transfer functions. I and Q are used as inputs for each neuron. Two neurons are sufficient in this case to solve a function fitting problem with two outputs of low complexity like IQ imbalance [22].

5.4.2 Simulation Results

The IQ model depicted in Fig. 50 has been used to simulate the performance of the FFNN shown in Fig. 51. Model parameters are given by Table 4. The 5 MHz LTE test signal has been shifted to a low IF of 5 MHz to make the effect of IQ imbalance visible in spectrum and not being covered by the signal itself.
5.4 IQ Imbalance Mitigation

Figure 51: FFNN for IQ imbalance mitigation [22]

Table 4: Parameters for IQ imbalance model

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>0.5</td>
</tr>
<tr>
<td>$p$</td>
<td>2</td>
</tr>
<tr>
<td>$d_{c_i}$</td>
<td>0.004</td>
</tr>
<tr>
<td>$d_{c_q}$</td>
<td>0.001</td>
</tr>
<tr>
<td>$F_i(z)$</td>
<td>[0.01 0.2 0.9 0 0]</td>
</tr>
<tr>
<td>$F_p(z)$</td>
<td>[-0.01 -0.02 0.9 -0.02 -0.01]</td>
</tr>
<tr>
<td>$F_q(z)$</td>
<td>[0 -0.75 2.5 -0.75 0]</td>
</tr>
</tbody>
</table>

The performance for a static IQ imbalance model without filters $F(z)$ mitigated by an FFNN without TDLs is shown by the spectrum in Fig. 52. Estimated error vector magnitude (EVM) value of the IQ model is 3.92% and can be compensated to 0% by the FFNN. Training takes about two iterations, which is very fast.

The IQ imbalance mitigation performance of an FFNN with three and five TDLs in combination with 5th order filters in the IQ model for memory effects is shown in Fig. 53. Estimated EVM value of the IQ model with memory filters is 17.98% and can be mitigated to 0.17% with 5 TDLs at the input. Training the network with TDLs takes less than ten iterations [22].

5.4.3 Measurement Results

IQ imbalance mitigation performance has been measured with a 5 MHz WCDMA test signal and the FFNN shown in Fig. 51 with five TDLs. Shifting the test signal to a low IF makes DC and mirror of the signal visible. Fig. 54 shows the spectrum of the uncompensated IQ modulator and Fig. 55 shows
Figure 52: Simulated IQ imbalance mitigation performance, 5 MHz LTE, 1 MHz video bandwidth (VBW), 100 kHz resolution bandwidth (RBW)

Figure 53: Simulated IQ imbalance mitigation performance with TDLs, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW [23]
the output spectrum after IQ imbalance mitigation. A suppression of about 20 dB for the mirror and 43 dB for the DC carrier has been achieved. The mitigation performance is limited by the noise floor of the feedback (FB) path which is not depicted and actually higher than the noise floor of the spectrum analyzer [22].

Figure 54: Measurement result of IQ imbalance, 300 kHz VBW, 30 kHz RBW [22]

Figure 55: Measurement result of IQ imbalance mitigation performance, 300 kHz VBW, 30 kHz RBW [22]
5.5 Power Amplifier Linearization

Linearization of PAs is much more complex than IQ imbalance mitigation due to the nonlinear and memory behavior of the PA described in Chapter 4. A PA constitutes a nonlinear time-variant (NLTV) system. This means for the FFNN that a more advanced design with nonlinear transfer functions in the neurons is required.

5.5.1 Neural Network Design

The straightforward method to design an FFNN for linearization is to use the network design proposed for IQ imbalance mitigation as depicted in Fig. 51 and enhance it with one or two hidden layers of neurons with nonlinear transfer functions.

Which transfer function to choose for the hidden layer neurons is mainly a question of the value range of the inputs and outputs of the network. It is a good choice to use fractional valued IQ samples which means that the range of I and Q values is between -1 and 1. This leads to the assumption that symmetrical functions may fit best for this purpose but the negative values of inputs and outputs don’t restrict the hidden layer neurons to symmetrical functions. Hyperbolic tangent-sigmoid functions are a good choice for this purpose as any nonlinear function can be fit by superpositioning sigmoid functions [11].

While the number of neurons in the output layer is clearly defined by the number of network outputs, the number of neurons needed to design the hidden layers is not simply to determine. One can say that the number of hidden layer neurons must be chosen that the whole FFNN is able to fit the function without over- and underfitting. It has turned out in experiments that about 10 to 30 neurons is a good measure for each hidden layer to achieve proper results in DPD applications for LTE signals up to 20 MHz bandwidth.

How many hidden layers to use is another open question. It is basically desirable to keep the computational efforts and resource costs low, which means that the number of neurons and layers should be kept as low as possible. One hidden layer is sufficient to fit any nonlinear function while two hidden layers have shown higher stability and fit faster in previous work as described in more detail in Section 5.2. The performance of FFNNs with two hidden layers of the same size is evaluated for PA linearization in this work. Fig. 56 depicts a straightforward designed FFNN with two hidden layers. TDLs are optional and can be used for memory effect mitigation.
5.5 Power Amplifier Linearization

It turned out in experiments that the use of the absolute value of IQ at the input leads to a significantly better performance and training convergence of the network.

5.5.2 Linearization of Saleh Model

The FFNN depicted in Fig. 56 has been simulated to linearize Saleh’s model described in Section 3.1.2. Fig. 57 shows the spectrum of the linearization performance with the 5 MHz test signal. Training the network has taken 385 iterations in this simulation and has been terminated by reaching the defined MSE breaking condition of $5 \cdot 10^{-10}$.

The adjacent channel power (ACP) has been increased from 37.8 dB to 64.9 dB which is an improvement of 27.1 dB. Furthermore, the estimated EVM value of this test could be reduced from 47.9 % to 0.09 %.

5.5.3 Linearization of Polynomial Model

The FFNN shown in Fig. 56 is applied to a 5th order polynomial model as described in Section 3.2.1.4. The nonlinear polynomial model has been tested with the 5 MHz and 20 MHz LTE test signals. The linearization performance with the 5 MHz test signal is shown in Fig. 58. The network adapts within 54 training iterations in simulation to the defined MSE of $5 \cdot 10^{-10}$ (see Fig. 61). The FFNN is able to almost fully linearize this polynomial PA model to a very low MSE value of $4.98 \cdot 10^{-10}$.

AM-AM and AM-PM characteristics related to this test are depicted in Fig. 60. The estimated EVM value is reduced from 4.65 % to 0.02 %. ACP is increased from 46.3 dB to 65.3 dB which is an improvement of 19 dB.

Another simulation with the same PA model and FFNN has been performed using a 20 MHz LTE test signal. The linearization performance shown in Fig. 62 is worse compared to the linearization performance with the 5 MHz LTE test signal but still sufficient. The ACP has been increased from 53.3 dB to
74.8 dB which is an improvement of 21.5 dB. Estimated EVM has been reduced from 4.77 % to 0.05 %. Training the FFNN takes significantly more iterations and hasn’t reach the MSE breaking condition of $5 \cdot 10^{-10}$ at all. Training has been terminated after 1000 iterations (see Fig. 63). Linearization complexity increases with higher signal bandwidth [70].

The results on the polynomial model show that the proposed FFNN with an overall size of 22 neurons is sufficient to linearize LTE test signals up to 20 MHz in simulation [23].
5.5 Power Amplifier Linearization

Figure 58: Linearization performance of polynomial model, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW [23]

Figure 59: Linearization performance of polynomial model in TD, 5 MHz LTE
Figure 60: AM-AM (a) and AM-PM (b) performance of polynomial model

Figure 61: Training performance of polynomial model, 5 MHz LTE
5.5 Power Amplifier Linearization

Figure 62: Linearization performance of polynomial model, 20 MHz LTE, 1 MHz VBW, 100 kHz RBW

Best Training Performance is 1.5307e−09

Figure 63: Training performance of polynomial model, 20 MHz LTE
5.5.4 Linearization of Memory Polynomial Model

TDLs have been discussed in Section 5.2.1 and are tested on a memory polynomial model as described in Section 3.2.1.4. The FFNN shown in Fig. 56 is enhanced by five TDLs at the input. Fig. 64 shows the resulting spectrum of this test. The ACP performance has been increased from 56.78 dB without linearization to 59.36 dB with linearization but without TDLs. Adding five TDLs improves the ACP by 3.41 dB compared to the same FFNN without TDLs. The ACP has been overall reduced by 5.99 dB with five TDLs compared to the nonlinearized model which means an ACP value of 62.77 dB. Training performance with TDLs is shown in Fig. 65. EVM has been reduced from 6.68 % to 0.39 % without TDLs and to 0.26 % with five TDLs.

Figure 64: Linearization performance of memory polynomial model, 20 MHz LTE, 1 MHz VBW, 100 kHz RBW
5.5 Power Amplifier Linearization

5.5.5 Measurement Results

Linearization performance of the proposed network with five TDLs has been measured using the setup described in Chapter 2 together with the class AB power amplifier (see Section 2.2.1) at a carrier frequency of 2.14 GHz. Linearization results on the 5 MHz LTE test signal show an ACP improvement of about 11 dB as depicted in Fig. 66 and an EVM improvement from 11.4 % to 0.83 %. ACP has been increased from 41.3 dB to 52.7 dB which is an improvement of 11.4 dB. The FFNN has been trained 300 iterations with a final MSE of $3.6 \times 10^{-5}$. The training performance (Fig. 67) shows that a local minimum has been quickly reached while the final MSE is higher compared to simulations which can be explained by the additional distortions in the measured feedback signal.

The hidden layer size of the FFNN has been increased to 15x15 for the next experiment. Fig. 68 shows the linearization performance with the 20 MHz LTE test signal. EVM has been reduced from 9.7 % to 0.97 %. ACP without linearization is 39.4 dB and has been reduced to 48.7 dB which is an improvement of 9.3 dB.

![Figure 65: Training performance with five TDLs of memory polynomial model, 20 MHz LTE](image)

**Best Training Performance is 1.1457e–07**

- **Train**
- **Best**
- **Goal**

Mean Squared Error (mse) vs. 136 iterations
Figure 66: Measured linearization performance of class AB PA, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW

Figure 67: Training performance of class AB PA, 5 MHz LTE
Figure 68: Measured linearization performance of class AB PA, 20 MHz LTE, 1 MHz VBW, 100 kHz RBW
5.6 IQ Imbalance Mitigation and PA Linearization

In order to enhance the difficulty of the NLTV problem solved by the FFNN the IQ modulator and PA are cascaded and trained to the FFNN. Fig. 69 shows the basic principle of cascading IQ modulator and PA.

Figure 69: Block diagram of cascaded IQ modulator and PA model

5.6.1 Neural Network Design

This problem has been trained to the FFNN described in Section 5.5.1. This straightforward FFNN design is able to solve the problem but it turned out in experiments that this configuration is not ideal concerning training speed and overall performance. A redesign of the FFNN helped to increase accuracy and training performance significantly. The input vector of the network is used as an additional input of the output layer as shown in Fig. 70. IQ imbalance is a highly linear problem and can be solved by an FFNN with two neurons and linear transfer functions while the highly nonlinear transfer function of a PA can be solved by additional layers with nonlinear sigmoid functions (see Section 5.5).

Figure 70: Custom FFNN for IQ imbalance mitigation and PA linearization [22]
The training performance of the comparison between the straight forward FFNN (see Fig. 56) and the redesigned custom FFNN is depicted in Fig. 71. The custom network adapts much faster and reaches the MSE goal of $5 \cdot 10^{-10}$ after 78 iterations. Both networks have the same hidden layer configuration of 10x10 neurons. It is assumed that the use of the input vector in the linear output layer guides the Levenberg-Marquardt training (LMT) to a better optimum in the sense of separability of both problems [20, 68].

![Training Performance](image)

Figure 71: Training performance of custom- vs. straight FFNN

### 5.6.2 Simulation Results

Both blocks of the cascaded model are represented by a suitable baseband model. The memoryless version of the IQ modulator model described in Section 5.4.2 is used in combination with the memoryless polynomial model of Section 5.5.3 to simulate the predistortion performance of the FFNN.

The FFNN is stable to train and almost fully mitigates both effects. Estimated EVM value can be mitigated from 12.63 % down to 0.2 %. Fig. 72 shows the simulated output spectrum of the model with the 5 MHz LTE test signal. ACP has been improved from 47.54 dB for the nonlinearized model to 64.43 dB which is an improvement of 16.89 dB.

AM-AM and AM-PM characteristics of the combined model are shown in Fig. 73 and Fig. 74 respectively. A zoom into the characteristic curves shows
that the FFNN mitigates the overlaying PA nonlinearities as well as the underlying distortions caused by the IQ modulator [23].
5.6 IQ Imbalance Mitigation and PA Linearization

![Simulated AM-PM characteristic of IQ/PA model (a) and zoomed (b)](image_url)

5.6.3 Measurement Results

The FFNN has been applied to the HTP where the IQ modulator and PA are represented by the measurement setup described in Chapter 2 with the class AB PA from Section 2.2.1. While the simulation has been memoryless, three TDLs have been added at the input for hardware measurements to mitigate memory effects of the real system. Fig. 75 shows the linearization performance of the proposed FFNN on the measurement setup. An image suppression of around 22.3 dB and a DC suppression of about 21.5 dB have been achieved. ACP has been increased from 40.7 dB to 49.3 dB which is an improvement of 8.6 dB.
Figure 75: Measured IQ imbalance mitigation & PA linearization, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW
5.7 Linearization of Thermal Memory

Thermal memory is one of the most significant LTM effects and therefore of high interest for predistortion. Previous work shows that the transfer behavior of the device depends on the channel temperature of the transistor which is called thermal memory [69]. This effect can be measured by thermal sensors within the PA device as described in Section 2.2.2. The DPD uses the sensor data to mitigate thermal memory. The goal of this approach is to train the PA's thermal dependent transfer characteristics to an FFNN with temperature input.

5.7.1 Neural Network Design

Designing an FFNN with thermal input is straightforward. There are other modeling approaches like circuit based models which try to model this effect by thermal dependent equivalent circuitries [7], polynomial based approaches trying to find the thermal dependent functions of coefficients [30] and others try to simplify this problem to a linear shift of the transfer characteristics [69]. From the perspective of an FFNN, processing this sensor data is simply an expansion of the input vector. The sensor data adds a further dimension to the input. Applying the sensor input to the FFNN proposed in Section 5.5.1 leads to the network design shown in Fig. 76 [25].

![Figure 76: FFNN for PA linearization with thermal input [25]](image)

5.7.2 Results on Linearization of Thermal Memory

The FFNN linearizes the thermal memory model described in Section 3.3.4. The simulation compares the linearization performance of the proposed FFNN to the same FFNN without thermal input.

5.7.2.1 Gaussian Thermal Distribution

The thermal behavior of this test is modeled by a Gaussian distribution function over the half length of the 5 MHz LTE test signal which is 5 ms. The
thermal distribution is depicted in Fig. 77 and distributes the channel temperature of the transistor in the range of $100^\circ C$ to $200^\circ C$. This data is used to train the FFNN with thermal input.

![Thermal Distribution Function](image)

![Time Domain Signal](image)

**Figure 77:** Gaussian thermal distribution function over 5 MHz LTE test signal in TD [25]

### 5.7.2.2 Envelope Shaped Thermal Distribution

The experiment has been performed with a different thermal distribution function which is depicted in Fig. 78. The first part of the function is a saturation function that models the switch on behavior of a PA while the second part of the function is the low pass filtered envelope or baseband signal. Low pass filter is set to a passband frequency of 0.5 kHz which is derived from the thermal low pass characterization of the thermal sensors described in Section 3.3.2.

![Time Domain Signal](image)

The linearization performance in frequency domain (FD) of this experiment is shown in Fig. 80 and Fig. 79 shows the spectral performance. An ACP of 51.65 dB has been achieved without thermal input while the network with thermal input gives an ACP of 57.13 dB which is an improvement of 5.48 dB. Training performance of the thermal memory FFNN is given in Fig. 81 which shows a quick descent in MSE. Estimated EVM is improved by 0.73 % with the thermal memory FFNN compared to the same network without thermal input.
5.7 Linearization of Thermal Memory

Figure 78: Envelope shaped thermal distribution function

The EVM result without thermal input is 1 \% compared to 0.27 \% with thermal input. Fig. 82 depicts the AM-AM and Fig. 83 the AM-PM characteristic of this test.

Figure 79: Linearization performance of thermal memory model with envelope shaped thermal distribution, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW
Figure 80: Linearization performance of thermal model in TD with envelope shaped thermal distribution, 5 MHz LTE

Figure 81: Training performance of thermal memory FFNN with envelope shaped thermal distribution, 5 MHz LTE
5.7 Linearization of Thermal Memory

Figure 82: Simulated AM-AM characteristic of thermal model with envelope shaped thermal distribution (a) and zoomed (b)

Figure 83: Simulated AM-PM characteristic of thermal model with envelope shaped thermal distribution (a) and zoomed (b)
5.7.3 Linearization Performance for TDD

Time division duplex (TDD) operation means that uplink (UL) and downlink (DL) transmission share the same carrier frequency and are separated in TD. DL and UL transmission don’t overlap in time which classifies TDD as half-duplex operation. This means the PA and its transistor device has to be switched on and off depending on the operational mode. Switching the PA heavily influences the transfer characteristic especially due to temperature drifts. Switching on heats up the device while switching off cools down the device. These load changes are critical for the stability of the linearization algorithm. Since 5G, TDD is fully dynamic which means the slot format and allocation is handled based on transmission and traffic load conditions [12].

The slot format and configuration depicted in Fig. 84 has been chosen for performance evaluation which results in the thermal characteristic shown in Fig. 85.

![Slot configuration for TDD](image)

D = downlink  U = uplink

The spectral performance of this test is depicted in Fig. 86 which shows an increased performance for the FFNN with thermal input in case of TDD scenarios. The ACP of the uncompensated model for TDD is 45.87 dB with an estimated EVM of 6.33 %. Without thermal compensation an ACP of 58.5 dB and EVM of 0.54 % can be achieved which can be further reduced to an ACP of 60.65 dB and an estimated EVM of 0.19 % with thermal compensation in case of this TDD scenario.
5.7 Linearization of Thermal Memory

Figure 85: Thermal distribution for TDD

Figure 86: Linearization performance for TDD operation, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW
5.8 Linearization of Thermal Memory and IQ Imbalance

As a further predistortion challenge thermal memory and IQ imbalance effects have been mitigated by a single FFNN like in Section 5.6. The difference is in this case, that the IQ modulator (see Section 3.3.4) is combined with the thermal memory PA model from Section 3.3 to take thermal memory effects into account.

![Block diagram of cascaded IQ imbalance and thermal memory PA model](image)

**Figure 87:** Block diagram of cascaded IQ imbalance and thermal memory PA model

5.8.1 Neural Network Design

The proposed custom FFNN to mitigate IQ imbalance and linearize PAs in Section 5.6.1 is enhanced by a thermal input. Fig. 88 depicts the FFNN design for this experiment. Hidden layers are designed with 20 neurons each and 5 TDLs are used at the input to enhance the performance for this challenging test.

![Custom FFNN for PA linearization and IQ imbalance mitigation with thermal input](image)

**Figure 88:** Custom FFNN for PA linearization and IQ imbalance mitigation with thermal input

[22]
5.8.2 Results on Thermal Memory and IQ Imbalance

The system has been simulated with the 5 MHz LTE test signal and Fig. 89 shows the spectral performance of this test. The performance has been evaluated against the same FFNN without thermal input like in Section 5.7. The envelope shaped thermal distribution function (see Fig. 78) is used for this experiment. The spectrum shows an ACP improvement of almost 8.15 dB for the FFNN with thermal input compared to the reference network without. ACP has been reduced from 41.03 dB for the nonlinearized model to 45.85 dB without thermal input and 53 dB with thermal input. Estimated EVM values have been decreased from 7.32 % to 2.55 % without thermal input and 0.3 % with thermal input which gives an estimated EVM improvement of 2.23 %. A MSE performance of $1.05 \cdot 10^{-8}$ for the FFNN with thermal input and $2.77 \cdot 10^{-6}$ without thermal input is achieved after 500 training iterations which is a significant improvement and explains the results in improved EVM and ACP. Training performance is depicted in Fig. 90 where the network with thermal input shows a significantly better performance than the reference network without thermal input.

![Power Spectral Density (PSD)](image)

Figure 89: Linearization performance of IQ imbalance and thermal memory, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW
Fig. 90: Training performance of IQ imbalance and thermal memory predistortion

Fig. 91 shows the AM-AM and AM-PM characteristic and gives an impression about the level of distortions.

The custom FFNN with direct feed of the input vector to the linear output layer and additional temperature input is able to mitigate both effects and outperforms the same network configuration without thermal input.
5.9 Linearization of Charge Trapping Memory

There is another significant LTM effect beside thermal memory that results in a dynamic behavior of the PA. Especially GaN semiconductors which are one of the latest semiconductor technologies for mobile communication systems suffer from charge trapping effects. The performance of GaN based semiconductors is limited by the presence of electron traps. Physical states on the surface of the device and crystallographic defects in the bulk act as traps for electrons [38]. Trapping of electrons effects the device’s current response to changing gate and drain voltages as described in Section 3.4 which leads to nonlinearities. Temperature and lighting seems to influence the charge trapping effect as well [9]. The dependency between trapping conditions and the intrinsic electric field between drain and gate leads to the next experiment where the voltage of the intrinsic field $V_{field}$ is fitted to an FFNN to mitigate charge trapping effects. The experiment is based on the circuit envelope simulation environment as described in Section 2.1.1. Due to the complex modulated signal, the device shows gate and drain lag effects because of varying gate and drain voltages. The test signal for training has a length of 5 ms which covers the short term trapping and detrapping effects.

5.9.1 Neural Network Design

Different hidden layer and TDL configurations of FFNNs have been designed and tested. The FFNN depicted in Fig. 92 is considered for presented results and performance tests. The electric field $V_{field}$ is the difference of the intrinsic gate and drain voltage according to equation 41 and derived from a PCHIP based peak envelope detector (see Section 3.4.2). $V_{field}$ is a further input for the FFNN with three TDLs and two hidden layers with 30 neurons each. Taking the difference of the intrinsic gate and drain voltage is a simplification and doesn’t reflect the different time constants of the gate and drain lag effect properly and limits therefore the performance. This simplification is done to reduce the complexity of the input, enhance stability as well as convergence of the neural network.

$$V_{field} = V_{ds, peak} - V_{gs, peak}$$

(41)

This leads to the assumption that the trapping condition could be determined by measuring the voltage of the intrinsic field $V_{field}$ described by equation 41.
5.9.2 Results on Linearization of Charge Trapping

The hidden layer size has been swept from 10 to 50 neurons in each layer and the TDLs have been set to 1, 3, 5 and 10. It turned out that the proposed configuration with three TDLs and 30x30 neurons in the hidden layers is a good choice for predistortion with $V_{\text{field}}$ as a fourth input. Complexity of the networks for this experiment is greater than in the tests before which can be explained by the relatively fast variation of $V_{\text{field}}$. The model for this experiment is explained in Section 3.4. Peak voltage of $V_{gsi}$ and $V_{dsi}$ is detected with PCHIP and normalized like all other values of the network. A reference network with the same configuration but without $V_{\text{field}}$ input has been trained on the same data. A sequence of the first 5 ms of the 5 MHz test signal has been used for training while the second half of 5 ms has been used for performance evaluation. The 5 ms covers the short term behavior of charge trapping effects which limits the performance. The long term behavior in the range of seconds requires longer training sequences and computational time and the training extends massively.

The nonlinearized model shows an estimated EVM of 9.31% and ACP of 48.8 dB which could be linearized to 0.174% EVM and 59.5 dB ACP for the network with $V_{\text{field}}$ input. The reference network shows a performance of 0.219% EVM and 60.23 dB ACP which is an EVM improvement of 0.045% and the ACP is improved by 0.73 dB in comparison. Fig. 94 and Fig. 95 show the spectral performance.

Training the FFNN with $V_{\text{field}}$ input takes longer at the beginning but outperforms the reference network after 75 iterations as depicted in Fig. 93.

The performance gain is small but stable. The experiments cover only the short term gate and drain lag effects with a 5 ms test signal. The computational effort and training time for the FFNN increases with longer signals but may lead to a better predistortion performance. Fig. 96 and Fig. 97 depict the AM-AM and AM-PM characteristic of this experiment.
5.9 Linearization of Charge Trapping Memory

Figure 93: Training performance of charge trapping predistortion

Figure 94: Linearization performance of charge trapping predistortion, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW
Figure 95: Zoomed linearization performance of charge trapping predistortion, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW

Figure 96: AM-AM characteristic of charge trapping simulation (a) and zoomed (b), 5 MHz LTE
5.9 Linearization of Charge Trapping Memory

Figure 97: AM-PM characteristic of charge trapping simulation (a) and zoomed (b), 5 MHz LTE
5 Artificial Neural Networks for Predistortion of RF Power Amplifiers

5.10 Comparison of Results

This section gives an overview of the achieved simulation and measurement results. Significant parameters and resulting EVM and ACP values are listed.

5.10.1 Simulation Results

Table 5: IQ imbalance simulation results

<table>
<thead>
<tr>
<th></th>
<th>Memoryless</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test signal</td>
<td>5 MHz</td>
<td>5 MHz</td>
</tr>
<tr>
<td>No. of TDLs</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>No. of neurons</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Image suppression</td>
<td>37 dB</td>
<td>44 dB</td>
</tr>
<tr>
<td>DC suppression</td>
<td>40 dB</td>
<td>44 dB</td>
</tr>
<tr>
<td>Est. EVM nonlin.</td>
<td>3.92 %</td>
<td>17.98 %</td>
</tr>
<tr>
<td>Est. EVM reduction</td>
<td>3.92 %</td>
<td>17.81 %</td>
</tr>
<tr>
<td>Achieved est. EVM</td>
<td>0 %</td>
<td>0.17 %</td>
</tr>
</tbody>
</table>

Table 6: PA linearization simulation results

<table>
<thead>
<tr>
<th>Model</th>
<th>Saleh</th>
<th>Polyn. Model</th>
<th>MPM</th>
<th>TMM</th>
<th>CTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test signal</td>
<td>5 MHz</td>
<td>5 MHz</td>
<td>20 MHz</td>
<td>20 MHz</td>
<td>5 MHz</td>
</tr>
<tr>
<td>No. of TDLs</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>No. of neurons</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>42</td>
</tr>
<tr>
<td>ACP nonlinearized</td>
<td>37.8 dB</td>
<td>46.3 dB</td>
<td>53.3 dB</td>
<td>56.78 dB</td>
<td>43.24 dB</td>
</tr>
<tr>
<td>ACP improvement</td>
<td>27.1 dB</td>
<td>19 dB</td>
<td>21.5 dB</td>
<td>5.99 dB</td>
<td>13.89 dB</td>
</tr>
<tr>
<td>Achieved ACP</td>
<td>64.9 dB</td>
<td>65.3 dB</td>
<td>74.8 dB</td>
<td>62.77 dB</td>
<td>57.13 dB</td>
</tr>
<tr>
<td>Est. EVM nonlin.</td>
<td>47.9 %</td>
<td>4.67 %</td>
<td>4.77 %</td>
<td>6.68 %</td>
<td>5.74 %</td>
</tr>
</tbody>
</table>
Table 7: Simulation results with cascaded models

<table>
<thead>
<tr>
<th></th>
<th>IQ &amp; PA</th>
<th>TMM &amp; IQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test signal</td>
<td>5 MHz</td>
<td>5 MHz</td>
</tr>
<tr>
<td>No. of TDLs</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>No. of neurons</td>
<td>22</td>
<td>42</td>
</tr>
<tr>
<td>ACP nonlinearized</td>
<td>47.54 dB</td>
<td>41.03 dB</td>
</tr>
<tr>
<td>ACP improvement</td>
<td>16.89 dB</td>
<td>11.97 dB</td>
</tr>
<tr>
<td>Achieved ACP</td>
<td>64.43 dB</td>
<td>53 dB</td>
</tr>
<tr>
<td>Est. EVM nonlin.</td>
<td>12.63 %</td>
<td>7.32 %</td>
</tr>
<tr>
<td>Est. EVM reduction</td>
<td>12.43 %</td>
<td>7.02 %</td>
</tr>
<tr>
<td>Achieved est. EVM</td>
<td>0.2 %</td>
<td>0.3 %</td>
</tr>
<tr>
<td>Image supression</td>
<td>34.23 dB</td>
<td>24.27 dB</td>
</tr>
<tr>
<td>DC supression</td>
<td>39.35 dB</td>
<td>25.32 dB</td>
</tr>
</tbody>
</table>

5.10.2 Measurement Results

Table 8: IQ imbalance measurement results

<table>
<thead>
<tr>
<th></th>
<th>Class AB PA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test signal</td>
<td>5 MHz</td>
</tr>
<tr>
<td>No. of TDLs</td>
<td>0</td>
</tr>
<tr>
<td>No. of neurons</td>
<td>2</td>
</tr>
<tr>
<td>Image supression</td>
<td>19.8 dB</td>
</tr>
<tr>
<td>DC supression</td>
<td>43.3 dB</td>
</tr>
</tbody>
</table>
Table 9: IQ imbalance and PA linearization measurement results

<table>
<thead>
<tr>
<th>Class AB PA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Test signal</td>
<td>5 MHz</td>
</tr>
<tr>
<td>No. of TDLs</td>
<td>3</td>
</tr>
<tr>
<td>No. of neurons</td>
<td>22</td>
</tr>
<tr>
<td>ACP nonlinearized</td>
<td>40.7 dB</td>
</tr>
<tr>
<td>ACP improvement</td>
<td>8.6 dB</td>
</tr>
<tr>
<td>Achieved ACP</td>
<td>49.3 dB</td>
</tr>
<tr>
<td>Image supression</td>
<td>22.3 dB</td>
</tr>
<tr>
<td>DC supression</td>
<td>21.5 dB</td>
</tr>
</tbody>
</table>

Table 10: PA linearization measurement results

<table>
<thead>
<tr>
<th>Class AB PA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Test signal</td>
<td>5 MHz</td>
</tr>
<tr>
<td>No. of TDLs</td>
<td>5</td>
</tr>
<tr>
<td>No. of neurons</td>
<td>22</td>
</tr>
<tr>
<td>ACP nonlinearized</td>
<td>41.3 dB</td>
</tr>
<tr>
<td>ACP improvement</td>
<td>11.4 dB</td>
</tr>
<tr>
<td>Achieved ACP</td>
<td>52.7 dB</td>
</tr>
<tr>
<td>EVM nonlin.</td>
<td>11.4%</td>
</tr>
<tr>
<td>EVM reduction</td>
<td>10.57%</td>
</tr>
<tr>
<td>Achieved EVM</td>
<td>0.83%</td>
</tr>
</tbody>
</table>
5.10 Comparison of Results

5.10.3 Comparison with Volterra DPD

An FFNN DPD approach has been compared to a Volterra series based approach in [19]. An FFNN with one hidden layer of 20 neurons and 4 to 8 TDLs outperformed the Volterra series derived DPD. Real and imaginary part have been treated as separate real valued inputs in this comparison.

5.10.4 Comparison with Memory Polynomial DPD

A generalized memory polynomial model DPD [42] with three memory taps and five coefficients has been compared to an FFNN based DPD with three TDLs and a hidden layer configuration of 10x10 and 30x30 neurons respectively. The nonlinearized PA model shows an ACP of 46.24 dB and an EVM of 9.31%. The experiment has been performed with the circuit envelope simulation described in Section 2.1.1. The FFNN DPD shows a slightly worse performance for 10x10 hidden layer neurons while it outperforms the memory polynomial DPD with 30x30 hidden layer neurons significantly. Performance values are described in Table 11.

Table 11: Simulation results of memory polynomial DPD compared to FFNN DPD

<table>
<thead>
<tr>
<th></th>
<th>PA Out</th>
<th>MemPol DPD</th>
<th>ANN 10x10</th>
<th>ANN 30x30</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACP reduction</td>
<td>-</td>
<td>9.54 dB</td>
<td>9.21 dB</td>
<td>10.65 dB</td>
</tr>
<tr>
<td>Achieved ACP</td>
<td>46.24 dB</td>
<td>55.78 dB</td>
<td>55.45 dB</td>
<td>56.89 dB</td>
</tr>
<tr>
<td>Est. EVM</td>
<td>9.31 %</td>
<td>0.36 %</td>
<td>0.38 %</td>
<td>0.25 %</td>
</tr>
</tbody>
</table>
Figure 98: Linearization performance of FFNN DPD and memory polynomial DPD in FD, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW.

Figure 99: Zoomed linearization performance of FFNN DPD compared to memory polynomial DPD, 5 MHz LTE.
6 FPGA Neurocomputing

FFNNs are basically a bunch of multiplications and summations. Each input of each neuron requires one multiplication and addition. This leads to a complexity of $O(n^2)$ with $n$ as the number of neurons and network inputs. A sequential computing platform is therefore not sufficient for highspeed real-time processing and training of FFNNs as it is required in DFEs in TRXs. An FPGA is a suitable platform for highspeed processing and parallelization especially of multiply and add functions. A comprehensive overview about FPGA neurocomputing is given by [76] and [47] which gives an advanced introduction in this topic. How to use floating point hardware accelerators to enhance FPGA neurocomputers is described in [33].

The aspects of FPGA neurocomputing break down to the implementation of the FFNN itself, the approximation of the transfer function of the neurons and the implementation of the training algorithm. The application of FFNNs for digital signal processing like DPD in TRX add a further constraint, the FFNN must remain in a stable state all the time for uninterrupted operation of the system which influences the training and update of FFNN parameters.

This work on FPGA neurocomputers for predistortion of PAs is based on Xilinx® Virtex 7 FPGA [1], Vivado 2014.1 [66] and makes assumptions based on this architecture and integrated development environment (IDE). First assumption is that FFNNs have to be built from scratch as there is no integrated technology or suitable block in the IP catalog available for distinct implementation of ANNs.

6.1 System Design

The general system design is depicted in Fig. 100 showing the main components of the FFNN based linearization system which is the hardware neural network (HWNN) designed as digital logic and a soft processor with optional operating system. The software neural network (SWNN), training algorithm (TA) and additional components like a parameter update logic (PUL) and phase modulation (PM) are designed in software for Xilinx® microblaze soft processor. Designing those components as digital logic circuitries would be probably faster in computation but leads to a much higher complexity and implementation effort. Especially complex training algorithms like LMT are difficult to implement in digital logic.
6 FPGA Neurocomputing

Figure 100: FPGA neurocomputing system for linearization

- **Hardware Neural Network (HWNN):** FFNN designed in digital logic using very highspeed integrated circuit hardware description language (VHDL).

- **Software Neural Network (SWNN)** A copy of the HWNN in software running on an embedded processor. It is used for training and updating the HWNN.

- **Parameter Update Logic (PUL)** Is responsible for transferring the weight and bias parameters into the HWNN without interruption.

- **Performance Monitor (PM)** Monitors the linearization performance, initiate re-training of the SWNN and decides if the SWNN has reached a better state than the actual HWNN to trigger parameter transfer.

- **Training Algorithm (TA)** Trains the weight and bias values based on forward and feedback signal.

### 6.2 Implementation of Transfer Functions

An efficient and accurate implementation of the transfer function for a neuron is a topic by its own. While a linear transfer function can be simply realized by a multiply and accumulate function, heavily nonlinear functions like hyperbolic tangent sigmoid are more difficult to implement. Enhancing the multiply and accumulate method for e.g. higher degree polynomials leads quickly to a high computational effort, resource requirement and is therefore not desirable for
implementing transfer functions. The most common method is to implement such functions as piecewise linear approximations [47, 63].

6.2.1 Piecewise Linear Approximation of Sigmoid Function

The principle of the hardware architecture to linearly approximate a sigmoid function is shown in Fig. 101. The function is sliced in $2^k$ intervals: $[L, L + \frac{\Delta}{2^k}], [L + \frac{\Delta}{2^k}, L + \frac{2\Delta}{2^k}], \ldots, [L + \frac{2^{k-1}\Delta}{2^k}, U]$ where $\Delta = U - L$, $L$ is the lower limit and $U$ is the upper limit of the function. Uniformly sized intervals are not the ideal choice to minimize the error function. In the ideal case intervals would be dependent on the gradient of the function to approximate for minimizing errors as it is obvious that the intervals should be ideally shorter in regions with higher inclination. The data of the interval points are stored in read only memory (ROM) $C_1$ and $C_2$. Therefore, it is desirable to choose uniformly sized intervals for a fast and easy implementation as $k$ bits of the input $x$ can be used to directly address the ROMs. Choosing small intervals, in this case the function has been sliced in $2^{10} = 1024$ intervals, can reduce the overall error but is not optimized in terms of hardware resource cost. In order to store 1024 values with 16 bit length, both ROMs need 4 kB of memory together. The values stored in the ROMs are used in a linear function to approximate the actual output value [47].

![Figure 101: Linear approximator for sigmoid function](image-url)

Figure 101: Linear approximator for sigmoid function
6.2.2 Assessment on Approximation Error

The piecewise linear approximation of the sigmoid function causes a nonlinear error compared to the ideal and continuous sigmoid function. Fig. 102 shows the error for an equidistant approximated sigmoid function with 10 bit resolution.

![Error of sigmoid function approximation](image)

**Figure 102: Error of sigmoid function approximation**

6.3 Hardware Neural Network

The HWNN is the main part of the whole system and represents the FFNN for predistortion in general (see Fig. 103). The I and Q values of the baseband signal are input and output of the system. The "Calc Abs" block calculates the absolute value of the current sample which is used as an additional input to the FFNN beside I and Q as described in Chapter 5. The whole design is driven by a synchronous clock (CLK) which is common in FPGA designs for signal processing. The other inputs are required to update the weight and bias values of the neurons. Arithmetic is based on fractional signed values and fix point multiplications. A critical issue to mention is the update of the weight and bias values. In order to ensure interruption free processing of the baseband signal, the values have to be updated stepwise according to the parameter transfer principle described in Section 4.4.2. The PUL as part of the embedded processor is responsible to control the update of the weight and bias values and is described in Section 6.4.4. Table 12 lists and describes the input and output ports of the HWNN.
6.3 Hardware Neural Network

Figure 103: Hardware neural network (HWNN)

Table 12: Input and output ports of HWNN

<table>
<thead>
<tr>
<th>Signal</th>
<th>Bus size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>1</td>
<td>Synchronous clock input at sampling rate</td>
</tr>
<tr>
<td>I</td>
<td>16</td>
<td>Real part of baseband signal</td>
</tr>
<tr>
<td>Q</td>
<td>16</td>
<td>Imaginary part of baseband signal</td>
</tr>
<tr>
<td>w_data_in</td>
<td>25</td>
<td>Weight value to set</td>
</tr>
<tr>
<td>weight_addr</td>
<td>4</td>
<td>Weight address in certain neuron</td>
</tr>
<tr>
<td>neuron_addr</td>
<td>4</td>
<td>Address of neuron in certain layer</td>
</tr>
<tr>
<td>layer_addr</td>
<td>4</td>
<td>Address of layer</td>
</tr>
<tr>
<td>write_en</td>
<td>1</td>
<td>Enable write to weight registers</td>
</tr>
<tr>
<td>upd_weights</td>
<td>1</td>
<td>Signal to synchronize update of weight values</td>
</tr>
</tbody>
</table>
6.3.1 Absolute Value Calculation

The absolute value of I and Q is calculated according to equation 42. Implementing a square root in FPGA is not simply possible and is usually done using a coordinate rotation digital computer (CORDIC) architecture which is very resource costly. Therefore, the calculation of a pseudo absolute value is done without the square root according to equation 43 which only needs two multipliers and one accumulator and is much faster to compute without square root.

\[ abs = \sqrt{I^2 + Q^2} \]  \hspace{1cm} (42)

\[ abs = I^2 + Q^2 \]  \hspace{1cm} (43)

6.3.2 HWNN Layers

The general structure of the layers is shown in Fig. 104. Each layer stores a preset of weight and bias values in a register array. This is necessary because only one weight value can be set per clock cycle with the proposed interface by the PUL while updating the values in the neurons requires changing the values in the whole layer in one clock cycle to ensure interruption free processing. The array is loaded by the PUL and preset for the next state of weight values for the layer’s neurons. A state machine in each layer is triggered by the upd_weights signal and is responsible for interruption free transfer of the preset weight values into neurons.

![Figure 104: HWNN layers](image_url)
6.3.3 Digital Design of Neurons

The basic design of a neuron with one input is shown in Fig. 105. It consists of a multiply and adder block, a bias register and a weight register for each input. The multiplications and additions scale with the number of inputs according to the equation described in Section 5.1. The multiplier of a Xilinx® Virtex 7 has an input data width of 25 bits which is chosen for the weight values. The output of multipliers and adders are rounded to 32 bits.

![Figure 105: Hardware design of single neuron](image)

6.3.4 Parallel Computation

Designing FPGAs is always a trade-off between resource cost and processing speed. FFNNs are a quite suitable architecture for high parallelization at the cost of resources but with the benefit of high speed signal processing. Not taking the calculation of the absolute value into account, this means that only one clock cycle per layer is required to process the data, which is only 3 clock cycles of the system clock for the whole predistortion process. Based on a common sampling-rate (SR) of e.g. 122.88 MSa/s in TRX systems this means a clock speed of factors higher than the SR within the layers and on neuron level. Table 13 gives an overview of the clock cycles needed for processing of each block. If the inner clock rate of the blocks is scaled up by the factor of
clock cycles the processing speed can be significantly enhanced in the manner of parallelization. Limitation is the maximum achievable clock rate in the FPGA.

Table 13: Clock cycles for each block

<table>
<thead>
<tr>
<th>Block</th>
<th>Clock Cycles t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calc. Abs.</td>
<td>$t_{ca} = 2$</td>
</tr>
<tr>
<td>Sigmoid transfer function</td>
<td>$t_{tf} = 3$</td>
</tr>
<tr>
<td>Neuron</td>
<td>$t_n(n_i) = 2 \cdot n_i + t_{tf}; n_i$ is the number of inputs</td>
</tr>
<tr>
<td>1st layer</td>
<td>$t_n(n_i = 3) = 9$</td>
</tr>
<tr>
<td>2nd layer</td>
<td>$t_n(n_i = 10) = 23$</td>
</tr>
<tr>
<td>output layer</td>
<td>$t_n(n_i = 10) = 23$</td>
</tr>
</tbody>
</table>

6.4 Co-Processing

Co-processing is done on an embedded processor and implemented in software. A common processor for Xilinx® FPGAs is Microblaze, a 32-bit RISC Harvard architecture core with a reduced instruction set optimized for embedded applications [36]. A monolithic implementation of the whole software is generally possible but due to the complexity not desirable. Therefore, a low-level real time operating system (RTOS) like freeRTOS [61] might be used as an operating system. The different software modules can be implemented as single threads and run as a multi-threading application. Fig. 106 shows the software architecture.

6.4.1 Software Neural Network

The SWNN is a representation of the FFNN implemented in hardware (HWNN) written e.g. in C language. It is used for training and performance evaluation. This FFNN is trained and compared to the state of the HWNN before the weight and bias values are transferred into the HWNN. The data types and formats are designed according to the HWNN to ensure the same computation results of both FFNNs.
6.4 Co-Processing

Figure 106: Co-processor for FPGA neurocomputing

6.4.2 Training Algorithm

The Levenberg-Marquardt training described in Section 5.3 is used to train the SWNN and implemented in software. An efficient open-source implementation of LMT can be found in [52].

6.4.3 Performance Monitor

The PM is an essential part of the system. It observes the performance of the HWNN by calculating the adjacent channel leakage ratio (ACLR) and EVM of the feedback signal. If one of the values is below a defined threshold the PM initiates a training sequence for the SWNN. It compares the performance of the SWNN to the performance of the HWNN and decides if the performance of the SWNN is better than the actual performance of the HWNN to transfer the weight and bias values to the HWNN. In the other case, if the performance is worse, the SWNN will be re-trained.

6.4.4 Parameter Update Logic

The PUL is responsible for an interruption free transfer of the weight and bias values of the SWNN into the HWNN. It is triggered by the performance
monitor if the SWNN has reached a better state than the actual HWNN. The PUL maps the weight and bias values of the SWNN to hardware addresses and writes them into the register arrays in the HWNN layers. After transferring all values into the HWNN the PUL rises the upd_weights bit to start the state machines of the HWNN layers that transfer the values from the temporary array in the layers into registers in the neurons.

Figure 107: Flow chart of performance monitor (PM)
7 Conclusion

This thesis reflects the first time a predistorter based on ANNs which has been extended to linearize dynamic effects by measuring the intrinsic state of the PA device. ANNs are models inspired by neuroscience and fall into the field of AI and ML. There is no clear definition for AI as there is none for intelligence. Machine learning means the ability of a machine or computer to learn from data. Neurons are represented in a mathematical manner, structured in networks and values are adjusted by suitable training algorithms either supervised or unsupervised. FFNNs are a simple ANN model for supervised training which is highly stable, resource efficient and suitable for low latency and real-time hardware signal processing.

Furthermore, this work exclusively shows the strength of ANNs to mitigate concatenated effects like IQ imbalance and PA nonlinearities in RF transmitters with a single FFNN. These function fitting problems of more or less unknown and dynamic transfer functions of IQ modulators or PAs should be determined in a flexible, stable and generalized manner to use the inverse function in a predistortion system. From ML perspective a composition of perceptrons in FFNNs with linear and sigmoid activation functions is sufficient to fit any nonlinear function efficiently. A short set (<10 ms) of measured or simulated signal samples in time domain at the relevant sampling rate is sufficient for small bandwidths up to 20 MHz to train the FFNN with LMT.

Mathematically seen, proposed algorithms have a maximum complexity of $x^3$ as a power of 3 is the highest order of a three layer FFNN. The FFNNs can be simply represented and computed as matrices. Training with LMT means to compute the Jacobian matrix to iteratively reduce the error between actual network output compared to an expected output from the training data in supervised learning.

Nonlinearities of PAs and other distortions are a serious problem in TRX systems. These effects influence energy efficiency and signal quality negatively. Characterizing, understanding and finding an analytical description of these effects is the common way and results in a more or less complex solution of the problems. Shortening this way to a sufficient solution for certain nonlinear effects has benefits in the design phase of TRX systems. A generalized approach like FFNNs with sigmoidal transfer functions which is able to fit any nonlinear function of a PA by training with measured samples as proposed is a beneficial solution to relax TRX designs and enhance flexibility of the
Conclusion

systems. Furthermore, the work has shown for the first time that FFNNs based on the same neurons and training algorithm can be used to mitigate IQ imbalance and PA nonlinearities with a single network. Enhancing the input space of an FFNN is also fairly simple compared to inserting a further variable into a mathematical function. This benefit is used to enhance the predistortion system to process intrinsic device parameters such as intrinsic temperature and intrinsic voltages. With these enhancements, the FFNN is able to represent a physically inspired model or inverse model of the PA. This physically inspired model for DPD opens new possibilities in TRX designs. The relatively complex and wideband RF feedback path could be relaxed or in some applications even completely saved by measuring and processing intrinsic parameters of the PA device.

Real-time processing is an important requirement for online TRX systems. Therefore, the implementability in hardware based on FPGA architectures has been studied. FFNNs are suitable for parallel processing with a concatenation of multiply and add functions, a common structure in FPGAs. The amount of required multipliers and adders scales with the amount of neuron inputs as each input is multiplied with a weight and summed up. Implementation of nonlinear transfer functions like sigmoid is challenging and can be done by linear approximation. Arithmetic conditions of the hardware have to be taken into account for training. Low complex FFNN designs are preferred for real-time processing in respect to processing speed, signal delay and resource cost.

Future Work

This work deals with dynamic nonlinear or memory effects of semiconductor devices for mobile basestation PAs with shallow learning ANNs. This is driven by the vision to find the most resource efficient and flexible solution with low latency by bottom up engineering. The smaller the FFNNs, the lower the resource effort for hardware implementation and computation as well as processing time which results in a small time delay or latency. The presented ANNs are small and fast to compute but have limited generalization capabilities especially when it comes to deep memory effects which depend on many parameters and conditions. Deep learning models are on the other side of ML and might show a better linearization and generalization performance at the cost of complexity. It might be definitely interesting to study the linearization performance of deep learning architectures in the context of basestation PA predistortion.
7 Conclusion

The work on the predistortion of charge trapping effects with intrinsic electric field processing and ANNs can be enhanced for longer time constants and measurements with real devices may be carried out.

While this work deals with two significant dynamic effects influencing the performance of PAs, thermal memory and charge trapping, there is another cause for memory effects in the circuitry of basestation PAs. Dynamic effects caused by the biasing and matching networks, especially in more complex PA designs also show significant influence on the PA performance. Further studies might have a closer look on those effects and how to deal with them in ML based predistortion algorithms.

Efficient hardware neurocompting is a big challenge, too. Low cost implementations for mass products require very efficient algorithms. Especially deep learning ANNs are very resource intensive to train and compute.
Bibliography


[65] Virtex-6 Family Overview. DS150. v2.5. Xilinx, Inc. 2015.


Appendix

A List of Publications by the Author

A.1 Patents


[P2] Envelope tracking radio frequency power amplifier (EP000002278705B1), earliest priority: 2009/06/30, JUESCHKE PATRICK, WIEGNER DIRK


A.2 Publications


Appendix


[Q8] Dieter Ferling, Patrick Jueschke, Xin Yu, Thomas Bohn, Anton Ambrosy, Yolanda Fernandez, Aykut Erdem, Andreas Pascht. Power Saving by Sleep Modes in Base Station Transceivers for LTE. IEEE Asia Pacific Microwave Conference (APMC) 2012


B Levenberg-Marquardt Training

The Levenberg-Marquardt algorithm was independently developed by Kenneth Levenberg [31] and Donald Marquardt [34] and is a numerical solution to minimize nonlinear functions of small- and medium sized problems. This method is widely used to train ANNs beside many other methods. One of the most significant breakthroughs in training of multilayer ANN was the EBP algorithm [53]. The following derivation of the LMT starts with the EBP algorithm [71].

Sum-squared error method is used to evaluate the training performance which is calculated by equation 44. \( p \) is the index of patterns from 1 to \( P \), \( m \) is the index of outputs from 1 to \( M \), \( i \) and \( j \) are the indices of weights from 1 to \( M \) and \( k \) is the index of iterations.

\[
E(x, w) = \frac{1}{2} \sum_{p=1}^{P} \sum_{m=1}^{M} e_{p,m}^2
\]  

(44)

where \( x \) is the input and \( w \) the weight vector. \( e_{p,m} \) is the training error at output \( m \) when pattern \( p \) is applied. It is given by

\[
e_{p,m} = t_{p,m} - y_{p,m}
\]

(45)

where \( t \) is the target vector and \( y \) the actual output vector.

The EBP method uses the first-order derivative of the total error function 44 to find a minimum in error space. Its gradient is defined by

\[
g = \frac{\partial E(x, w)}{\partial w} = \left[ \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, ..., \frac{\partial E}{\partial w_N} \right]
\]

(46)

with this definition of gradient \( g \) the EBP algorithm can be written as

\[
w_{k+1} = w_k - \alpha g_k
\]

(47)

Newton’s method assumes that \( g_1, g_2, ..., g_N \) are weight functions which are linearly independent.
Appendix

\[ g_1 = F_1(w_1, w_2, \ldots, w_n) \]
\[ g_2 = F_2(w_1, w_2, \ldots, w_n) \]
\[ \ldots \]
\[ g_N = F_N(w_1, w_2, \ldots, w_n) \]

(48)

Functions \( F_N \) form the relationship between weights and gradients. Taking
the first-order approximation of a Taylor series by unfolding each equation \( g_i \)
leads to the following equations.

\[ g_1 \approx g_{1,0} + \frac{\partial g_1}{\partial w_1} \Delta w_1 + \frac{\partial g_1}{\partial w_2} \Delta w_2 + \ldots + \frac{\partial g_1}{\partial w_N} \Delta w_N \]
\[ g_2 \approx g_{2,0} + \frac{\partial g_2}{\partial w_1} \Delta w_1 + \frac{\partial g_2}{\partial w_2} \Delta w_2 + \ldots + \frac{\partial g_2}{\partial w_N} \Delta w_N \]
\[ \ldots \]
\[ g_N \approx g_{N,0} + \frac{\partial g_N}{\partial w_1} \Delta w_1 + \frac{\partial g_N}{\partial w_2} \Delta w_2 + \ldots + \frac{\partial g_N}{\partial w_N} \Delta w_N \]

(49)

In combination with gradient vector \( g \) in equation 46 it follows that

\[ \frac{\partial g_i}{\partial w_j} = \frac{\partial}{\partial w_j} \left( \frac{\partial E}{\partial w_i} \right) = \frac{\partial^2 E}{\partial w_i \partial w_j} \]

(50)

This equation can be inserted in A.6:

\[ g_1 \approx g_{1,0} + \frac{\partial^2 E}{\partial w_1^2} \Delta w_1 + \frac{\partial^2 E}{\partial w_1 \partial w_2} \Delta w_2 + \ldots + \frac{\partial^2 E}{\partial w_1 \partial w_N} \Delta w_N \]
\[ g_2 \approx g_{2,0} + \frac{\partial^2 E}{\partial w_2 \partial w_1} \Delta w_1 + \frac{\partial^2 E}{\partial w_2^2} \Delta w_2 + \ldots + \frac{\partial^2 E}{\partial w_2 \partial w_N} \Delta w_N \]
\[ \ldots \]
\[ g_N \approx g_{N,0} + \frac{\partial^2 E}{\partial w_N \partial w_1} \Delta w_1 + \frac{\partial^2 E}{\partial w_N \partial w_2} \Delta w_2 + \ldots + \frac{\partial^2 E}{\partial w_N^2} \Delta w_N \]

(51)
The gradient vector has to be zero to minimize the total error function $E$.

\[
0 \approx g_{1,0} + \frac{\partial^2 E}{\partial w_1^2} \Delta w_1 + \frac{\partial^2 E}{\partial w_1 \partial w_2} \Delta w_2 + ... + \frac{\partial^2 E}{\partial w_1 \partial w_N} \Delta w_N
\]

\[
0 \approx g_{2,0} + \frac{\partial^2 E}{\partial w_2 \partial w_1} \Delta w_1 + \frac{\partial^2 E}{\partial w_2^2} \Delta w_2 + ... + \frac{\partial^2 E}{\partial w_2 \partial w_N} \Delta w_N
\]

\[
...\]

\[
0 \approx g_{N,0} + \frac{\partial^2 E}{\partial w_N \partial w_1} \Delta w_1 + \frac{\partial^2 E}{\partial w_N \partial w_2} \Delta w_2 + ... + \frac{\partial^2 E}{\partial w_N^2} \Delta w_N
\]

Combining these terms with equation 46 leads to the following equations.

\[
\frac{\partial E}{\partial w_1} = -g_1 \approx g_{1,0} + \frac{\partial^2 E}{\partial w_1^2} \Delta w_1 + \frac{\partial^2 E}{\partial w_1 \partial w_2} \Delta w_2 + ... + \frac{\partial^2 E}{\partial w_1 \partial w_N} \Delta w_N
\]

\[
\frac{\partial E}{\partial w_2} = -g_2 \approx g_{2,0} + \frac{\partial^2 E}{\partial w_2 \partial w_1} \Delta w_1 + \frac{\partial^2 E}{\partial w_2^2} \Delta w_2 + ... + \frac{\partial^2 E}{\partial w_2 \partial w_N} \Delta w_N
\]

\[
...\]

\[
\frac{\partial E}{\partial w_N} = -g_N \approx g_{N,0} + \frac{\partial^2 E}{\partial w_N \partial w_1} \Delta w_1 + \frac{\partial^2 E}{\partial w_N \partial w_2} \Delta w_2 + ... + \frac{\partial^2 E}{\partial w_N^2} \Delta w_N
\]

Equation A.10 written in matrix form:

\[
g = H \times w
\]

$H$ is the square Hessian matrix:

\[
H = \begin{bmatrix}
\frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\
\frac{\partial^2 E}{\partial w_1 \partial w_2} & \frac{\partial^2 E}{\partial w_2^2} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_N} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 E}{\partial w_N \partial w_1} & \frac{\partial^2 E}{\partial w_N \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_N^2}
\end{bmatrix}
\]

From equation 46, A.11 and A.12 follow:
Appendix

\[-g = H \Delta w \quad (56)\]

So follows:

\[\Delta w = -H^{-1} g \quad (57)\]

Which results in the Newton’s update rule:

\[\Delta w_{k+1} = -w_k - H_k^{-1}g_k \quad (58)\]

The Newton’s method has the disadvantage that the second order derivates of the total error functions have to be calculated, which is very complex for big ANNs with many weights. Introducing the Jacobian matrix \( J \) can simplify this calculation process.

\[
J = \begin{bmatrix}
\frac{\partial e_{1,1}}{\partial w_1} & \frac{\partial e_{1,1}}{\partial w_2} & \ldots & \frac{\partial e_{1,1}}{\partial w_N} \\
\frac{\partial e_{1,2}}{\partial w_1} & \frac{\partial e_{1,2}}{\partial w_2} & \ldots & \frac{\partial e_{1,2}}{\partial w_N} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial e_{1,M}}{\partial w_1} & \frac{\partial e_{1,M}}{\partial w_2} & \ldots & \frac{\partial e_{1,M}}{\partial w_N} \\
\frac{\partial e_{P,1}}{\partial w_1} & \frac{\partial e_{P,1}}{\partial w_2} & \ldots & \frac{\partial e_{P,1}}{\partial w_N} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial e_{P,M}}{\partial w_1} & \frac{\partial e_{P,M}}{\partial w_2} & \ldots & \frac{\partial e_{P,M}}{\partial w_N}
\end{bmatrix}
\quad (59)\]

The elements of the gradient vector can be calculated by integrating equation 44 and 46.

\[
g_i = \frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \left( \frac{1}{2} \sum_{p=1}^{P} \sum_{m=1}^{M} e_{p,m}^2 \right) = \sum_{p=1}^{P} \sum_{m=1}^{M} \left( \frac{\partial e_{p,m}}{\partial w_i} e_{p,m} \right) \quad (60)\]

Writing equation 59 and 60 in matrix form:
\[ g = J e \]  \hspace{1cm} (61)

where \( e \) is the error vector as follows:

\[
e = \begin{bmatrix}
e_{1,1} \\
e_{1,2} \\
\vdots \\
e_{1,M} \\
\vdots \\
e_{p,1} \\
e_{p,2} \\
\vdots \\
e_{p,M}
\end{bmatrix}
\]  \hspace{1cm} (62)

The elements of the Hessian matrix can be calculated by inserting 44 into 55:

\[
h_{i,j} = \frac{\partial^2 E}{\partial w_i \partial w_j} = \frac{\partial^2}{\partial w_i \partial w_j} \left( \frac{1}{2} \sum_{p=1}^{P} \sum_{m=1}^{M} e_{p,m}^2 \right) = \sum_{p=1}^{P} \sum_{m=1}^{M} \left( \frac{\partial e_{p,m}}{\partial w_i} \frac{\partial e_{p,m}}{\partial w_j} + S_{i,j} \right)
\]  \hspace{1cm} (63)

Where \( S_{i,j} \) is:

\[
S_{i,j} = \sum_{p=1}^{P} \sum_{m=1}^{M} \left( \frac{\partial^2 e_{p,m}}{\partial w_i \partial w_j} e_{p,m} \right)
\]  \hspace{1cm} (64)

Rewriting Hessian and Jacobian matrix:

\[ H \approx J^T J \]  \hspace{1cm} (65)

A combination of Newton’s update rule 58, equation 60 and 65 gives the Gauss-Newton rule:

\[
w_{k+1} = w_k - (J_k^T J_k)^{-1} J_k e_k
\]  \hspace{1cm} (66)
Calculating second-order derivates of total error function is no more necessary by introducing Jacobian matrix. The unstable convergence of the Gauss-Newton comes from the non-invertability of matrix $J^TJ$.

Levenberg-Marquardt solves this problem by introducing another approximation for the Hessian matrix.

$$H \approx J^TJ + \mu I$$  \hspace{1cm} (67)

The coefficient $\mu$ is always positive and multiplied with identity matrix $I$.

Inserting equation 67 in equation 66 gives the update rule of Levenberg-Marquardt training:

$$w_{k+1} = w_k - (J_k^TJ_k + \mu I)^{-1}J_ke_k$$  \hspace{1cm} (68)
C  Piecewise Cubic Interpolation Algorithm

This chapter describes the algorithm [18] to calculate the derivatives $\delta_1, \delta_2, ..., \delta_n$ for constructing the piecewise interpolants described in Section 3.4.2.

**Step 1:**

Initialize the derivatives $\delta_1, \delta_2, ..., \delta_n$ such that $sgn(\delta_i) = sgn(\delta_{i+1}) = sgn(\Delta_i)$. If $\Delta_i = 0$, set $\delta_i = \delta_{i+1} = 0$.

**Step 2:**

For each interval $I_i$ in which $(\alpha_i, \beta_i) \notin M$, modify $\delta_i$ and $\delta_{i+1}$ to $\delta_i^*$ and $\delta_{i+1}^*$ such that $(\alpha_i^*, \beta_i^*) \in M$, where

$$\alpha_i^* = \frac{\delta_i^*}{\Delta_i}$$
$$\beta_i^* = \frac{\delta_{i+1}^*}{\Delta_i}$$

It is necessary to note the interactions between adjacent intervals like $\beta_{i-1}\Delta_{i-1} = \delta_i = \alpha_i\Delta_i$. $\beta_{i-1}$ also changed when modifying $\delta_i$ to produce monotonocity on $I_i$. It must be also taken care to produce monotonocity on $I_{i-1}$. A way to accomplish this is by selecting a subset $Y \subset M$ that

(a) If $(\alpha, \beta) \in Y$, then $(\alpha^*, \beta^*) \in Y$, whenever $0 \leq \alpha^* \leq \alpha$ and $0 \leq \beta^* \leq \beta$.

(b) If $(\alpha, \beta)$, then $(\beta, \alpha) \in Y$.

The symmetry property is not essential, it is present in $M$. Step 2 may be replaced with

**Step 2A:**

For each $I_i$ in which $(\alpha_i, \beta_i) \notin Y$, modify $\delta_i$ and $\delta_{i+1}$ to $\delta_i^*$ and $\delta_{i+1}^*$ such that $0 \leq \alpha_i^* \leq \alpha_i$, $0 \leq \beta_i^* \leq \beta_i$, and $(\alpha_i^*, \beta_i^*) \in Y$. 
D Error Vector Magnitude Estimation

EVM is a relative measure of the in-band distortion. EVM describes the relative deviation of the receive vector in relation to a reference vector. Mobile communication standards define EVM requirements. These values depend on the constellation and therefore the modulation scheme of the signal as well as the signal standards. For QPSK modulated LTE signals the overall EVM budget is 17.5% while the EVM budget for 64 QAM is just 8%. EVM is usually counted in budget and these values define the limits of the whole transmit and receive chain. According to 3GPP standards, the EVM value is always measured on the receiver output based on standard defined test models. Using this standard related EVM measurement method is not feasible in simulations and measurements used in this work. As mentioned, EVM is about budget. It is desirable that each component in the whole chain keeps the additional EVM cost as low as possible. Under this assumption, an estimated EVM value calculation is used in this work to evaluate the performance of the ANNs and devices or models. The EVM value of each test sample is calculated following equation 69 while equation 70 calculates the summed estimated EVM value on a test vector.

\[
EVM = \sqrt{\frac{P_{\text{error}}}{P_{\text{ref}}}} = \sqrt{\frac{|x(t) - y(t)|^2}{|x(t)|^2}} \quad (69)
\]

\[
EVM_{\text{total}} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} EVM_i^2} \quad (70)
\]
ACP Calculations

ACP is an important value to rate the performance of a signal processing system. It describes the ratio between the power of the carrier signal and noise according to (71). Mobile communication standards like 4G LTE and 5G New Radio (NR) define further metrics and measures like ACLR, spectrum emission masks etc. As the proposed ANN based predistortion solutions are generally standard independent, a more general measure is needed to specify the performance. The out of band emissions of mobile communication signals are mainly caused by intermodulation products due to the nonlinear behavior of the system. The ACP figures give the distance of the signal carrier to the average level of sideband emissions as shown in Fig. 108.

\[
ACP = 10 \log \left( \frac{P_{signal}}{P_{noise}} \right) dB
\]  

(71)

Figure 108: ACP calculation in FD
F  Circuit Envelope Simulation Subcircuits

This section depicts the subcircuits of the PA design which is part of the circuit envelope simulation described in Section 2.1.1. Fig. 109 shows the transistor subcircuit with a Wolfspeed® 120W GaN transistor model, DC blocks and bias feeds.

Figure 109: Transistor circuit of envelope simulation

The input and output impedance of the transistor model has been derived from the datasheet and optimized in source-pull and load-pull simulations. The input and output matching networks have been derived from smith chart analysis and tuned to optimize the PA performance. Fig. 110 shows the IMN. The parallel capacitor and resistor at the output is needed to stabilize the PA according to the transistor datasheet. Simulations have shown that a capacitance of $6\,\text{pF}$ and a resistance of $1.5\,\Omega$ gives a decent stability without decreasing performance and efficiency too much. The impedance transformation of this stability circuit has to be considered for the design of the IMN. The 3-stage OMN (see Fig. 111) has been designed to match the impedance derived from load-pull simulation and slightly tuned to enhance the PA performance and optimize the center frequency to 2.14 GHz.
Figure 110: IMN of circuit envelope simulation

Figure 111: OMN of circuit envelope simulation
G Characterization Results of Class-AB PA

The class AB PA described in Section 2.2.1 has been measured and characterized. Picture 112 shows the lab setup for large signal characterization and predistortion tests. The S-Parameter characteristics derived from vector network analysis are shown in Fig. 113.

Figure 112: Lab setup for PA characterization and predistortion
Figure 113: Small signal measurement results of class-AB PA

Figure 114: CW power sweep of class-AB PA at 2.65 GHz
Appendix

H Characterization Results of Class-ABJ PA

The class ABJ PA described in Section 2.2.2 has been measured and characterized. The S-Parameter characteristics derived from vector network analysis are shown in Fig. 115 and Fig. 116. Gate voltage $V_g$ has been set to -2 V with a quiescent drain current $I_d$ of 250 mA.

Figure 115: $S_{21}$ of class-ABJ PA

Figure 116: $S_{11}$ (a) and $S_{22}$ (b) of class-ABJ PA
I Measurement Results with 3G UMTS Signal

Fig. 118 shows the linearization performance of an FFNN as described in Section 5.5.1 with 20 neurons in each hidden layer and three TDLs. The test has been performed with a 5 MHz UMTS test signal.

Figure 117: PA output, 5 MHz UMTS [22]

Figure 118: Linearized PA output, 5 MHz UMTS [22]
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Global mobile traffic growth forecast [60]</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Spectrum of LTE test signals with 5 MHz BW</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Structure of simulation environment in UML</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>Schematic of circuit envelope simulation in ADS</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>Basic principle of HTP [24]</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>Picture of HTP [24]</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>Block diagram of HTP</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>Picture of class-AB PA</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>CW power sweep of class-AB PA at 2.15 GHz</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>Picture of class-ABJ PA [25]</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>Spectral shape of amplified signals for linear amplification (a), static distortions (b), dynamic distortions (c) and both (d) [17]</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>Basic system model</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>AM-AM and AM-PM characteristic of a class-AB PA [72]</td>
<td>15</td>
</tr>
<tr>
<td>14</td>
<td>PA symbol</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>History of signal influencing present output of a PA [70]</td>
<td>17</td>
</tr>
<tr>
<td>16</td>
<td>Wiener model [57]</td>
<td>18</td>
</tr>
<tr>
<td>17</td>
<td>Hammerstein model [72]</td>
<td>18</td>
</tr>
<tr>
<td>18</td>
<td>Wiener-Hammerstein model [55]</td>
<td>19</td>
</tr>
<tr>
<td>19</td>
<td>Memory polynomial model [29]</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>PA model with LTM [56]</td>
<td>21</td>
</tr>
<tr>
<td>21</td>
<td>Picture of GaN power bar (a) and illustration with sensors (b) [25]</td>
<td>22</td>
</tr>
<tr>
<td>22</td>
<td>Filter circuit and voltage divider of temperature sensors [25]</td>
<td>23</td>
</tr>
<tr>
<td>24</td>
<td>Time constant measurement of temperature sensor [25]</td>
<td>24</td>
</tr>
<tr>
<td>25</td>
<td>AM-AM (a) and AM-PM (b) of class-ABJ PA over temperature [25]</td>
<td>25</td>
</tr>
<tr>
<td>26</td>
<td>PA block with temperature input</td>
<td>26</td>
</tr>
<tr>
<td>27</td>
<td>Surface plot of AM-AM characteristic over in situ temperature [24]</td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td>Surface plot of AM-PM characteristic over in situ temperature [24]</td>
<td>27</td>
</tr>
<tr>
<td>29</td>
<td>$V_{gs}$ and $I_{ds}$ response for gate lag with trapping effects [9]</td>
<td>28</td>
</tr>
<tr>
<td>30</td>
<td>$V_{ds}$ and $I_{ds}$ response for drain lag with trapping effects [9]</td>
<td>29</td>
</tr>
<tr>
<td>31</td>
<td>Equivalent circuit of a GaN HEMT model with gate and drain lag circuits [9]</td>
<td>30</td>
</tr>
</tbody>
</table>
List of Figures

32 Peak voltage detector ............................................. 30
33 Peak voltage detection with PCHIP ............................... 32
34 Power amplifier drain efficiency and linearity [40] .................. 33
35 Feedback linearization block diagram [40] ......................... 34
36 Feedforward linearization block diagram [40] ...................... 35
37 Predistortion principle ............................................. 36
38 Input-output characteristic of predistortion system ................. 37
39 Adaptive predistortion system [72] ................................ 37
40 Baseband DPD principle [72] ..................................... 38
41 Direct learning architecture with DPD parameter transfer ............ 39

42 Neuron model [20] .................................................. 42
43 Hyperbolic tangent sigmoid function ................................ 45
44 Single layer FFNN [20] ............................................. 45
45 Multilayer FFNN [20] ............................................... 46
46 Neuron model with TDLs .......................................... 47
47 EBP for ANN with one output ...................................... 49
48 Training performance for different TDL configurations .......... 51
49 Training performance for different hidden layer configurations .... 51
50 IQ imbalance model [5] ............................................. 52
51 FFNN for IQ imbalance mitigation [22] .............................. 53
52 Simulated IQ imbalance mitigation performance, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW .................. 54
53 Simulated IQ imbalance mitigation performance with TDLs, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW [23] ................. 54
54 Measurement result of IQ imbalance, 300 kHz VBW, 30 kHz RBW [22] .................................................. 55
55 Measurement result of IQ imbalance mitigation performance, 300 kHz VBW, 30 kHz RBW [22] ................................. 55
56 FFNN for PA linearization [23] ...................................... 57
57 Linearization performance of Saleh model, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW .................. 58
58 Linearization performance of polynomial model, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW [23] ......................... 59
59 Linearization performance of polynomial model in TD, 5 MHz LTE .................................................. 59
60 AM-AM (a) and AM-PM (b) performance of polynomial model 60
61 Training performance of polynomial model, 5 MHz LTE ........... 60
62 Linearization performance of polynomial model, 20 MHz LTE, 1 MHz VBW, 100 kHz RBW ................................. 61
63 Training performance of polynomial model, 20 MHz LTE ...
64 Linearization performance of memory polynomial model, 20 MHz LTE, 1 MHz VBW, 100 kHz RBW ...
65 Training performance with five TDLs of memory polynomial model, 20 MHz LTE ...
66 Measured linearization performance of class AB PA, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW ...
67 Training performance of class AB PA, 5 MHz LTE ...
68 Measured linearization performance of class AB PA, 20 MHz LTE, 1 MHz VBW, 100 kHz RBW ...
69 Block diagram of cascaded IQ modulator and PA model ...
70 Custom FFNN for IQ imbalance mitigation and PA linearization [22] ...
71 Training performance of custom- vs. straight FFNN ...
72 Simulated IQ imbalance mitigation & PA linearization performance, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW [23] ...
73 Simulated AM-AM characteristic of IQ/PA model (a) and zoomed (b) ...
74 Simulated AM-PM characteristic of IQ/PA model (a) and zoomed (b) ...
75 Measured IQ imbalance mitigation & PA linearization, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW ...
76 FFNN for PA linearization with thermal input [25] ...
77 Gaussian thermal distribution function over 5 MHz LTE test signal in TD [25] ...
78 Envelope shaped thermal distribution function ...
79 Linearization performance of thermal memory model with envelope shaped thermal distribution, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW ...
80 Linearization performance of thermal model in TD with envelope shaped thermal distribution, 5 MHz LTE ...
81 Training performance of thermal memory FFNN with envelope shaped thermal distribution, 5 MHz LTE ...
82 Simulated AM-AM characteristic of thermal model with envelope shaped thermal distribution (a) and zoomed (b) ...
83 Simulated AM-PM characteristic of thermal model with envelope shaped thermal distribution (a) and zoomed (b) ...
84 Slot configuration for TDD ...
85 Thermal distribution for TDD ...
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>86</td>
<td>Linearization performance for TDD operation, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW</td>
<td>77</td>
</tr>
<tr>
<td>87</td>
<td>Block diagram of cascaded IQ imbalance and thermal memory PA model</td>
<td>78</td>
</tr>
<tr>
<td>88</td>
<td>Custom FFNN for PA linearization and IQ imbalance mitigation with thermal input [22]</td>
<td>78</td>
</tr>
<tr>
<td>89</td>
<td>Linearization performance of IQ imbalance and thermal memory, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW</td>
<td>79</td>
</tr>
<tr>
<td>90</td>
<td>Training performance of IQ imbalance and thermal memory predistortion</td>
<td>80</td>
</tr>
<tr>
<td>91</td>
<td>AM-AM (a) and AM-PM (b) performance of IQ imbalance and thermal memory predistortion, 5 MHz LTE</td>
<td>80</td>
</tr>
<tr>
<td>92</td>
<td>FFNN for PA linearization with $V_{field}$ input</td>
<td>82</td>
</tr>
<tr>
<td>93</td>
<td>Training performance of charge trapping predistortion</td>
<td>83</td>
</tr>
<tr>
<td>94</td>
<td>Linearization performance of charge trapping predistortion, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW</td>
<td>83</td>
</tr>
<tr>
<td>95</td>
<td>Zoomed linearization performance of charge trapping predistortion, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW</td>
<td>84</td>
</tr>
<tr>
<td>96</td>
<td>AM-AM characteristic of charge trapping simulation (a) and zoomed (b), 5 MHz LTE</td>
<td>84</td>
</tr>
<tr>
<td>97</td>
<td>AM-PM characteristic of charge trapping simulation (a) and zoomed (b), 5 MHz LTE</td>
<td>85</td>
</tr>
<tr>
<td>98</td>
<td>Linearization performance of FFNN DPD and memory polynomial DPD in FD, 5 MHz LTE, 1 MHz VBW, 100 kHz RBW</td>
<td>90</td>
</tr>
<tr>
<td>99</td>
<td>Zoomed linearization performance of FFNN DPD compared to memory polynomial DPD, 5 MHz LTE</td>
<td>90</td>
</tr>
<tr>
<td>100</td>
<td>FPGA neurocomputing system for linearization</td>
<td>92</td>
</tr>
<tr>
<td>101</td>
<td>Linear approximator for sigmoid function</td>
<td>93</td>
</tr>
<tr>
<td>102</td>
<td>Error of sigmoid function approximation</td>
<td>94</td>
</tr>
<tr>
<td>103</td>
<td>Hardware neural network (HWNN)</td>
<td>95</td>
</tr>
<tr>
<td>104</td>
<td>HWNN layers</td>
<td>96</td>
</tr>
<tr>
<td>105</td>
<td>Hardware design of single neuron</td>
<td>97</td>
</tr>
<tr>
<td>106</td>
<td>Co-processor for FPGA neurocomputing</td>
<td>99</td>
</tr>
<tr>
<td>107</td>
<td>Flow chart of performance monitor (PM)</td>
<td>100</td>
</tr>
<tr>
<td>108</td>
<td>ACP calculation in FD</td>
<td>121</td>
</tr>
<tr>
<td>109</td>
<td>Transistor circuit of envelope simulation</td>
<td>122</td>
</tr>
<tr>
<td>110</td>
<td>IMN of circuit envelope simulation</td>
<td>123</td>
</tr>
<tr>
<td>111</td>
<td>OMN of circuit envelope simulation</td>
<td>123</td>
</tr>
<tr>
<td>112</td>
<td>Lab setup for PA characterization and predistortion</td>
<td>124</td>
</tr>
</tbody>
</table>
Appendix

113 Small signal measurement results of class-AB PA .......... 125
114 CW power sweep of class-AB PA at 2.65 GHz ............ 125
115 S21 of class-ABJ PA ........................................... 126
116 S11 (a) and S22 (b) of class-ABJ PA ....................... 126
117 PA output, 5 MHz UMTS [22] .............................. 127
118 Linearized PA output, 5 MHz UMTS [22] ................... 127

List of Tables

1 LTE baseband signals for performance evaluation .......... 5
2 Neuron transfer functions [20] .................................. 44
3 Advanced training algorithms for FFNN [71] ............... 50
4 Parameters for IQ imbalance model ......................... 53
5 IQ imbalance simulation results ............................... 86
6 PA linearization simulation results ............................ 86
7 Simulation results with cascaded models .................... 87
8 IQ imbalance measurement results ............................. 87
9 IQ imbalance and PA linearization measurement results ...... 88
10 PA linearization measurement results ......................... 88
11 Simulation results of memory polynomial DPD compared to FFNN DPD ........................................ 89
12 Input and output ports of HWNN .............................. 95
13 Clock cycles for each block .................................. 98
Mobile communication is rapidly growing. Increasing demands on capacity and bandwidth have to be addressed by future developments. This means higher signal requirements and bandwidth for transceivers in mobile base stations. Transceivers are the component with highest power consumption in a base station. Especially analog components show different impairments and nonideal behavior with negative effects on energy efficiency and signal integrity. These effects can be analyzed and mathematically described to build a specific digital signal processing algorithm, which mitigates certain effects. This work treats impairments from machine learning perspective. IQ imbalance of modulators as well as power amplifier nonlinearities are representative impairments with significant influence on the signal quality. These effects are trained to artificial neural networks (ANNs) for digital impairment mitigation. Furthermore it is shown that the ANNs are able to model different impairment effects with a single network and can be simply enhanced by further input parameters to mitigate dynamic effects. Physically inspired modeling of long term memory effects like thermal memory and charge trapping are a special focus of this work.