GRANT REPORT

N00014 -93-1-0855

July 1, 1993 - December 31, 1996

"Neural Network Models for Yield Enhancement in Semiconductor Manufacturing" and "Neural Networks for Inverse Parameter Modeling of IC Fabrications Stages"

Date: February 1997

Department of Electrical Engineering
University of Louisville
Louisville, KY 40292
jmzura02@starbase.spd.louisville.edu

Authors: Dr. Jacek M. Zurada, PI
Dr. Gregory L. Creech
Dr. Aleksander Malinowski
Andrzej G. Lozowski

19970317 044

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DTIC QUALITY INSPECTED 3

"Neural Network Models for Yield Enhancement in Semiconductor Manufacturing" and "Neural Networks for Inverse Parameter Modeling of IC Fabrication Stages" Final Report ONR Grant

February 25, 1997

Contents

List of Figures 3

List of Tables 5

Summary 6

1 Objectives 7

2 GaAs Integrated Circuit Fabrication and Characterization 9

2.1 Material and Fabrication Process Stages 12

2.1.1 Substrate/Active Layer Stage 12

2.1.2 Ohmic/Contact Metal Stage 14

2.1.3 Gate Recess Stage 15

2.1.4 Gate Metal Stage 15

2.1.5 Final Stage 15

2.2 High Density Test Reticle 16

2.2.1 Reticle Description 16

2.2.2 Characterization Data 16

2.2.3 Identification Scheme 18

3 Modeling of Process Stages 20

3.1 The SCRG-F Model 21

3.2 The S-F, CR-F, and G-F Models 25

DTIC QUALITY INSPECTED 3

4 Analysis of Sensitivity for Various Inputs **30**
4.1 Sensitivity Calculation 30
4.2 Sensitivity Analysis of Yield Models 32
4.2.1 S-F Model 32
4.2.2 CR-F Model 33
4.2.3 G-F Model 33
4.3 Reduction of Test Requirements through Network Pruning 34
4.4 Sensitivity Analysis for Yield Enhancement 35

5 Design Centering and Yield Maximization Approach **36**
5.1 The Approach 37
5.2 Principal Component Analysis 38
5.3 Inverse Projection through Neural Model 40
5.4 Optimization Algorithm 41

6 Results of Yield Maximization for Stage-to-Final Models **44**
6.1 Model Analysis 44
6.2 Design Centering 48
6.3 Yield Enhancement Test 54

7 Conclusions **70**

References **71**

8 Appendix **75**
8.1 Wtab Program Description 75
8.2 Inverse Mapping through Exhaustive Search Program 77
8.3 Process Stages Models 78
8.4 DESCENT Package 94
8.5 List of Published Papers 114
8.6 Papers 116

List of Figures

1 Microfabrication stage model block diagram. 8
2 Critical MESFET fabrication stages at which characterization takes place: a) substrate/active layer; b) ohmic/post-contact; c) gate recess; d) gate formation. 13
3 Visual representation of substrate electronic absorption, EL2: a) wafer map; b) histogram of measured EL2 values. 14
4 Visual representation of post-contact drain-source current, C-Idss: a) wafer map; b) histogram of measured C-Idss values. 15
5 High-density test reticle, HDTR. 17
6 Overlay of a 3" wafer on a grid high-density test reticle illustrating the identification scheme. 19
7 The SCRG-F non-reduced model: scattering plots for (a) training data, (b) testing data. 26
8 The SCRG-F non-reduced PCA pre-processed model: scattering plots for (a) training data, (b) testing data. 26
9 The S-F non-reduced model: scattering plots for (a) training data, (b) testing data. 27
10 The S-F non-reduced PCA pre-processed model: scattering plots for (a) training data, (b) testing data. 27
11 The CR-F non-reduced model: scattering plots for (a) training data, (b) testing data. 28
12 The CR-F non-reduced PCA pre-processed model: scattering plots for (a) training data, (b) testing data. 28
13 The G-F non-reduced model: scattering plots for (a) training data, (b) testing data. 29
14 The G-F non-reduced PCA pre-processed model: scattering plots for (a) training data, (b) testing data. 29
15 Sensitivity measure $\langle S_{ki} \rangle$ of each input for each output of the S-F model. . 33
16 Sensitivity measure $\langle S_{ki} \rangle$ of each input for each output of the CR-F model. 34
17 Sensitivity measure $\langle S_{ki} \rangle$ of each input for each output of the G-F model. . 35
18 Projection of a target and tolerance region into the input space of the model. 37
19 Modeling the microfabrication stage. 38
20 Approximating the yield probability. 42
21 Movement of the solution with respect to σ. 43
22 PCA error $\Delta_{\%}$ for: (a) SCRG distribution (b) S distribution. 47
23 PCA error $\Delta_{\%}$ for: (a) CR distribution (b) G distribution. 47
24 The SCRG-F model. Scattering plots for testing data. 49

25 The S-F model. Scattering plots for testing data. 50
26 The CR-F model. Scattering plots for testing data. 51
27 The G-F model. Scattering plots for testing data. 52
28 Design centering in SCRG-F fabrication stage. 57
29 Design centering in S-F fabrication stage. 58
30 Design centering in CR-F fabrication stage. 59
31 Design centering in G-F fabrication stage. 60
32 Percentage change to the inverse solution after centering. SCRG stage. . . . 61
33 Percentage change to the inverse solution after centering. S stage. 62
34 Percentage change to the inverse solution after centering. CR stage. 63
35 Percentage change to the inverse solution after centering. G stage. 64
36 Yield test for the SCRG-F reduced model. 66
37 Yield test for the S-F reduced model. 67
38 Yield test for the CR-F reduced model. 68
39 Yield test for the G-F reduced model. 69

DTIC QUALITY INSPECTED 3

List of Tables

1 The SCRG characteristics list. 22
2 The F characteristics list. 23
3 The S, CR, G and F characteristics list. 24
4 Eigenvalues of the SCRG distribution. 45
5 Eigenvalues of S, CR, and G distributions. 46
6 Target F values for SCRG-F process. 53
7 Target F values for S-F, CR-F, and G-F process. 54
8 Inverse and centered solutions for a given target and tolerances of 5, 10, 20%; SCRG stage. 55
9 Inverse and centered solutions for a given target and tolerances of 5, 10, 20%; S, CR, and G stages. 56
10 Fabrication yield for inverse solution and centered solution with allowed tolerances δ. 65

"Neural Network Models for Yield Enhancement in Semiconductor Manufacturing" and "Neural Networks for Inverse Parameter Modeling of IC Fabrication Stages"

February 1997

Summary

This project utilizes the neurocomputing technology towards modeling semiconductor fabrication processes for which analytical descriptions do not exist. Using data measured on GaAs fabrication lines of microwave circuits, partial fabrication stages as well as the complete process have been modeled. The developed models allow yield estimation and the determination as to which devices/wafers should be continued in the fabrication line. Subsequently, sensitivity analysis can be performed on process input factors to reveal which inputs carry more importance in producing final electronic devices having targeted specifications.

The concept of neural network models of fabrication process has also been applied for achieving improved yield of fabricated devices. Process data have been evaluated for principal components and reduced neural network models developed. Perceptron networks have then been inverted and process inputs recentered to maximize the yield. To achieve this, optimization has been performed in the reduced input space. The principal component analysis allows for re-adjustment of actual inputs for maximum yield. The software DESCENT, developed as a part of this project, can be used as a tool for practical design centering for maximum yield. It should be noted that results of modeling and centering, including the DESCENT package, are available to model and improve yield of other fabrication and manufacturing techniques.

1 Objectives

The majority of the development cost for many military systems is in the design and fabrication of the associated microelectronic integrated circuits (IC). In order to achieve acceptable fabrication yields, the integrated circuits must meet certain demanding system specifications involving complexity and frequency requirements [1].

The focus of this work is to develop a methodology allowing the maximization of the fabrication yield of Gallium Arsenide (GaAs) Microwave/Millimeter Wave Monolithic Integrated Circuits (MMIC) with respect to the material, process, and device parameters, while achieving acceptable circuit performance. The techniques developed in this project are applicable to GaAs IC technology and are also valid for other fabrication technologies, such as CMOS or BiCMOS technology.

Stages of the microelectronic circuit fabrication process can be efficiently modeled with multilayer perceptron neural networks (MPNN) after pre-processing by Principal Component Analysis (PCA) of the underlying data. These methods are found to be useful for capturing the relationships between various stages of the manufacturing process, as well as between the process parameters and the resulting device parameters. Once the process model is identified, a practical degree of design centering can be achieved by inverse modeling [2]. In most cases, the design centering problem requires the solution for the desired values of early manufacturing parameters or process attributes given the target performance of the final product, or output.

The first step in design centering is identification of the fabrication process [3]. The following critical stages of the GaAs IC fabrication process were selected for separate modeling [4]: substrate/active layer (S), post-contact/recess (CR), post-gate-metal (G), and final (F). The measurement data for the S process stage consists of ten substrate characteristics: optical scattering, Neut deep donor density, substrate resistivity, Hall mobility and carrier concentration, doping concentration, implant activation, drift mobility I, and drift mobility II.

Measurements for the CR stage include: drain-source saturation currents and resistances (both contact and recess), contact resistance, contact and ohmic metal sheet resistance, and ohmic metal layer width. G and F stage characteristics consist of the MESFET DC parameters: drain, gate, source, drain-source, drain-gate and gate-source resistances, drain-source saturation current, pinch-off voltage, and device transconductance. Also, gate metal sheet resistance and gate metal width are included in the G stage measurements.

The modeling of each stage requires first PCA pre-processing and then building a neural network, as shown in Fig. 1. The PCA extracts orthogonal principal directions in the multidimensional input space in descending order as characterized by corresponding eigenvalues (variances). This allows for reduction of the original input data dimension relevant

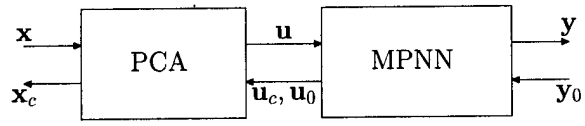


Figure 1: Microfabrication stage model block diagram.

for subsequent inverse modeling. It has been found that the characteristics describing the consecutive fabrication stages are mutually correlated. Our simulations have shown that the data distribution present in measurements for stages S (10 variables), CR (8 variables) and G (8 variables) can effectively be reduced to 5 abstract variables, with normalized estimation error below 10% (after compression and expansion).

Following the dimension reduction, a multilayer perceptron neural network (MPNN) is used to approximate the relationship between the input and output characteristics of a modeled stage. After training, the MPNN performs a nonlinear vector function f, which represents the stage-to-stage process model.

The model acquired in this manner can be used for design centering. Assuming target values and tolerances for the final semiconductor device characteristics at stage F of the fabrication process, the desired values of earlier stages S, CR, or G parameters can be obtained in two steps: first, the value of the abstract variables at the network input satisfying the output target can be found by inverting the function performed by the trained network. Afterwards, the optimum values of these variables ensuring maximum yield probability under the assumption of non-correlated normal distribution of process variations in the principal directions are estimated. Finally, the optimal center settings for the original input variables are evaluated based on using the inverse PCA operator.

2 GaAs Integrated Circuit Fabrication and Characterization

As new technologies begin to mature, it is essential that research facilitates the transition of technology into the development cycle of industrial systems. Artificial neural networks is an intelligent computing technology that has matured to the point where the feasibility of its application in the development of such complex systems needs to be explored and a theoretical framework established.

Neurocomputing models are helpful in situations when analytical solutions are not practical due to the complexity of physical models, and this applies to the pertinent stages of the IC process. This, in turn, is caused by the lack of analytical relationships of key process/device parameters from one processing stage to the next or among each other. These relationships are not yet fully understood nor are all the effects they have on the final DC performance parameters [4].

This research effort is focused on the development of the conceptual framework for the utilization of neurocomputing technology to achieve enhanced yield in integrated circuit manufacturing. As the complexity and speed requirements of ICs increase, resulting in submicron geometries and compact designs, it is increasingly difficult to achieve acceptable fabrication yields, and this work addresses solutions of the yield improvement problem.

Because modern ICs are vulnerable to inevitable statistical fluctuations of the starting material and are functions of rather complex and multivariate manufacturing processes, it seems that circuit yield may be increased to an acceptable level only through a costly iterative design and process adjustment approach. It is essential, therefore, that these fluctuations and relationships between various process stages are understood and properly modeled to possibly obtain a first-pass design. Such modeling typically involves relationships between the technological process attributes, layout dimensions, the resulting device parameters, and the final circuit performance.

Once a wafer enters the fabrication process, it is desirable and cost effective to predict and estimate the circuit yield as early in the process as possible. This prediction will aid in the decision of whether or not to continue processing this particular wafer. The data is gathered by performing measurements on device Parametric Test Structures and Process Control Monitors at certain stages in the process. After these measurements are compared to acceptance windows, a determination can be made whether to continue the wafer through the process.

Another yield-limiting factor in IC manufacturing is process uniformity. Although IC technologies are expected to produce uniform device properties over a large wafer area, this uniformity is especially difficult to achieve for GaAs IC technology because of material and processing deviations. From wafer to wafer, as well as within a wafer, there are large variations of material and process properties which strongly influence important factors in

final device/circuit performance.

Ideally, process engineers need to perform whole wafer characterization and analysis of key parameters at each critical fabrication step. This characterization can be attempted through testing. However, with the increasing complexity of the multivariate fabrication processes, the comprehensive testing required to provide proper characterization results in large quantities of data. Moreover, these data are costly to acquire. To reduce the testing requirements, it becomes necessary to determine a minimum number of parameters to which the yield-limiting characteristics are most sensitive [5].

One of the objectives of this research is to demonstrate the feasibility and to develop the theoretical framework for neurocomputing techniques for use as a practical and cost-effective tool suitable for IC development. The reasons for a neural computation approach to IC manufacturing are numerous. Neural networks are known for the ability to encapsulate multidimensional statistical properties present in large data sets [6]. By examining the network model, complex cause-effect relationships between input and output parameters become more evident [7].

Traditionally, a variety of deterministic and statistical approaches have been used to reduce the data to forms that could be easily interpreted by the user. Very often, however, the large volume of these results and the "curse of dimensionality" place them beyond the ability of the user to readily and effectively interpret [8, 9]. When accurate conclusions cannot be made in a relatively short time period, the data from these tests are ignored and potentially valuable information is lost. Also, these IC process/device modeling approaches, whether analytical or empirical, do not utilize the parametric values specific to a certain device location on a wafer. Variations of parametric values are typically represented statistically. Actually, the values are often treated as mutually independent random variables described by joint probability density functions [10, 11]. Once the statistical distribution is determined, the effect of the process/device variation on the device/circuit's performance is analyzed by performing simulations using Monte Carlo and other simulation techniques [12, 9, 13].

As shown in [14, 15, 16], many of these parametric variations do not occur in a random manner across a wafer but in a radial and/or axial pattern. These parameters should not be treated as uncorrelated mutually independent random variables [17, 18]. The approach presented below establishes a methodology in which a specific device's characteristics can be modeled based on its physical location within a wafer.

The initial focus of this research was to develop neural network models of material, process, and device characteristics at several critical stages of the GaAs IC fabrication process. This would allow capturing of the relationships between the various stages of the fabrication process, and between the process parameters and the resulting device parameters.

Measurements of the starting material and in-process device characteristics were used to develop neural network-based approximators of IC parameters at critical stages of the fabrication process (intermediate process stage modeling). Careful selection of architectures has been made to achieve proper network generalization from the training data sets available. Networks were trained using the generalized delta training rule, commonly known to be suitable for large data sets.

As a result of this research, the feasibility of utilizing neural networks to predict and increase the manufacturing yield of semiconductor devices, while reducing the test requirements, is established. These models developed for specific process stages help us evaluate predicted characteristics at the input to the next processing step. In-process measurements are also used to develop neural network models of yield-limiting characteristics measured after wafer fabrication is complete (final stage modeling). These models provide an effective tool for early parametric yield prediction, as well as process characterization of process and device parameters.

Fabrication data for this research comes from Material/Device Correlation Database (contract#/company name: F33615-88-C-174), cleared for public release on 7/14/1993. The data concerns a number of industrial production lines which fabricate GaAs MMIC. Measurements available characterize the starting materials and material and device parameters at such processing steps as ohmic, gate recess, gate metalization, and final DC. Characterizations include doping concentrations, layer thicknesses, planar geometries, layer-to-layer alignment, resistivities, and device voltages and currents. Although the results of this research are directly applicable to GaAs technology, the methodologies established are also valid for other IC technologies, and other applications such as chemical processes or other fabrication processes.

GaAs is a technologically important material because of its properties as a semiconductor material. This III-V compound semiconductor has several attractive properties. It has a very high electron mobility ($5000\,\mathrm{cm}^2/\mathrm{Vs}$) and high saturation velocity ($1.2 \cdot 10^7\,\mathrm{cm/s}$) giving it excellent high frequency performance. In addition, its large bandgap permits high temperature operation. Gallium arsenide substrates can be grown with very high resistivities (10^6–$10^8\,\Omega/\mathrm{cm}$). This high-resistivity substrate is used as a dielectric medium for the device isolation necessary in MMICs [19].

GaAs MMIC technology is multidisciplinary, encompassing material, device and circuit technologies, circuit design, and fabrication techniques. MMIC circuits are functions of a rather complex and multivariate manufacturing process and, therefore, it is difficult to properly model device and/or circuit performance.

Integrated circuit fabrication is expected to produce uniform material and device properties over a large wafer area. This uniformity is difficult to achieve for GaAs IC technol-

ogy because of material and process deviations. From wafer to wafer, as well as within a wafer, there are variations of material and process properties which strongly influence final device/circuit performance [20]. It is therefore essential that these fluctuations and relationships between various process stages are adequately modeled. However, modeling is not an easy task, requiring large-scale in-process testing followed by appropriate process identification.

The classic method for extracting the characteristics of semiconductor materials, processes, and devices is to collect data from microelectronic test structures [21, 22]. The data used in this research was generated as part of a government research program to study the correlation between GaAs materials, process, and device properties. The program employed a standard high-density test reticle (chip) for baseline wafer processing by industrial foundries. The test reticle provided a high resolution instrument with which to examine substrate quality and wafer processing control. Approximately 200 reticles were produced per wafer. A standardized set of on-wafer tests were performed on each reticle at different stages of the fabrication process. The test structures included an orthogonal array of MESFET pairs, parametric test patterns, and the MMIC standard Process Control Monitors (PCM).

Whole wafer testing was conducted on the substrates and during wafer processing at four critical steps: Ohmic or Post-contact, Post-recess, Post-gate, and Final. The majority of the characteristics were measured on the 0.5×200 micron MESFETs. This test structure/device (referred to as device from this point on) is at the center of our modeling effort.

2.1 Material and Fabrication Process Stages

A sequence of cross-sections illustrating the critical fabrication stages for the ion-implanted, recessed gate MESFET used in this work is shown in Figure 2. The fabrication process uses standard photolithography techniques. Material, process, and device characteristics are measured after each of these processing steps. A brief discussion of each of these process steps, provided below, is helpful to understand the full scope of measurements performed.

2.1.1 Substrate/Active Layer Stage

Fabrication of the MESFETs begins with a mechanically qualified semi-insulating GaAs substrate. Substrate and active layer characteristics measured at this stage, denoted S, include among others such parameters as optical scattering (OBS) and dislocation density (EL2), which facilitate prediction of saturated current non-uniformities [23].

The active layer is then formed by ion-implantation. This active layer is characterized by sheet conductance (Rsh), carrier concentration (Nd), and mobility measurements (Mu0, Mu1). Variations of these parameters arise, in part, because of strong radial and axial

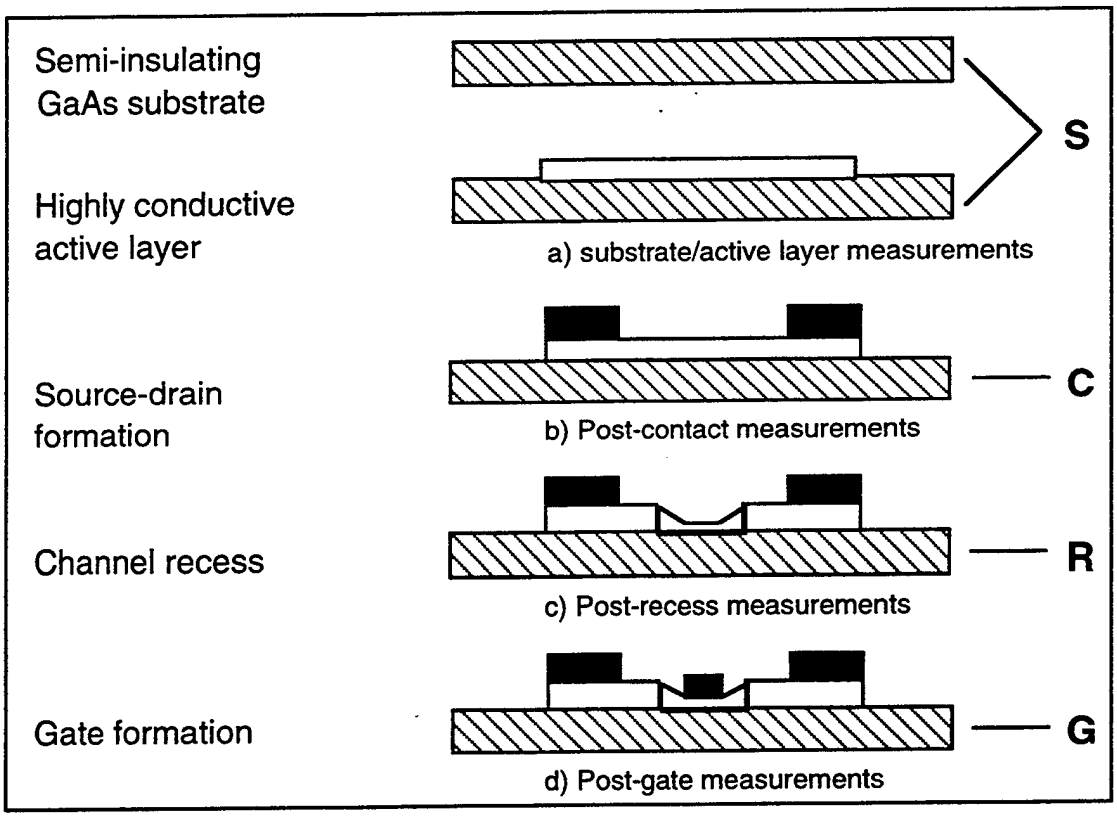


Figure 2: Critical MESFET fabrication stages at which characterization takes place: a) substrate/active layer; b) ohmic/post-contact; c) gate recess; d) gate formation. The MESFET geometry does not change from that at post-gate when the final stage measurements are taken.

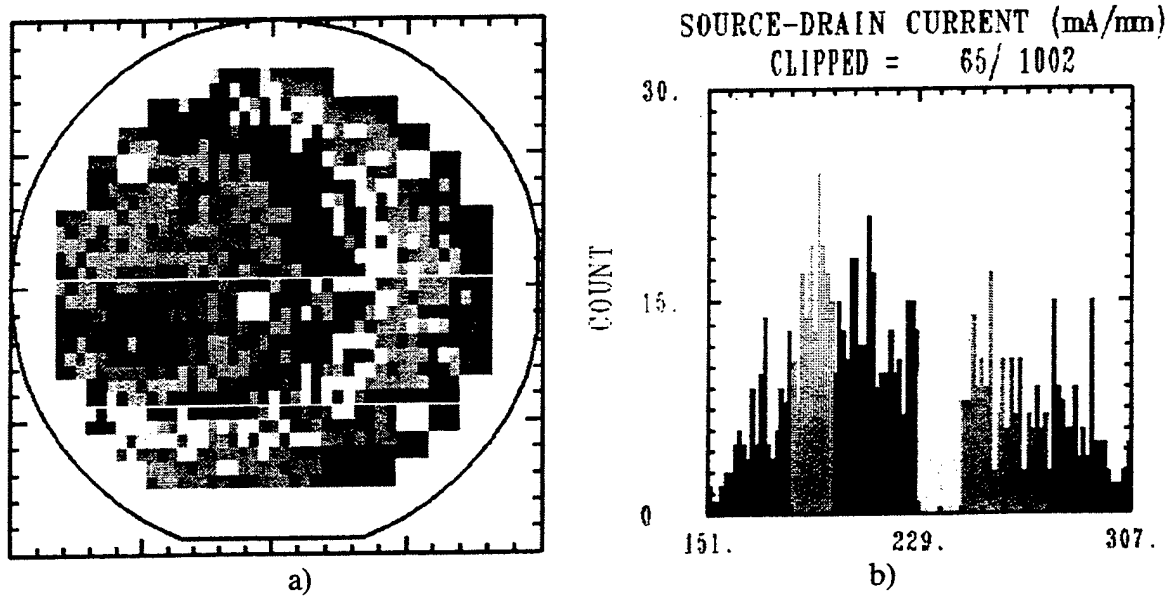


Figure 3: Visual representation of substrate electronic absorption, EL2: a) wafer map; b) histogram of measured EL2 values.

variations in thermal gradients during bulk crystal growth which affect local stoictriometry. Table 1 (entries 1–10) lists, among others, the S-stage characteristics which are measured at the beginning of the fabrication process and after the active layer is implanted.

Figure 3a shows a typical wafer map of the substrate electronic absorption characteristic which is a measure of the neutral deep donor density (EL2). This figure illustrates the radial variation exhibited by the EL2 values. Figure 3b illustrates the spread of EL2 values represented by the gray scales of the wafer map. This pattern is somewhat similar for many other characteristics making it important to properly model these variational effects on final device performance.

2.1.2 Ohmic/Contact Metal Stage

Ohmic metalization is next deposited during the ohmic contact stage, denoted C. These ohmic contacts, which form the FET source and drain, are attached to test probe pads. At this point, the wafer undergoes post-contact characterization. Each C-stage characteristics measured after the ohmic contact process stage is listed in Table 1 (rows 11–16).

C-stage measurements include the MESFET ungated drain-source saturation current (C-Idss), drain-source resistance (C-Rds), and the metalization width and resistivities. Mappings of C-Idss, such as that in Figure 4, show that most ion implanted active layers have characteristics which vary across the wafer. This typically is the result of the particular wafer tilt and rotation during the ion implantation [15].

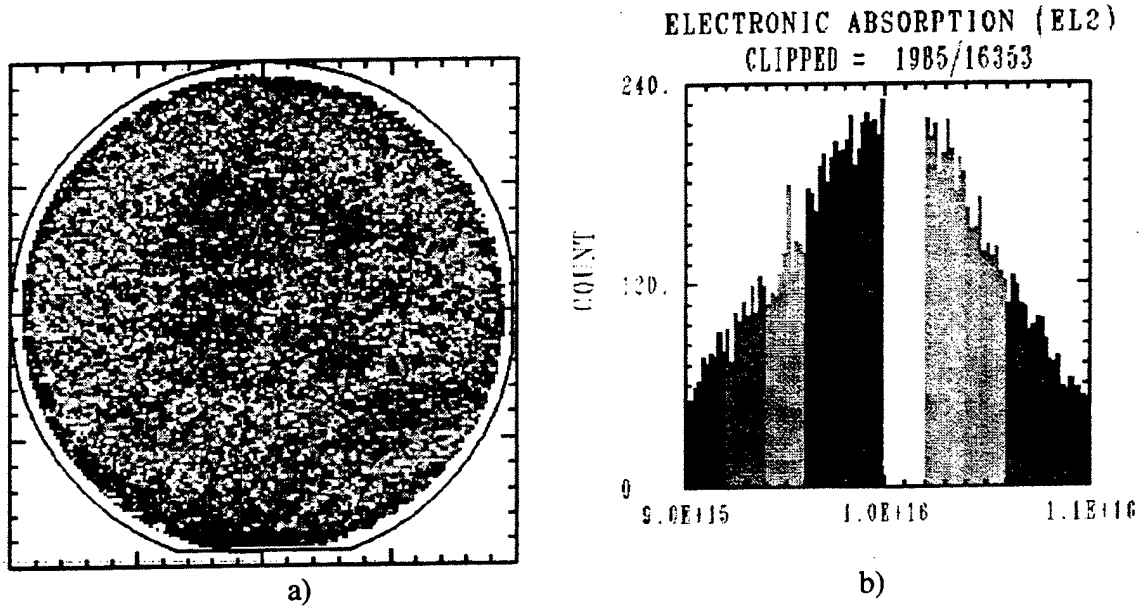


Figure 4: Visual representation of post-contact drain-source current, C-Idss: a) wafer map; b) histogram of measured C-Idss values.

2.1.3 Gate Recess Stage

The next fabrication step is the channel recess-etch process step, denoted R. This step is intended, in part, to be a fine-tuning adjustment to the FET current. The recess-etch process is a major contributor to wafer variations, as evidenced by post-recess characterization. The two R-stage characteristics taken after the gate-recess process step are also listed in Table 1 (rows 17–18).

2.1.4 Gate Metal Stage

After the gate recess-etch is complete, the gate metal is deposited on the patterned wafer. Data taken during the post-gate characterization, G, include the first measurement of complete FET characteristics (see again Table 1, rows 19–32). These characteristics describe FET symmetry and uniformity through parasitic resistances, drain-source current, and breakdown voltages. Non-uniformities may occur when the gate metal is deposited. Problems arise from, among other things, the photo-resist opening being too narrow, making the exact placement of the gate difficult. It is the variation present in these characteristics which makes it difficult to accurately model circuit performance.

2.1.5 Final Stage

After the gate metal is deposited, the wafer undergoes subsequent processing steps for making interconnections and IC passive components. These are first metal, dielectric deposition, silicon nitride passivation, and final plating metalization. Of all these steps, the most sig-

nificant is the silicon nitride passivation. In this process, a 2000 Å thick layer is deposited over the entire wafer and then removed by plasma etching from those areas not requiring nitride. Wafer passivation affects the FET breakdown voltages, drain current, pinch-off voltages, and transconductance. This phenomenon is neither fully understood nor has it been yet analytically modeled.

Upon completion of the fabrication process, the wafer undergoes the final DC characterization, F. Another complete set of FET characteristics are the measured together with several process control parameters, which are listed in Table 2.

2.2 High Density Test Reticle

The study of process-induced parametric variations requires an elaborate data collection effort. A large and representative number of measurements of process attributes, key device parameters, and layout geometries need to be taken during the fabrication process to provide a representative statistical database for analysis or interpretation. As a result of this effort, trends and correlations due to the fluctuations of the process can be modeled by analyzing the population of devices and their resulting electrical parameters.

The measurement data used for characterization originated from a 4×4.5 mm high-density test structure reticle (HDTR) repeated some 200 times per wafer. The reticle contains an array of the 0.5×200 micron MESFET, van der Pauw patterns, transmission line models, and standard process control monitor structures. In addition, the reticle has built-in redundancy in patterns and locations to reduce systematic and random measurement errors. Measurements taken on these test structures are used to study and model the impact of material and process variations on device/circuit performance.

2.2.1 Reticle Description

The high density material/process evaluation reticle, shown in Figure 5, was designed specifically to collect the data required for the in-depth analysis of parametric variations. Any detailed information about the reticle is restricted by the Arms Export Control Act and will not be included as part of this document. We can, however, say it contains a large number of test structures and is divided into three basic measurements: DC, RF, and process control monitors.

2.2.2 Characterization Data

In order to investigate the impact of various GaAs processing practices on the performance, uniformity of parameters, and production yield of MMICs, a comprehensive microelectronic test structure design must be in place. These test structures must be designed so that the

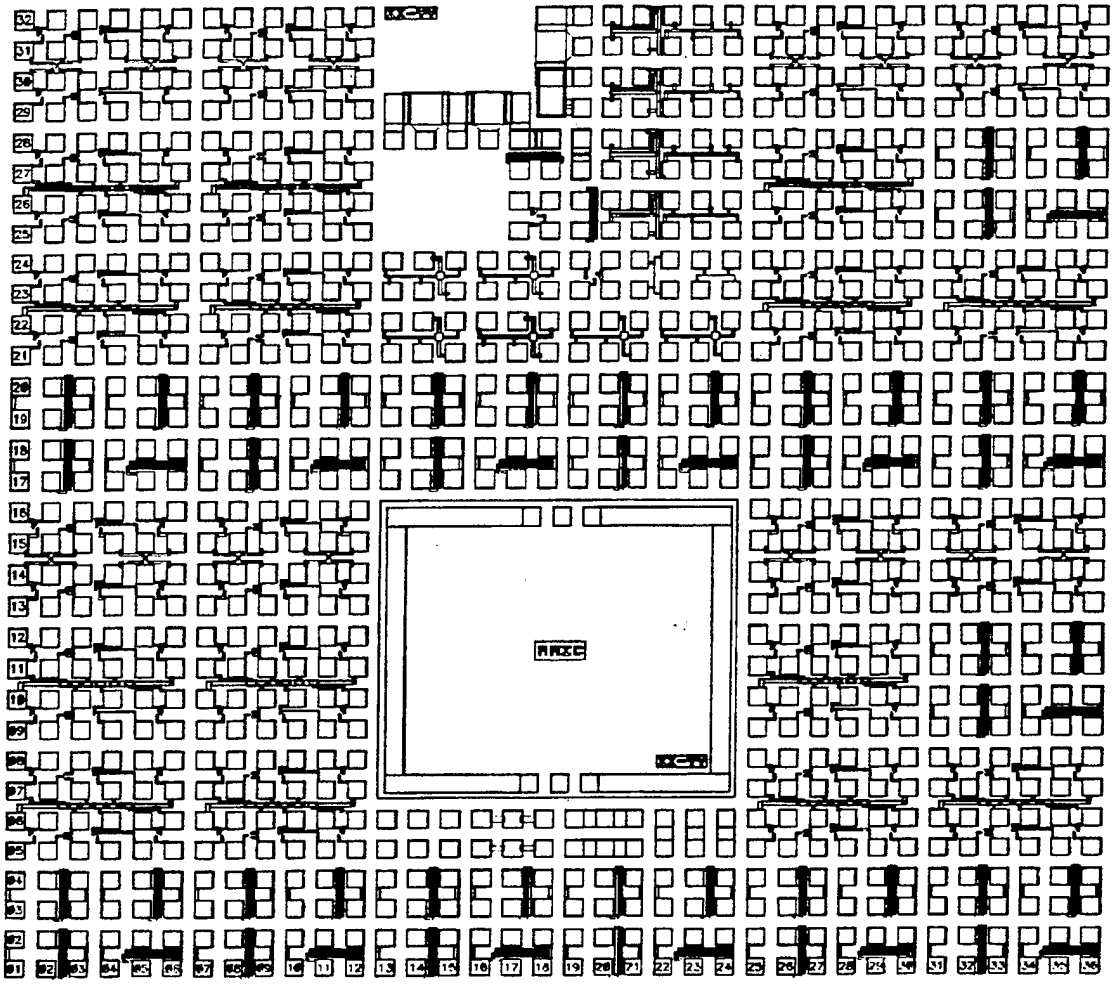


Figure 5: High-density test reticle, HDTR.

desired characteristic can be accurately measured using automated testing techniques.

The HDTR was designed to provide measurements of over 200 characteristics. Only 51 of these were used for neural network modeling, primarily because of the focus on the MESFET device. The characteristics selected are those which are believed to most directly affect MMIC performance. Each measurement was taken across the entire wafer at a sufficient density to fully characterize the wafer. This data is placed into two general categories: MESFET and Material/Process.

MESFET Characteristics As mentioned before, one of the objectives of this work was to model the effect that material and process variations have on the performance characteristics of active devices used in integrated circuits. The active device is typically where the impact of these variations become most evident since it is the major contributor to yield loss. Therefore, as mentioned earlier, the MESFET active device is at the center of

this modeling effort. The HDTR wafer is well populated with over 1000 MESFETs with measurements taken at a density of six per reticle.

Material/Process Characteristics The fabrication of MMICs involves many complex processing steps. The process includes the fabrication of active devices, resistors, capacitors, inductors, air-bridges, and via holes for ground connection. Process monitors and parametric test structures (PTS) capture variations resulting from these fabrication processes and related to the starting GaAs substrate, material, and each metal layer formed.

2.2.3 Identification Scheme

The characteristics were measured across the entire wafer in one automated test sweep. Figure 6 shows the common reticle overlay on a 3" wafer. Each square represents an HDTR. An identification scheme was developed to track the exact location of each test structure data point.

The test probe pads within the reticle shown on Figure 5, number from 1 to 36 in the x-direction and 1 to 32 in the y-direction, giving the exact test structure location. The reticle itself is numbered XXYY to identify its row and column location on the wafer (refer to Figure 6). This reticle layout and the reticle probe pad numbering scheme permit precise tracking of the location and identification of all test structures and data points. Measurements made at the different processing stages can be identified with the specific structure on which the measurement was made. This also allows material/process test structure characteristics to be linked with device characteristics within the reticle. This is important since many material/process characteristics vary across the wafer, from wafer-to-wafer and from lot-to-lot. The site-to-site linkage that this database provides is convenient for supervised training of feedforward neural networks.

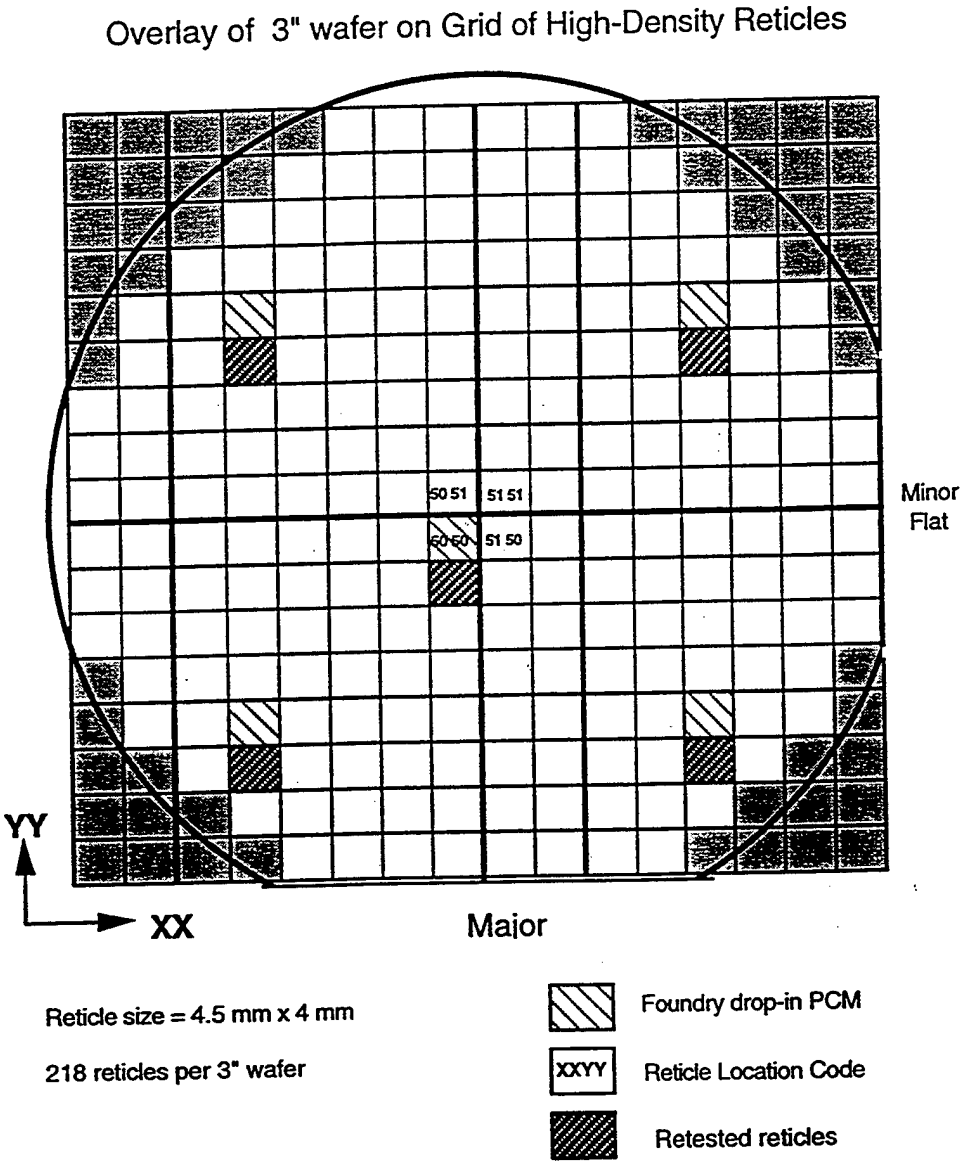


Figure 6: Overlay of a 3" wafer on a grid high-density test reticle illustrating the identification scheme.

3 Modeling of Process Stages

In the initial stage of work we focused on how to model the effect that material and process variations have on the performance characteristics of active devices used as components of integrated circuits. The performance parameters of active devices determine yield loss due to undesirable process variations. Therefore, the MESFET active device modeling and yield are at the center of the modeling effort.

In order to provide a statistical data set for neural network modeling of the process, we need a sufficient number of measurements of process attributes, device parameters, and layout geometries taken during the fabrication process. Data selection is initiated by choosing a representative data sample from a 4×4.5 mm HDTR repeated some 200 times per wafer. The reticle includes an array of 0.5×200 micron MESFETs, resistors, transmission line models, and other standard process control monitor structures. Moreover, there is a redundancy in the locations of the devices in order to reduce systematic and random measurement errors. For details of the data selection methods refer to [5].

A computer program denoted as Wtab (Wafer Table), developed during the initial phase of this work, was used to extract requested data from the database. See Appendix 8.1 for more details on Wtab. The following stages of the GaAs IC fabrication process were selected for device modeling: Substrate (S), Contact and Gate-Recess (GR), Gate metal (G), and Final (F). Training files were created from the master measurement data for each of the fabrication process stages S, CR, G, and F. The names of these data files, referred to as "characteristics", together with file identifiers, corresponding units, and brief descriptions are shown in Tables 1, 2, and 3. Selection of training files of a manageable size was desirable to train neural models in an efficient manner. Accordingly, a horizontal slice of 14 reticles across the middle of the wafer was chosen for training purposes. The 14 reticles at a density of 6 MESFETs per reticle provided 69 data vectors after discarding nonfunctional MESFETs, of which 50 were used to train neural networks and 19 were set aside for testing.

Yield estimation usually takes place after the final processing stage. The yield limiting device characteristics are compared to the target values, and the quality of the devices is then determined. Therefore, for design centering purposes, F-stage models need to be considered. Generally, parametric yield is found by determining whether the measured values of critical final performance parameters fall within a pre-determined tolerance range around the target value of each parameter. This can involve screening of F-stage DC device parameters, such as saturated drain current F-Idss, transconductance F-Gm and pinch-off voltage F-Vpo. Accurate estimation of parametric yield during the manufacturing process relies on the ability to predict the effect of material and process variations on device parameters. The neural network models described in the following sections allow for achieving this goal.

3.1 The SCRG-F Model

The computer program which implements the multilayer perceptron neural network, denoted as DES-PREP was developed as a part of the software package DESCENT. A user's manual and brief description of the program is included in Appendix 8.4. Architectures of neural networks used in this work were determined experimentally by varying the number of hidden neurons and selecting the number which resulted in the lowest testing error. By monitoring the learning profile on the testing data, the neural models were trained for their best generalization ability [24].

The SCRG-F network models the relationships between many characteristics measured throughout the entire IC fabrication processes S, CR, and G, and the final characteristics F. This model employs 32 inputs as listed in Table 1 and 19 outputs listed in Table 2. Twenty-two neurons were used in the network hidden layer. Many of the same characteristics measured at early stages are measured again at the final stage since these characteristics change in value as the fabrication process progresses.

Once the training is completed, the model was tested and evaluated for its ability to learn the mapping resulting from the underlying data. The modeled values were compared to the actual measurements in scattering plots as shown in Fig. 7a and 7b. All 19 output characteristics are shown in the same chart as various symbols listed in Table 2. Fig. 7a represents results of the training which was terminated as the testing error reached a minimum. This prevented the network from over-fitting to the training data, which would have affected the resulting model's generalization ability. Scattering plot of the testing data, shown in Fig. 7b, indicates a somewhat larger error (points distant from a diagonal line) as compared to scattering plot for the training data, which is typical for neural models. The testing error has been reduced by careful selection of the network architecture. Thorough evaluation of the Person product-moment correlation for the SCRG-F model is included in [5]. The resulting weights of the neural network for SCRG-F model is presented in Section 8.3 of the Appendix.

An alternative training method was also investigated to produce a SCRG-F model useful for design centering. The input SCRG data was first analyzed for principal components distribution and then linearly transformed prior to input to the neural network. The Principal Component Analysis (PCA) is described in more detail in Section 5.2. Results of this training method are shown in Fig. 8a and 8b in the form of scattering plots for comparison with the standard training technique. Comparing these results with those shown in Fig. 7 that models with the PCA input data preprocessing train to a lower error level. Other benefits from using the PCA in modeling will be evident later in Chapter 6.

	Characteristic	File identifier	Unit	Description
1	OBSA	s04sua		Substrate optical scattering angle A
2	EL2	s10sun	cm^{-2}	Substrate neutral deep donor density
3	OBSB	s04sub		Substrate optical scattering angle B
4	Rho	s05sun	Ω·cm	Substrate resistivity
5	MuH	s06sun	cm^2/V-cm	Substrate Hall mobility
6	ns	s07sun	cm^{-3}	Substrate carrier concentration
7	Nd	f44ffn	cm^{-3}	Peak doping concentration
8	ETA	f46ffn	%	Fatfet implant activation
9	Mu0	f49ffn	cm^2/V-cm	Drift mobility (Vg=0)
10	Mu1	f50ffn	cm^2/V-cm	Drift mobility (Vg=-1.5V)
11	C-Idss	c21rfv	mA/mm	Post-contact Idss
12	C-Rds	c22rfv	Ω·mm	Post-contact Rds
13	C-Rc	c41tln	Ω·mm	Contact resistivity
14	C-Rsh	c42tln	Ω/sq	Contact metal sheet resistance
15	O-Rsh	o42cbn	Ω/sq	Ohmic metal sheet resistance
16	O-W	o52cbn	μm	Ohmic metal layer width
17	R-Ids	r21rfv	mA/mm	Post-recess Ids
18	R-Rds	r22rfv	Ω·mm	Post-recess Rds
19	G-Idss	g21rfv	mA/mm	Post-gate Idss
20	G-Rds	g22rfv	Ω·mm	Post-gate Rds
21	G-Rgs	g23rfv	Ω·mm	Post-gate Rgs
22	G-Rs	g24rfv	Ω·mm	Post-gate Rs
23	G-Rdg	g25rfv	Ω·mm	Post-gate Rdg
24	G-Rd	g26rfv	Ω·mm	Post-gate Rd
25	G-Vbdg	g29rfv	V	Post-gate G-D breakdown voltage
26	G-Vbgs	g30rfv	V	Post-gate G-S breakdown voltage
27	G-Vpo	g31rfv	V	Post-gate pinch-off voltage
28	G-Gm	g32rfv	mS/mm	Post-gate transconductance
29	G-Ids-pk	g33rfv	mA/mm	Post-gate peak Ids
30	G-AL	g57eav	μm	Gate metal electrical alignment
31	G-Rsh	g42cbn	Ω/sq	Gate metal sheet resistance
32	G-W	g52cbn	μm	Gate layer width

Table 1: The SCRG characteristics list.

Characteristic	File identifier	Unit	Description	Symbol
F-Idss	f21rfv	mA/mm	Final DC Idss	◇
F-Rds	f22rfv	Ω·mm	Final DC Rds	+
F-Rgs	f23rfv	Ω·mm	Final DC Rgs	⊡
F-Rs	f24rfv	Ω·mm	Final DC Rs	×
F-Rdg	f25rfv	Ω·mm	Final DC Rdg	△
F-Rd	f26rfv	Ω·mm	Final DC Rd	✳
F-Vbdg	f29rfv	V	Final DC G-D breakdown voltage	◇
F-Vbgs	f30rfv	V	Final DC G-S breakdown voltage	+
F-Vpo	f31rfv	V	Final DC pinch-off voltage	⊡
F-Gm	f32rfv	mS/mm	Final DC transconductance	×
G-Ids-pk	f33rfv	mA/mm	Final DC peak Ids	△
Lg	f51lgv	μm	Gate length	✳
C	f55ctn	pF	MMIC capacitance	◇
i-Rsh	i42cbn	Ω/sq	Interconnect sheet resistance	+
i-W	i52cbn	μm	Interconnect metal layer width	⊡
p-Rsh	p42cbn	Ω/sq	Thin film sheet resistance	×
p-W	p52cbn	μm	Thin film layer width	△
BH	f54dsv		Barrier height	✳
BG	f58bgv		Vertical backgating percent change	◆

Table 2: The F characteristics list.

Characteristic	File identifier	Unit	Description	Symbol
OBSA	s04sua		Substrate optical scattering angle A	
EL2	s10sun	cm^{-2}	Substrate neutral deep donor density	
OBSB	s04sub		Substrate optical scattering angle B	
Rho	s05sun	Ω·cm	Substrate resistivity	
MuH	s06sun	cm^2/V-cm	Substrate Hall mobility	
ns	s07sun	cm^{-3}	Substrate carrier concentration	
Nd	f44ffn	cm^{-3}	Peak doping concentration	
ETA	f46ffn	%	Fatfet implant activation	
Mu0	f49ffn	cm^2/V-cm	Drift mobility (Vg=0)	
Mu1	f50ffn	cm^2/V-cm	Drift mobility (Vg=-1.5V)	
C-Idss	c21rfv	mA/mm	Post-contact Idss	
C-Rds	c22rfv	Ω·mm	Post-contact Rds	
C-Rc	c41tln	Ω·mm	Contact resistivity	
C-Rsh	c42tln	Ω/sq	Contact metal sheet resistance	
O-Rsh	o42cbn	Ω/sq	Ohmic metal sheet resistance	
O-W	o52cbn	μm	Ohmic metal layer width	
R-Ids	r21rfv	mA/mm	Post-recess Ids	
R-Rds	r22rfv	Ω·mm	Post-recess Rds	
G-Idss	g21rfv	mA/mm	Post-gate Idss	
G-Rds	g22rfv	Ω·mm	Post-gate Rds	
G-Rgs	g23rfv	Ω·mm	Post-gate Rgs	
G-Rs	g24rfv	Ω·mm	Post-gate Rs	
G-Rdg	g25rfv	Ω·mm	Post-gate Rdg	
G-Rd	g26rfv	Ω·mm	Post-gate Rd	
G-Vpo	g31rfv	V	Post-gate pinch-off voltage	
G-Gm	g32rfv	mS/mm	Post-gate transconductance	
F-Idss	f21rfv	mA/mm	Final DC Idss	$\diamond$
F-Rds	f22rfv	Ω·mm	Final DC Rds	$+$
F-Rgs	f23rfv	Ω·mm	Final DC Rgs	$\boxdot$
F-Rs	f24rfv	Ω·mm	Final DC Rs	$\times$
F-Rdg	f25rfv	Ω·mm	Final DC Rdg	$\triangle$
F-Rd	f26rfv	Ω·mm	Final DC Rd	$\divideontimes$
F-Vpo	f31rfv	V	Final DC pinch-off voltage	$\diamond$
F-Gm	f32rfv	mS/mm	Final DC transconductance	$+$

Table 3: The S, CR, G and F characteristics list.

3.2 The S-F, CR-F, and G-F Models

To gain an improved insight into these partial fabrication processes, three models of the F-stage DC characteristics were developed with each model having an input representing a different stage of the fabrication. The three models are used to compute the values of F-Idss, F-Gm, and F-Vpo, as well as the other F-stage characteristics as shown in Table 3. The model inputs, also listed in Table 3 as the first three parts, will be used for the process yield estimation. Models have 10, 8, and 8 inputs, for the S, CR, and G stages, respectively. Each of the models has 22 hidden neurons and 8 outputs each corresponding to the eight F characteristics. They were trained, tested, and evaluated in the same manner as the SCRG-F model.

Figures 9, 11, and 13, respectively, show scattering plots of training and testing results for these models. The PCA models were also prepared for these stages, and evaluation results are shown in Figs. 10, 12, and 14, respectively. As can be seen from comparisons of scattering plots, PCA pre-processed input data also yield better identification results for the S, CR, and G stages. Neurons with symmetric activation function tanh(net/2) were employed in the networks, and the neuron input, net, was evaluated with a bias unit equal to -1 [24]. Neural network weights developed for the S-F, CR-F, and G-F models are listed in Section 8.3 of the Appendix.

It can be noticed that among these three models, the single G-F stage model works with the lowest error for the testing data, and thus is considered as the best among the identified models.

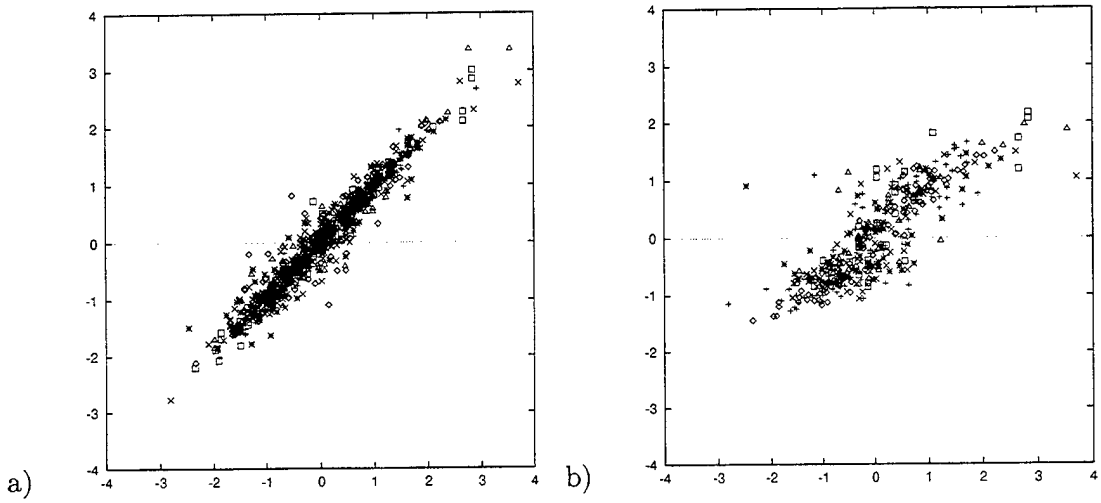


Figure 7: The SCRG-F non-reduced model: scattering plots for (a) training data, (b) testing data.

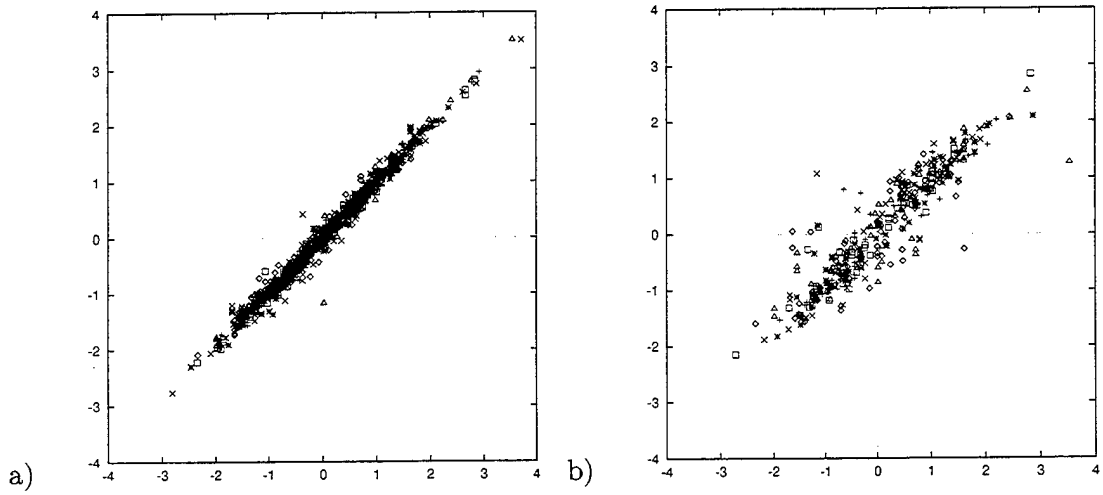


Figure 8: The SCRG-F non-reduced PCA pre-processed model: scattering plots for (a) training data, (b) testing data.

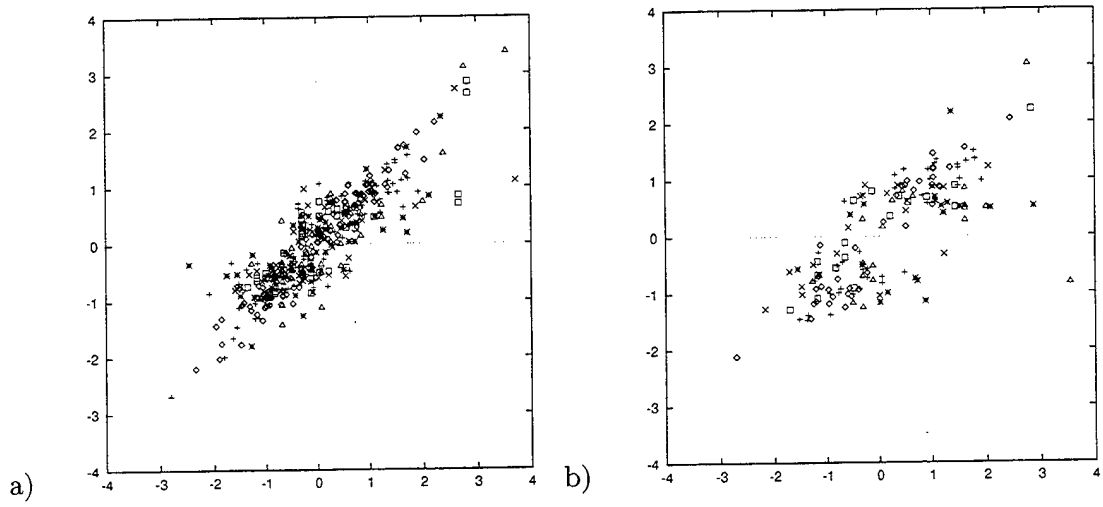


Figure 9: The S-F non-reduced model: scattering plots for (a) training data, (b) testing data.

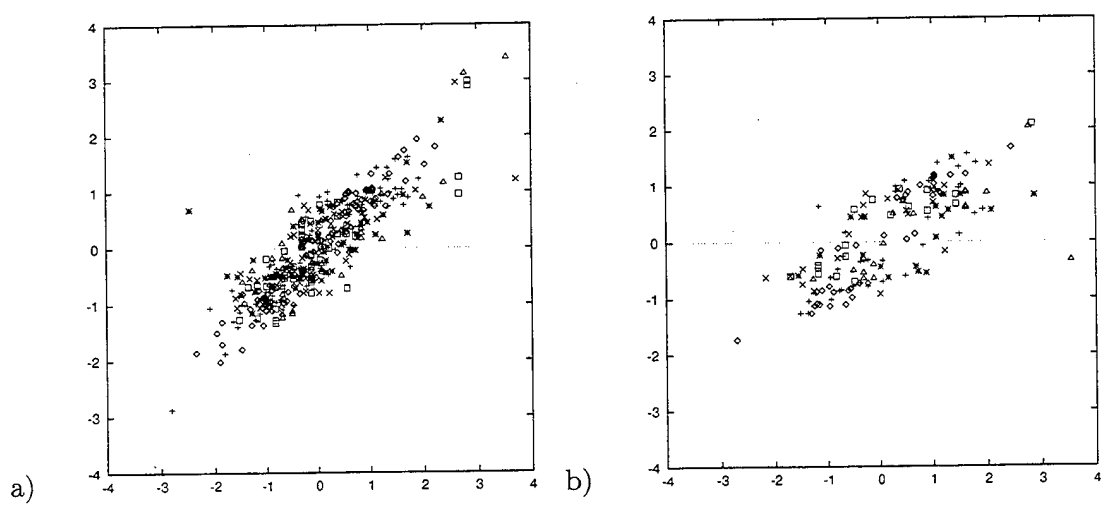


Figure 10: The S-F non-reduced PCA pre-processed model: scattering plots for (a) training data, (b) testing data.

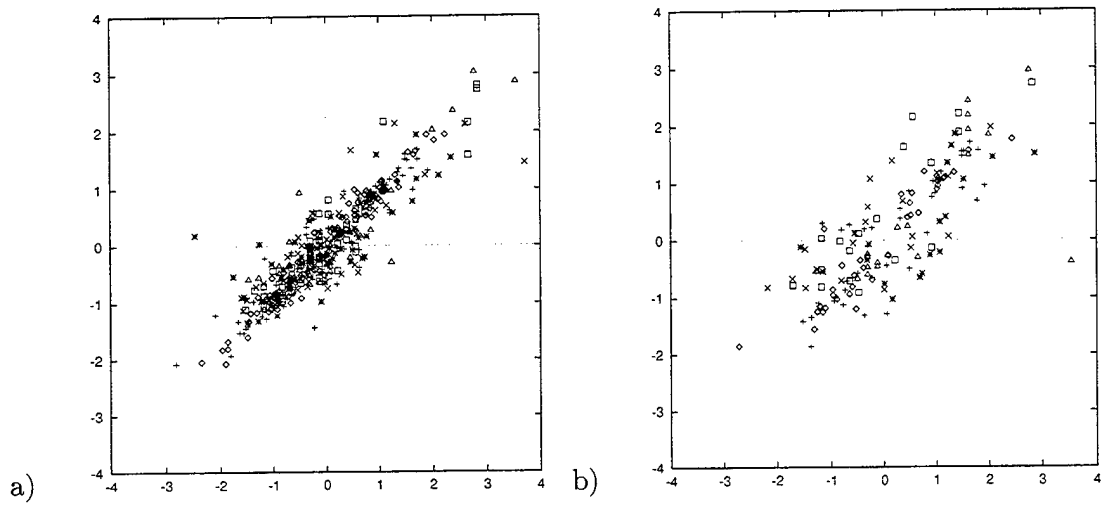


Figure 11: The CR-F non-reduced model: scattering plots for (a) training data, (b) testing data.

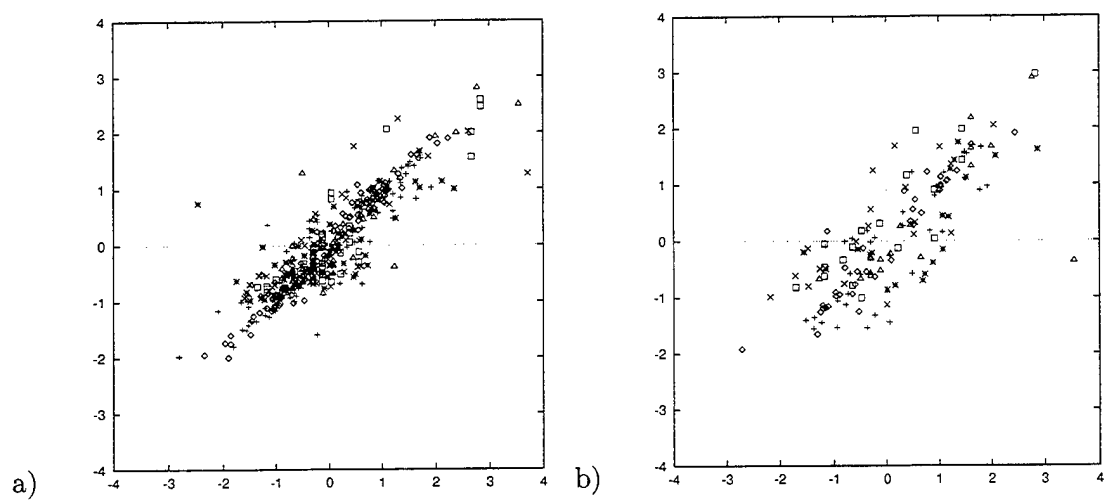


Figure 12: The CR-F non-reduced PCA pre-processed model: scattering plots for (a) training data, (b) testing data.

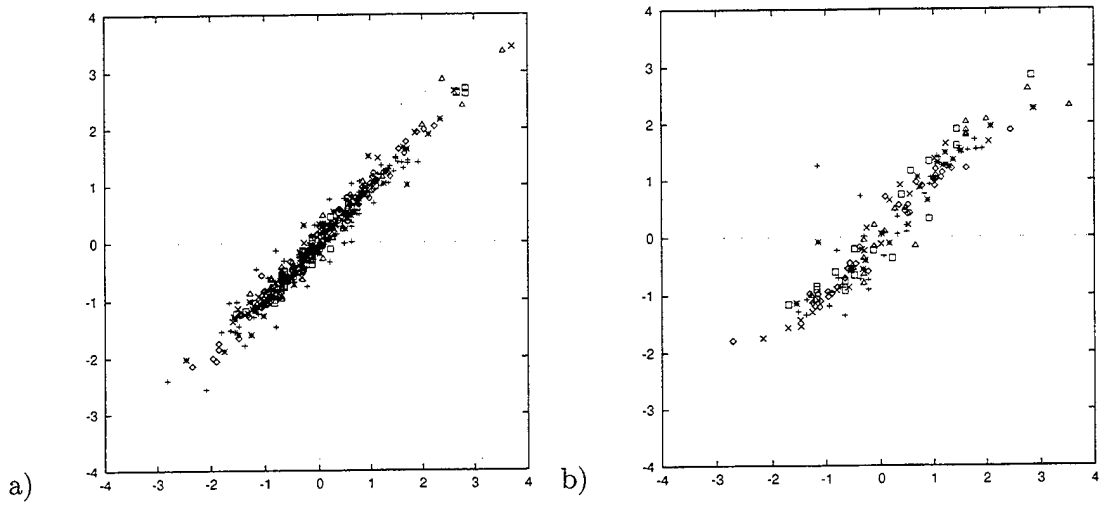


Figure 13: The G-F non-reduced model: scattering plots for (a) training data, (b) testing data.

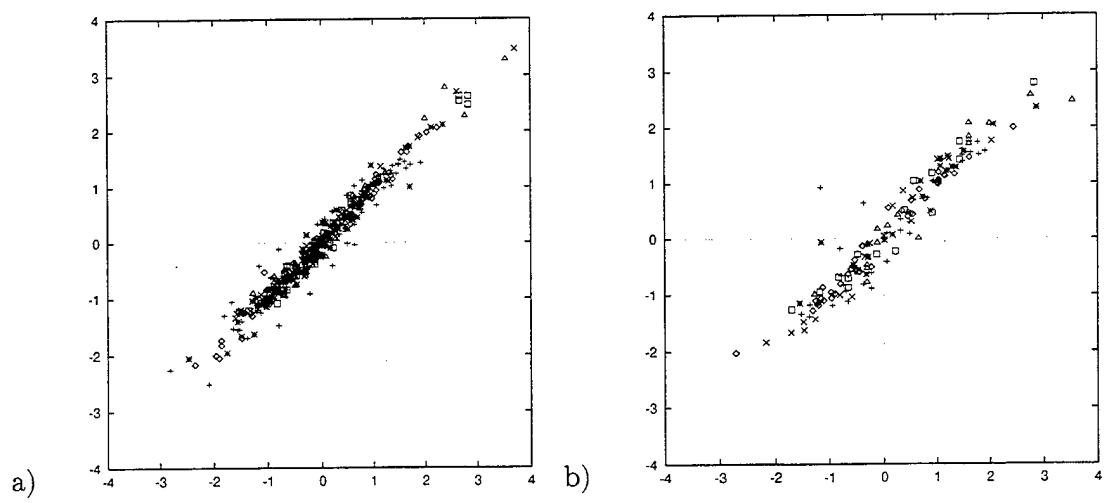


Figure 14: The G-F non-reduced PCA pre-processed model: scattering plots for (a) training data, (b) testing data.

4 Analysis of Sensitivity for Various Inputs

It is often very difficult to model the composite effect that material variations and fabrication process fluctuations have on the overall performance of the active device and/or circuits. Therefore, some form of factor analysis of the model becomes important for process and design optimization.

In part, factor analysis involves conducting a sensitivity analysis to determine which inputs of the model are most critical (sensitive) in determining the fabrication outcome. After identifying the sensitive parameters, special attention is given to the fabrication processes so that these parameter values may be better controlled. The advantage of sensitivity analysis over conventional modeling methods is the potentially useful information it makes available to the user. Monte Carlo methods can predict large variations in the output but will not provide insight as to why these variations occur. Sensitivity analysis specifies which output parameters are most affected by process fluctuations, process parameters, and/or process stage or stages for which they are most sensitive.

In addition, sensitivity analysis allows for the identification of irrelevant input parameters. This in turn may reduce the number of test parameters that will provide the quality of the modeling, thus leading to the possible elimination of certain test requirements [25].

Publications related to this project describe in more details techniques for performing sensitivity analyses on trained feedforward neural networks [26, 27, 28]. The sensitivities in question are computed by analyzing the total disturbance of network outputs due to perturbed inputs. They express the average norms of incremental output variations due to the disturbance of each input and indicate sensitive inputs.

Using the sensitivity analysis, sensitivity relationships are computed for the yield models obtained in this research. Specifically, a sensitivity analysis of the parametric yield models developed in Section 4.3 was performed. Additionally, sensitivity results were evaluated in an attempt to perform network pruning.

4.1 Sensitivity Calculation

The sensitivity of a trained MPNN output, o_k, with respect to its input, x_i, is defined in [26] as

$$S_{x_i}^{o_k} = \frac{\partial o_k}{\partial x_i} \tag{1}$$

which can also be written as

$$S_{ki} = S_{x_i}^{o_k} \tag{2}$$

By using a commonly adopted notation for the multilayer feedforward architecture [24], the derivative (1) can be expressed as

$$\frac{\partial o_k}{\partial x_i} = f'(\text{net}_k) \sum_{j=1}^{J-1} v_{kj} \frac{\partial y_j}{\partial x_i} \tag{3}$$

where y_j denotes the output of the j-th neuron in the hidden layer and $f'(\text{net}_k)$ is the value of the derivative of the activation function $o = f(\text{net})$ with respect to *net* at the k-th output neuron. Expanding (3) further yields

$$\frac{\partial o_k}{\partial x_i} = f'(\text{net}_k) \sum_{j=1}^{J-1} v_{kj} f'(\text{net}_j) w_{ji} \tag{4}$$

where $f'(\text{net}_j)$ is the value of the derivative of the activation function $y = f(\text{net})$ of the j-th hidden neuron ($y'_J = 0$ since the J-th neuron serves only as a bias input to the output layer). The $(K \times I)$ sensitivity matrix S, consisting of entries as in (4) is now expressed using matrix notation as

$$S = O'VY'W \tag{5}$$

where V $(K \times J)$ and W $(J \times I)$ are the output and hidden layer weight matrices, respectively, and O' $(K \times K)$ and Y' $(J \times J)$ are diagonal matrices defined as

$$O' = \text{diag}(o'_1, o'_2, \ldots, o'_K) \tag{6}$$

$$Y' = \text{diag}(y'_1, y'_2, \ldots, y'_J) \tag{7}$$

As can be seen from (5), the sensitivity matrix S depends not only on the network weights but on the slopes of the activation functions of all neurons as well. Therefore, each training vector $\boldsymbol{x}^{(p)}$ in the training set produces a different sensitivity matrix $S^{(p)}$. This is because although weights of a trained network remain constant, the activation values of neurons change across the set of training vectors. This produces different diagonal matrices O' and Y' which strongly depend upon the neuron operating points as determined by their activation values.

Since each training vector produces a different sensitivity matrix it is necessary to measure the sensitivity over the entire training set. This is accomplished in [27, 28] by computing the mean square average sensitivities defined as

$$\langle S_{ki} \rangle = \sqrt{\frac{1}{P} \sum_{p=1}^{P} (S_{ki}^{(p)})^2} \tag{8}$$

where P is the number of patterns in the training set.

The average sensitivity matrix entries defined in (8) provide useful information as to the importance of each input to the computation of each of the outputs for a known well trained feedforward neural network. A small value of $\langle S_{ki} \rangle$ in comparison to others means that for the particular k-th output of the network, the i-th input does not significantly contribute to output k. A high value of $\langle S_{ki} \rangle$ in comparison to the others means the i-th input does contribute significantly to output k. In order to distinguish between inputs with high and low importance the sensitivity measures derived from matrices $\langle S_{ki} \rangle$ relative to all outputs need to be sorted in descending order. Inspection of this ranking allows for the determination of inputs which affect the output least.

4.2 Sensitivity Analysis of Yield Models

The perturbation-based sensitivity analysis [28] was performed on each of the MPNN yield models developed and described in Chapter 3.2. The sensitivity measure, $\langle S_{ki} \rangle$, of each input for each output was computed over the training set for the S-F, CR-F, and G-F MPNN models.

4.2.1 S-F Model

The sensitivity measures $\langle S_{ki} \rangle$ for the S-F MPNN model were computed and are depicted in Fig. 15. The most significant input for all output characteristics is clearly Mu1, while Mu0 consistently ranks near the bottom. This result was not unexpected because the active layer mobility measurements were taken at different operating bias points, that is, Vgs = -1.5 V and Vgs = 0, respectively. The Mu1 mobility measurement test voltage is very close to the actual test bias voltage of the MESFET when the final stage characteristics are measured. Most of the output characteristics also exhibit high sensitivity to the inputs Nd and ETA. The peak doping density is a measure of the impurity concentration in the MESFET conductive channel formed during ion implantation. After ion implantation the wafer undergoes annealing to activate the implanted impurities. ETA is a measure of this impurity activation.

These results confirm practical clues known by process engineers. It has long been recognized that these characteristics are major factors, of equal importance, when considering final device performance. The S-F MPNN sensitivity measures confirm the importance of these characteristics and emphasize the necessity of taking the measurements to qualify the active layer material. As can be seen from the figure, none of the inputs to the S-F model exhibit negligibly small sensitivity measures across all outputs.

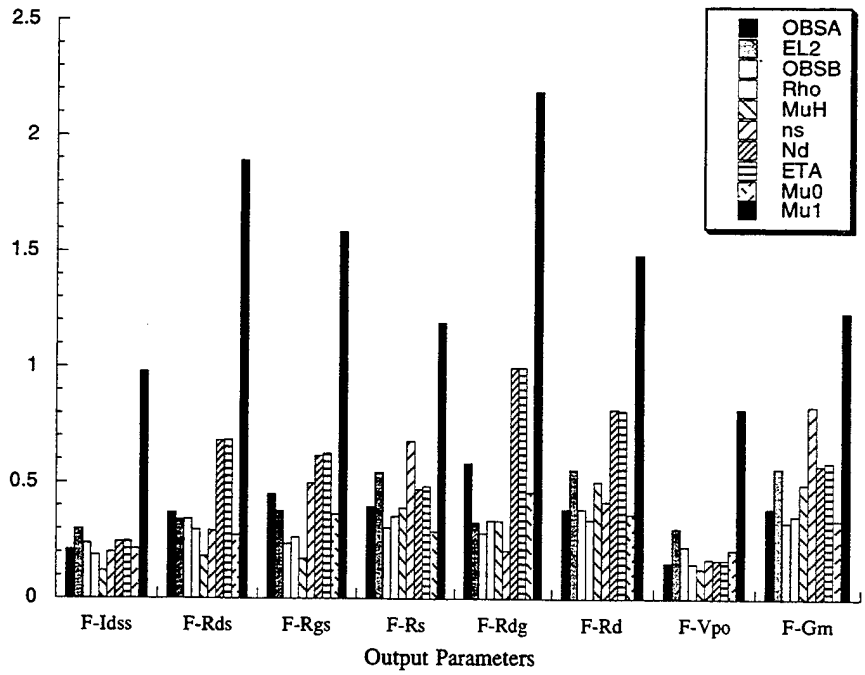


Figure 15: Sensitivity measure $\langle S_{ki} \rangle$ of each input for each output of the S-F model.

4.2.2 CR-F Model

Fig. 16 shows a bar chart showing the computed $\langle S_{ki} \rangle$ values for the CR-F model. As expected, the drain-source currents and resistances C-Idss, C-Rds, R-Idss and R-Rds are extremely important for the computation of the current related outputs (F-Idss, F-Rds, and F-Vpo), as is the ohmic metal sheet resistance, O-Rsh. It is unclear as to why the R-Idss and R-Rds inputs are not quite as important as C-Idss and C-Rds. It could be that the variations associated with the R-stage were not represented as well as the C-stage variations in the horizontal crossectional data used in training.

As expected, the materials-related characteristics contribute the most to MESFET parasitic resistances, although some also influence all output characteristics. For each given output there are relatively large differences among the individual inputs.

4.2.3 G-F Model

The G-F MPNN model's sensitivity measures are shown in Fig. 17. The current related outputs for the G-F model exhibit similarly high sensitivity measures as for the CR-F model. There is, however, a noticeable decrease in the sensitivity measures for the G-Vpo characteristic. As observed for the other models, the importance of each G-F input parameter varies from output to output. With the exception of G-Vpo, there are no input parameters that consistently rank very low in terms of the sensitivity measures for every output. This appears to be a good indicator that most of the inputs chosen for developing

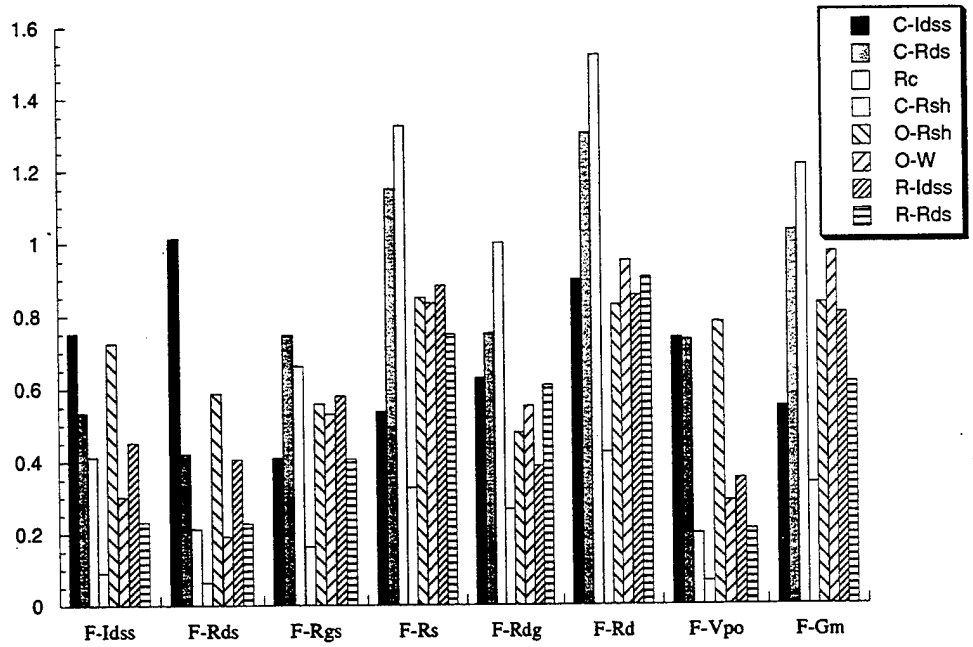


Figure 16: Sensitivity measure $\langle S_{ki} \rangle$ of each input for each output of the CR-F model.

process models do have some significance in the determination of the outputs. This also gives credibility to the design of the HDTR. The test structures and the characteristics they measure have significance in determining the outcome of the final MESFET characteristics.

4.3 Reduction of Test Requirements through Network Pruning

One of the goals associated with these sensitivity analyses is to identify input parameters which may be pruned from the input vector because of their lack of influence on the output parameters. The concept was to reduce cost by reducing the number of test parameters required to provide the level of characterization necessary to implement IC process modeling. In addition, deleting irrelevant data components would lead to smaller networks due to the reduced size of data vectors, thus resulting in computational savings.

Criteria for pruning input parameters were developed in the literature [28] along with the sensitivity approach used here. The criteria for determining which inputs can be pruned are based on the so-called 'gap' method. The sensitivity measures of the inputs for a specific output are ranked in sequence in descending order. The gap is defined as the ratio between two neighboring terms in the sequence. By examining these gaps across all outputs, a heuristic procedure can determine whether the gap associated with a given input is large enough to justify pruning [28].

As a result of evaluation of the computed gap ratios, it was determined that none of the inputs exhibited a gap large enough to fully justify input parameter pruning. This was

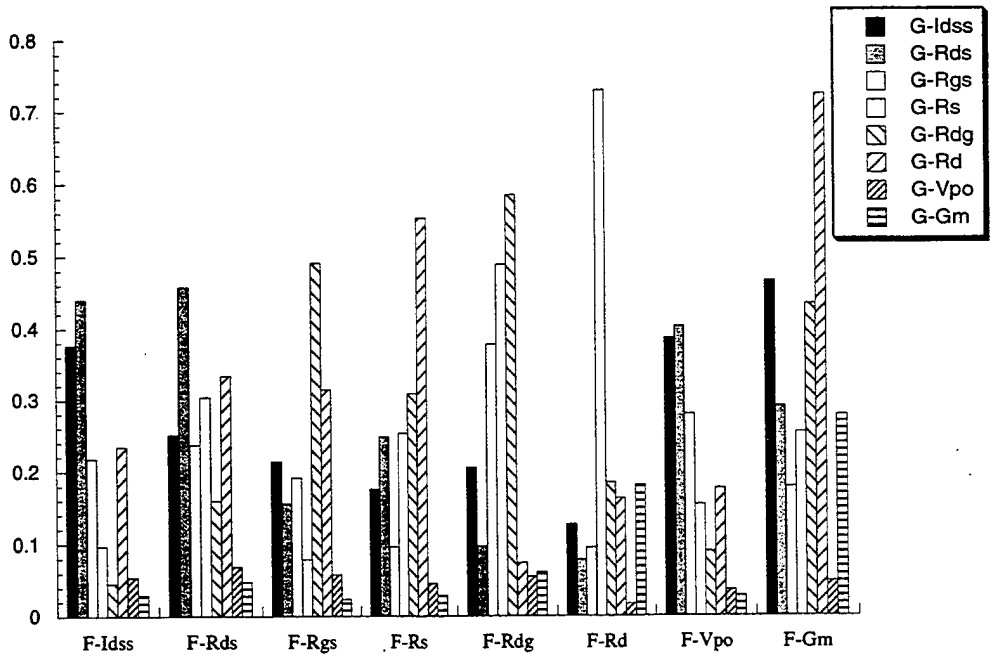


Figure 17: Sensitivity measure $\langle S_{ki} \rangle$ of each input for each output of the G-F model.

not surprising considering the manner in which the sensitivity measure of each input varied widely from output to output for each model.

4.4 Sensitivity Analysis for Yield Enhancement

So far the discussion has primarily centered on whether or not a particular input made a significant contribution to a modeled output. When considering sensitivity analysis for yield enhancement, a given output is examined to determine the input parameters for which it is most sensitive.

Design engineers routinely use sensitivity analysis to check the robustness of a design to selected features of fabrication processing. Circuit elements that are likely to be critical to production yields are randomly altered to study the variational impact on final circuit performance. If a foundry's tolerance is 5% for a specific parameter, the designer will vary the parameter 5% to estimate the impact on yield. To fully exploit sensitivity analysis, all of the parameters in a design should be examined. The full analysis becomes tedious, however, and a standard automated method is required.

Multivariate treatment of statistical analysis is practical provided two simplifying assumptions are made: variables are statistically uncorrelated, and variables obey Gaussian distributions. As discussed earlier, the first assumption is not fulfilled for FET parameters which do, for physical reasons, show correlations.

5 Design Centering and Yield Maximization Approach

As discussed in preceding chapters, VLSI microfabrication process can be described by its input and output characteristics that are captured in available measurement data. Each data point reveals the input/output relationship resulting from the material and the technology. This allows for fabrication process identification. From the viewpoint of analysis, measurement data collected at various locations of a wafer are regarded probabilistically as random events and are characterized by the respective input and output variable distributions. Thus, let $\boldsymbol{x}$ and $\boldsymbol{y}$ be the input and output random variables in the form of an n-vector and m-vector, respectively, with the assumption that there are n input characteristics and m output characteristics for the given stage.

The relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$ can be formally expressed as a function μ that maps input characteristics data into the output characteristics data:

$$\boldsymbol{y} = \mu(\boldsymbol{x}) \tag{9}$$

A center value $\boldsymbol{x}_c$ is to be maintained at the input in order to achieve a specific, required output (target value) $\boldsymbol{y}_0$ as a result of the fabrication stage process. However, due to the random distribution of the fabrication factors, equipment imperfection and fluctuation of process settings and material properties, the actual $\boldsymbol{x}$ is typically randomly distributed around $\boldsymbol{x}_c$. When many input factors are involved and many fabrication cases considered, the random spread can be approximated by a Gaussian distribution with the mean $\boldsymbol{x}_c$ and covariance matrix C_x. The input distribution thus reads

$$p(\boldsymbol{x}, \boldsymbol{x}_c) = N(\boldsymbol{x}_c, C_x) \tag{10}$$

Here, $p(\boldsymbol{x}, \boldsymbol{x}_c)$ represents the actual distribution of input values $\boldsymbol{x}$ when attempting to maintain the center value $\boldsymbol{x}_c$. Entries in vector $\boldsymbol{x}$ are typically expected to correlate to some extent with each other. This is manifested by non-zero off-diagonal entries in the covariance matrix C_x. Moreover, the technology-related spread of characteristics is assumed to be beyond control. Only the center input value $\boldsymbol{x}_c$ can be set when targeting at the desired output $\boldsymbol{y}_0$.

The goal of design centering in the fabrication process is to maximize the final product yield by choosing optimum settings of the input parameters. The output characteristics are then expected to produce a given target value and the largest number of manufactured products which fit within the tolerance limits. Assume that the product is acceptable if the target output value $\boldsymbol{y}_0$ is manufactured with tolerance δ_y. Define the target set ω_y as follows:

$$\omega_y = \{\boldsymbol{y} : y_{i\min} \leq y_i \leq y_{i\max}\} \tag{11}$$

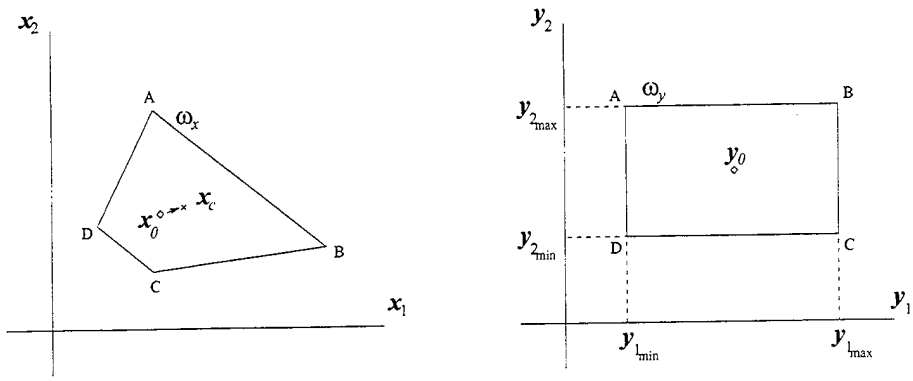


Figure 18: Projection of a target and tolerance region into the input space of the model.

Thus, each output y_i must belong to the region bounded by $y_{i\min} = (1 - \delta_{y_i})y_0$ and $y_{i\max} = (1 + \delta_{y_i})y_0$. The concept of the output tolerance region ω_y is roughly illustrated in Fig. 18 for $n = m = 2$. Formally, the process yield can now be characterized by the probability $\Pr(\boldsymbol{y} \in \omega_y)$. Since this probability is to be maximized and the only input parameter that can be controlled is the center input value $\boldsymbol{x}_c$ (assuming there is no control over input factors distributions), the definition of the design centering task now takes the following form:

$$\max_{\boldsymbol{x}_c} \Pr(\boldsymbol{y} \in \omega_y) \tag{12}$$

Equation (12) provides a functional for optimization. Maximizing this functional is equivalent to maximizing the process yield. Note that input and output distributions are related through equation (9) as determined through experimental data. The optimization at the input side starts at point $\boldsymbol{x}_0$ (see Fig. 18) which satisfies $\mu(\boldsymbol{x}_0) = \boldsymbol{y}_0$. The result of this process is a final point $\boldsymbol{x}_c$ which fulfills expression (12). Since generally $\mu(\boldsymbol{x}_c) \neq \boldsymbol{y}_0$ due to the nonlinearity of function μ, maximizing the functional (12) requires a multi-step computational solution for a multidimensional inverse problem.

5.1 The Approach

Typically, when creating a model of a stage, many measurements are taken of relevant process factors or characteristics. Many of the inspected characteristics are often related to each other due to their mutual correlations. Also, design centering is essentially an optimization process. It will involve all of these input factors and their mutual dependencies. Since optimization in a multidimensional space is both difficult and time consuming, especially when nonlinear process models are involved, the optimization space dimension is first reduced by using the mutual correlations. By reducing the input space dimensionality, opti-

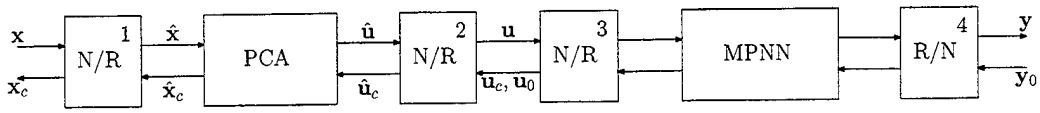


Figure 19: Modeling the microfabrication stage. "N" denotes normalization step, whereas "R" denotes denormalization.

mization algorithms can be used more efficiently while computational complexity is reduced to a lower level. The approach presented below has been first introduced in [29, 30].

The entire fabrication process model can be assumed to consist of two components: PCA and the neural network modeling through MPNN. Knowing both components, relationships can be calculated in both directions, i.e., from the input to the output and from the output to the input. Thus four separate operations are required to solve the design centering problem. The intermediate variable $\boldsymbol{u}$, referred to as an "abstract variable", represents the normalized and compressed space in which the design centering will be implemented.

The detailed overall block diagram of the fabrication stage which corresponds to the approach detailed in the paragraph above is shown in Fig. 19. The figure is an expanded version of Fig. 1. In the forward direction, the output sample $\boldsymbol{u}$ is to be computed from input data sample $\boldsymbol{x}$ by using the PCA operator that projects $\boldsymbol{x}$ into $\boldsymbol{u}$, and followed by the neural network mapping $\boldsymbol{u} \rightarrow \boldsymbol{y}$. Given the desired output value $\boldsymbol{y}_0$, the corresponding variable $\boldsymbol{u}_0$ (if it exists) can be computed by an iterative search for the solution using the inverse of the neural network mapping [31, 32]. Subsequently, the corresponding input $\boldsymbol{x}$ can be found by using the inverse PCA operator.

5.2 Principal Component Analysis

Principal Component Analysis (PCA) of input characteristic measurements is primarily needed in order to reduce the model input dimensionality [33]. As indicated in Fig. 19, the original input data $\boldsymbol{x}$ is transformed into $\boldsymbol{u}$ by PCA which is both preceded and terminated by normalization stages marked "N". The input normalization N1 of $\boldsymbol{x}$ is necessary to unbias the raw input data and balance their scaling. After this step all inputs have zero mean and unit variance. The PCA then changes basis vectors for input variable representation, and typically reduces dimension from n to m, where $m < n$. Also, the transformed data $\boldsymbol{u}$ becomes uncorrelated as a result of the PCA. Additionally, another normalization denoted as N2 equalizes the variance of each variable. The resulting data representation, referred to as $\boldsymbol{u}$, is a random variable composed of m entries u_i which have zero crosscorrelation and their variances are equal to 1. This property substantially simplifies the design centering task described below. The following is the analytical description of variable transformation $\boldsymbol{x} \rightarrow \boldsymbol{u}$.

The input data is characterized by means $\langle x_i \rangle$ and standard deviations σ_{xi}. Prior to the PCA, normalized input $\hat{\boldsymbol{x}}$ is calculated at N1 using the following equation:

$$\hat{x}_i = \frac{x_i - \langle x_i \rangle}{\sigma_{xi}} \tag{13}$$

The resulting variable $\hat{\boldsymbol{x}}$ now has zero mean and unit variance at each component x_i. Subsequently, the autocorrelation matrix R is calculated as follows:

$$R = \langle \hat{\boldsymbol{x}} \hat{\boldsymbol{x}}^T \rangle \tag{14}$$

In order to compute the PCA operator, the eigenvectors of matrix R must first be found. Let $\boldsymbol{v}_k$ be an eigenvector of matrix R, and λ_k its corresponding k-th eigenvalue such that they yield the equation:

$$R\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k, \qquad k = 1, \ldots, n \tag{15}$$

Additionally, let eigenvectors $\boldsymbol{v}_k$ be orthonormal so the norm $\boldsymbol{v}_k^T \boldsymbol{v}_k = 1$ for each of the eigenvectors. It is also beneficial to introduce a descending order of eigenvalues, such that $\lambda_k \geq \lambda_{k+1}$.

Eigenvectors $\boldsymbol{v}_k$ span a new basis for the input data representation. A PCA operator matrix M will now be defined to transform input $\hat{\boldsymbol{x}}$ into its projection $\hat{\boldsymbol{u}}$ in the new basis. Grouping the first m eigenvectors with the largest eigenvalues in a rectangular $m \times n$ matrix M yields:

$$M = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m]^T \tag{16}$$

which has the property that $MM^T = I$. Matrix M will be used below as the PCA operator which transforms input $\hat{\boldsymbol{x}}$ into vector $\hat{\boldsymbol{u}}$:

$$\hat{\boldsymbol{u}} = M\hat{\boldsymbol{x}} \tag{17}$$

The new data points $\hat{\boldsymbol{u}}$ belong to an m-dimensional space which is reduced as compared to the original input space. In addition, data points $\hat{\boldsymbol{u}}$ are now uncorrelated, which can be expressed by $\langle \hat{\boldsymbol{u}} \hat{\boldsymbol{u}}^T \rangle = \Lambda$, where Λ is a diagonal matrix with entries λ_k, $k = 1, \ldots, m$ on the diagonal. In other words $\langle \hat{u}_k \hat{u}_l^T \rangle = \lambda_k$ if $k = l$ and $\langle \hat{u}_k \hat{u}_l^T \rangle = 0$ if $k \neq l$. This means that $\hat{\boldsymbol{u}}$ belongs to the m-dimensional distribution and λ_k is a variance of the k-th variable in this distribution. Note that λ_k is also a variance of the data points $\hat{\boldsymbol{x}}$ projected onto the direction of the eigenvector $\boldsymbol{v}_k$ which represents the k-th principal direction of the input data distribution.

Since entries $\hat{u}_k$ are typically characterized by different variances, another normalization step can simplify the data analysis. Denote the normalized data points by $\boldsymbol{u}$. The

normalization in this step (N2) is simple and reads:

$$u_k = \frac{1}{\sqrt{\lambda_k}} \hat{u}_k \tag{18}$$

In summary, by utilizing equations (13), (17), and (18), each input data point $\boldsymbol{x}$ can be transformed into point $\boldsymbol{u}$ in the new, reduced space. The new data representation has the property $\langle \boldsymbol{u}\boldsymbol{u}^T \rangle = I$, making it suitable for design centering algorithms. Each point $\boldsymbol{u}$ can be inversely transformed to the original input space with a controlled degree of accuracy depending on the dimension m. Let B be the inverse PCA transformation operator

$$B = M^T \tag{19}$$

Due to the dimension reduction performed by the operator M in (17), point $\hat{\boldsymbol{x}}$ and the inversely obtained point $BM\hat{\boldsymbol{x}}$ are not identical if $m < n$. Let us define an error of data representation in the reduced space related to the model input as a difference between the original point $\hat{\boldsymbol{x}}$ and its representation:

$$\boldsymbol{e} = \hat{\boldsymbol{x}} - BM\hat{\boldsymbol{x}} \tag{20}$$

It may be shown [34] that the average squared error $||\boldsymbol{e}||^2$ equals the sum of all eigenvalues associated with the eigenvectors not included in the PCA operator matrix M:

$$||\boldsymbol{e}||^2 = \langle \boldsymbol{e}^T \boldsymbol{e} \rangle = \sum_{k=m+1}^{n} \lambda_k \tag{21}$$

The error norm $||\boldsymbol{e}||^2$ as in (21) can be used in computing and controlling the effective new dimension m of the data in the transformed $\boldsymbol{u}$ space. After scaling with respect to the largest eigenvalue, the error can be considered as a percentage of the maximum variance λ_1 that the input data has along the distribution's principal direction:

$$\Delta_{\%} = \frac{1}{\lambda_1} \sum_{k=m+1}^{n} \lambda_k \tag{22}$$

Note that this error is related to the PCA only and is a part of the total error of the fabrication process model.

5.3 Inverse Projection through Neural Model

As mentioned, the mapping $\boldsymbol{x} \rightarrow \boldsymbol{y}$ represents the process and is generally a continuous nonlinear vector function. The PCA component, however, of the entire model is a linear transformation. Hence, a function approximator has to be used to complete the task of modeling the fabrication process. An MPNN [24] is used for this purpose. Additional

normalization and renormalization steps need to be done at the network input and output to enable the network to learn the stage characteristics. Classic error backpropagation training of a two-layer architecture has been found sufficient to train the neural network.

For the sake of finding an input $\boldsymbol{u}_0$ given the target output $\boldsymbol{y}_0$ through the neural model, the algorithm introduced in [31] will be used. Define the solution error E as a norm:

$$E = ||\boldsymbol{y} - \boldsymbol{y}_0||^2 \tag{23}$$

The error gradient $dE/d\boldsymbol{u}$ will enable an iterative search in the $\boldsymbol{u}$ space for a solution corresponding to the desired output $\boldsymbol{y}_0$. The gradient entries read:

$$\frac{\partial E}{\partial u_k} = \sum_i \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial u_k}, \qquad k = 1, \ldots, m \tag{24}$$

Using (24) $\boldsymbol{u}$ can be evaluated iteratively according to the steepest descent method:

$$u_i' = u_i - \kappa \frac{\partial E}{\partial u_i} \tag{25}$$

where constant $\kappa > 0$ controls the algorithm convergence rate [35]. Using this approach we can iteratively find the $\boldsymbol{u} = \boldsymbol{u}_0$ variable corresponding to the target output $\boldsymbol{y}_0$.

5.4 Optimization Algorithm

The solution to expression (12) will be searched for in u-coordinates since they represent an orthonormalized space for the input data distribution with a reduced dimension. Region ω_y represents all acceptable output variable values resulting both from the tolerance and target point requirements as defined in (11). Define region ω_u such that implication $(\boldsymbol{u} \in \omega_u) \Rightarrow (\boldsymbol{y} \in \omega_y)$ is valid. In other words all the points $\boldsymbol{u}$ which belong to the region ω_u will result in acceptable output values $\boldsymbol{y} = f(\boldsymbol{u})$. Note that the output space dimension is greater than m–therefore the inverse implication does not necessarily hold true. Under the assumptions made, the following probabilities are equal:

$$\Pr(\boldsymbol{y} \in \omega_y) = \Pr(\boldsymbol{u} \in \omega_u) \tag{26}$$

Since the variable u space is orthonormalized, the data points distribution can now be represented by a symmetric m-dimensional Gaussian $p(\boldsymbol{u}, \boldsymbol{u}_c)$ centered at some $\boldsymbol{u}_c$ that will be searched for during the optimization process:

$$p(\boldsymbol{u}, \boldsymbol{u}_c) = N(\boldsymbol{u}_c, \sigma) = \frac{1}{(\sqrt{2\pi}\sigma)^m} e^{-\frac{\|\boldsymbol{u}-\boldsymbol{u}_c\|^2}{2\sigma^2}} \tag{27}$$

Here σ equals 1 and is used later for further purposes, and $\|\boldsymbol{u} - \boldsymbol{u}_c\|$ is a norm of a distance between the variable $\boldsymbol{u}$ and the center point $\boldsymbol{u}_c$. The design centering will provide

some value for $\boldsymbol{u}_c$ that will be considered as a solution $\boldsymbol{x}_c$ when transformed back into the input space.

Denote the yield probability as p_c. In the u-space it can be described by the integral:

$$p_c = \Pr(\boldsymbol{u} \in \omega_u) = \int_{\omega_u} p(\boldsymbol{u}, \boldsymbol{u}_c) du \tag{28}$$

Now assume that the space is uniformly covered by random points $\boldsymbol{u}_k$, as shown in Fig. 20. The point neighborhoods s_k combined together fill the entire space. The points belonging to the region ω_u create set δ such that the volume of ω_u equals $V_{\omega_u} = \sum_{k\in\delta} s_k$. If the number of points is sufficiently large, the probability p_c can be approximated by the following sum:

$$p_c = \sum_{k\in\sigma} s_k p(\boldsymbol{u}_k, \boldsymbol{u}_c) \tag{29}$$

The goal of design centering now becomes equivalent to maximizing probability p_c by moving the center point $\boldsymbol{u}_c$ such that (30) is fulfilled

$$\max_{\boldsymbol{u}_c} p_c \tag{30}$$

The solution to (30) is a point $\boldsymbol{u}_c^*$ that could be found as a result of an optimization algorithm with the functional p_c. Define the gradient of p_c that will be useful for this algorithm:

$$\frac{dp_c}{d\boldsymbol{u}_c} = \sum_{k\in\delta} s_k p(\boldsymbol{u}_k, \boldsymbol{u}_c)(-\frac{1}{2\sigma^2})\frac{d}{d\boldsymbol{u}_c}\|\boldsymbol{u}_k - \boldsymbol{u}_c\|^2 \tag{31}$$

Gradient (31) indicates the direction toward which the center point $\boldsymbol{u}_c$ should be moved in order to increase the yield probability p_c. At the solution $\boldsymbol{u}_c^*$ the gradient is zero:

$$\left.\frac{dp_c}{d\boldsymbol{u}_c}\right|_{\boldsymbol{u}_c=\boldsymbol{u}_c^*} = \vec{0} \tag{32}$$

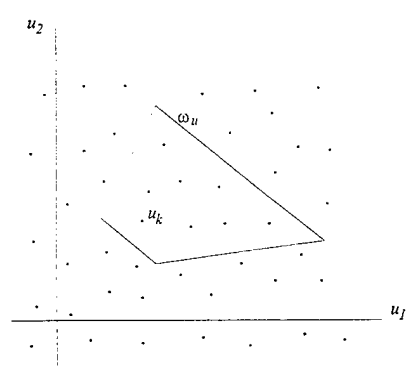


Figure 20: Approximating the yield probability.

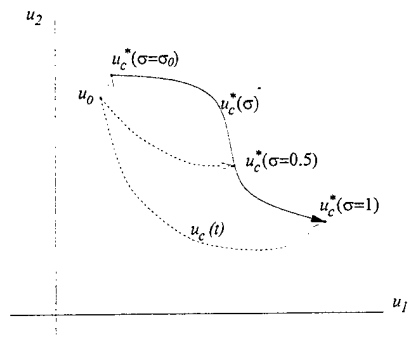


Figure 21: Movement of the solution with respect to σ.

Generally, this gradient can be zero at more than one point; however, each of these points may or may not represent the global solution of maximum probability p_c. Assume that the optimization algorithm is used in the neighborhood of the global solution at this stage. The following simple gradient-based optimization algorithm is proposed:

$$\frac{d\boldsymbol{u}_c}{dt} = \frac{dp_c}{d\boldsymbol{u}_c}, \qquad \boldsymbol{u}_c(0) = \boldsymbol{u}_0 \tag{33}$$

Regarding $\boldsymbol{u}_c$ as time variable $\boldsymbol{u}_c = \boldsymbol{u}_c(t)$ with initial condition $\boldsymbol{u}_0$, the differential equation has a fixed point at $\boldsymbol{u}_c^*$ satisfying equation (32). As long as the initial condition is in the neighborhood of the global solution, the proposed algorithm will generate a trajectory $\boldsymbol{u}_c(t)$ that leads from $\boldsymbol{u}_0$ to $\boldsymbol{u}_c^*$. Refer to Fig. 21 for explanation of the fixed point concept. Intuitively, choosing $\boldsymbol{u}_0$ such that $f(\boldsymbol{u}_0) = \boldsymbol{y}_0$ brings $\boldsymbol{u}_c$ close to $\boldsymbol{u}_c^*$. This would work perfectly if f was linear, but is sufficient for a nonlinear f with the properties of smoothness and monotonicity resulting from the fabrication processes.

The need to evaluate terms at every point k in the algorithm described by (33) is a distinct disadvantage. Although dimensionality of the u-space is reduced due to PCA, the algorithm can still be computationally inefficient. The efficiency can be improved by the following redefinition: Note that $\frac{dp_c}{dt} = -\frac{d}{dt}(1 - p_c)$. Probability $(1 - p_c)$ represents points that miss the target region and can be used by the algorithm as well:

$$\frac{d\boldsymbol{u}_c}{dt} = \sum_{k \notin \delta} s_k p(\boldsymbol{u}_k, \boldsymbol{u}_c) \frac{1}{2\sigma^2} \frac{d}{d\boldsymbol{u}_c} \|\boldsymbol{u}_k - \boldsymbol{u}_c\|^2 \tag{34}$$

By now σ was treated as a unit constant. However, during the optimization σ can be slowly varied, and the result $\boldsymbol{u}_c^*$ will be the same provided that the final value of σ is 1. Let σ be the parameter which slowly changes from some small initial value σ_0, up to 1 at the end of optimization. Perturbing σ will affect the solution $\boldsymbol{u}_c^*$ which now becomes a function of σ:

$$\boldsymbol{u}_c^* = \boldsymbol{u}_c^*(\sigma) \tag{35}$$

Parameter σ can be used to control the number of points affecting location of the solution.

6 Results of Yield Maximization for Stage-to-Final Models

The conceptual framework, introduced in Chapter 5, was implemented to achieve the yield enhancement in the MESFET fabrication process. Prior to design centering, the neural models of SCRG-F, S-F, CR-F, and G-F process stages were developed using DES-PREP program from the DESCENT software package. The models employ the PCA data preprocessing. Eigenvalues of the autocorrelation matrix of the input data characteristics distribution correspond to variances of the data spread in principal directions. By analyzing the variances, dimensionality reduction from n to m was made possible. Although the dimension reduction can potentially contribute to an error in solution for the centering problem, the entire approach to effective yield enhancement was found successful and it enabled optimization of the centering process.

The yield is estimated by comparing the modeled F-stage characteristics values with the tolerance ranges. If the value falls within that range, the yield test is considered as passed; if not, as failed. The tolerance ranges are defined as deviations allowed for respective characteristics around their target values. With the use of DES-CENT programs, desired values for SCRG, S, CR, and G characteristics were found for assumed tolerances and then the process yield was estimated for the new center values of these characteristics. Numerical simulations indicated that the yield can be significantly improved as compared with simple inversion without corrective design centering.

Our approach was based on the assumption that the underlying data models sufficiently characterize the fabrication stages and the relationships captured by the models are valid throughout the entire IC manufacturing process.

6.1 Model Analysis

The analysis of the measurement data used for building stage models from the perspective of further design centering was the first step in this work. Fabrication process identification becomes more reliable if dependencies between characteristics are better understood. Since the measurement process involves randomness, all the collected data needs to be regarded as a set of probabilistic distributions. Moreover, the characteristics describe the same fabrication process so it is reasonable to expect the mutual correlation of the distribution. For these reasons the input data of each model stage was first investigated through an autocorrelation matrix (14) by means of its principal components.

Four process models: SCRG-F, S-F, CR-F, and G-F, introduced in 3.1 and 3.2 were considered in this project. The input characteristics, listed previously in Tables 1 and 3, describe various process parameters and as such they differ in values, magnitudes and ranges. Therefore, a normalization technique (13) was found useful from the point of view of

k	λ_k	k	λ_k	k	λ_k	k	λ_k
1	16.1272	11	0.35186	21	0.02574	31	8.616e−07
2	4.23340	12	0.26605	22	0.02178	32	1.659e−10
3	2.71224	13	0.19993	23	0.01498		
4	1.72234	14	0.17673	24	0.01115		
5	1.50039	15	0.15153	25	0.00807		
6	1.25370	16	0.13849	26	0.00686		
7	1.00954	17	0.11797	27	0.00291		
8	0.83187	18	0.07011	28	0.00155		
9	0.54545	19	0.05616	29	0.00010		
10	0.39244	20	0.04934	30	4.933e−06		

Table 4: Eigenvalues of the SCRG distribution.

statistical analysis. Besides the need for normalization, mean values and standard deviations of the characteristics will be necessary in further testing procedures as well as directions in the multivariate space where the data is most strongly correlated. These parameters should be considered when creating testing data for the models. Also, they are necessary for the acceptance or rejection of solutions obtained from the design centering problem.

Eigenvalues of a model input data autocorrelation matrix represent variances along principal directions of the data in the input space. They can be found by numerically solving equation (15). The SCRG data distribution was characterized by 32 eigenvalues listed in Table 4. Resulting eigenvalues of the S, CR, and G data distributions are listed in columns Table 5. As expected, the inputs are strongly correlated along a few principal directions since the calculated eigenvalues significantly differ in magnitude and only a few have a distinct non-zero value.

The process of neural model development was improved after rotating the coordinate system of the input data accordingly. As a result of this rotation, the new system axes became aligned with the principal directions. The rotation, as in (17) was performed by the linear PCA operator (16) consisting of eigenvectors representing the principal directions arranged in descending order, corresponding to their significance from the most to the least important.

The input data representation error $\Delta_{\%}$, referred to as "PCA error" and expressed by (22), was evaluated for each of the models developed. Reducing the input space to m dimensions created an error which is shown for each distribution in Figures 22 and 23. The bar heights in these figures is scaled with respect to the largest eigenvalue. Thus, the error can be considered as a percentage of the maximum variance that the input data

k	λ_k (S)	λ_k (CR)	λ_k (G)
1	6.12445	4.32555	4.20851
2	1.29124	2.41000	1.59055
3	1.01986	0.56728	1.18649
4	0.77943	0.33684	0.81762
5	0.47338	0.21927	0.09877
6	0.23510	0.08336	0.07890
7	0.04448	0.03845	0.01028
8	0.03203	0.01920	0.00886
9	8.542e−06		
10	2.607e−09		

Table 5: Eigenvalues of S, CR, and G distributions.

had along the distribution principal direction. Limiting the number of models inputs to $m = 5$ resulted in an approximate 10% ratio for all the distributions except for the SCRG distribution which was found to be less than 50%.

The improvement of the training is shown in the test data scattering plots, in Figures 24 to 27, where scatterings denoted by (a) represent the testing data for models developed on original data without the PCA transformation, whereas scatterings denoted by (b) represent performance of the model with PCA pre-processed models inputs. Again notice that the PCA transformation of the model inputs enabled the test points to be aligned better with the scattering plot diagonal axis.

In addition to efficient measurement data representation, the PCA transformation enables reduction of the number of model input variables. In other words, the input space dimension for the models can be reduced by disregarding the least important directions of the data distribution. Although this creates an error in the data distribution representation, this is still feasible because of the significant differences in the eigenvalues. Also, the input space reduction plays a crucial role for the optimization algorithm employed in the DESCENT software package for design centering purposes. It is desirable to keep the number of the model inputs small; however, the input data representation error should not drastically affect model quality. This leads to a tradeoff between accuracy and computing time when deciding on the number of variables employed as the model inputs.

Successively, process models with 5 inputs were trained and their test scattering plots are shown in Fig. 24c to 27c. As indicated in these figures, the testing errors for the reduced models were larger as compared to the originals and non-reduced ones, however, they were comparable to the errors obtained for the models without either PCA or input

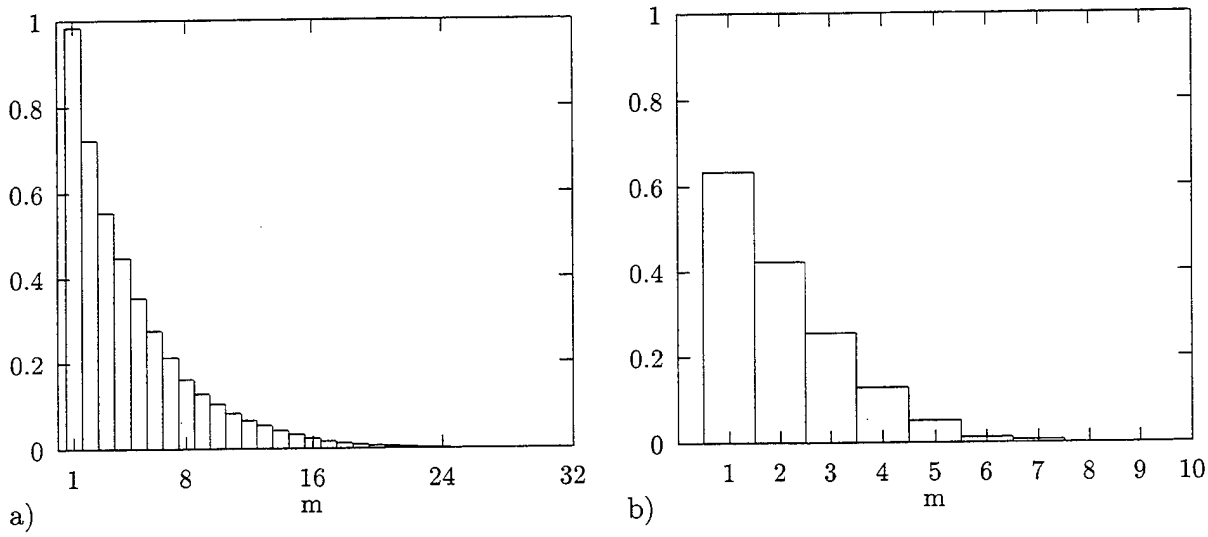


Figure 22: PCA error $\Delta_{\%}$ for: (a) SCRG distribution (b) S distribution.

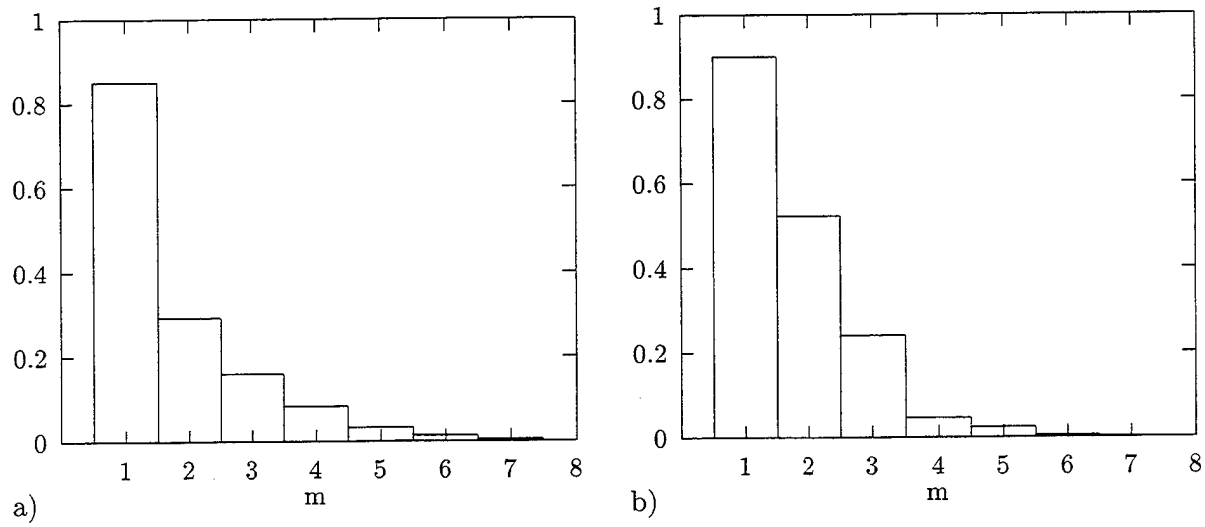


Figure 23: PCA error $\Delta_{\%}$ for: (a) CR distribution (b) G distribution.

space reduction. This was considered acceptable for the further design centering efforts.

6.2 Design Centering

The process model, as shown in Fig. 19, consists of two main components: PCA and the neural network. The inputs of the PCA portion are the input characteristics of a model process whereas the outputs of the neural network represent the process output characteristics. The intermediate variable $\boldsymbol{u}$ represents the model input and is so normalized that its mean value is a zero vector. Also, the data distribution represented in this abstract space is reduced to $m = 5$ variables (e.g. entries of vector $\boldsymbol{u}$) and the mutual correlation between these variables is zero. Standard deviations and thus the variances of each of the variables is one. This enables easy visualization of the set of data as points in the reduced abstract space, projected onto a two-dimensional coordinate system with u_1 and u_2 on the system axes. Figures 36 through 39 represent points visualized in this manner as dots. Location of points in this projection is evaluated for only two coordinates, and $u_3 = u_4 = u_5 = 0$ is assumed only for the purpose of visualization.

The PCA portion of the model is linear so the process nonlinearity is hidden in the neural network mapping. The model output value can be easily obtained using that mapping. To evaluate the model input value given an output target requires calculating $\boldsymbol{u}$ through the neural network mapping inversion. The first attempt at design centering was to find an input characteristic value by means of the model inversion. However, the model nonlinearity made this attempt non-optimal, especially when larger output characteristics tolerances were allowed.

This fact can be intuitively understood after representing the target location and the tolerance region in the abstract space. Consider the SCRG-F process model. Fig. 28 shows the target and tolerance regions for three tolerances: 5%, 10%, and 20%. Target values selected for this process are listed in Table 6. The assumed tolerances concern the selected three output characteristics Idss, Gm, and Vpo. The remaining characteristics assumed as non-critical are not restricted to any tolerance region, so their tolerances are assumed to be infinitively large. On the figures, the abstract representation of the tolerance region is shown as the unshaded area. The inverse to the target location is shown as the diamond. Although only two out of four abstract coordinates are included in the figures, it may be concluded that due to the nonlinearity of the model, the target is located off the optimal position within the tolerance region. Thus deviations in the input characteristics values will cause larger yield loss than if they were distributed around the center of the tolerance region.

The design centering algorithm that enables yield maximization has been introduced in section 5.4 of this report. The algorithm is implemented by the program DES-CENT

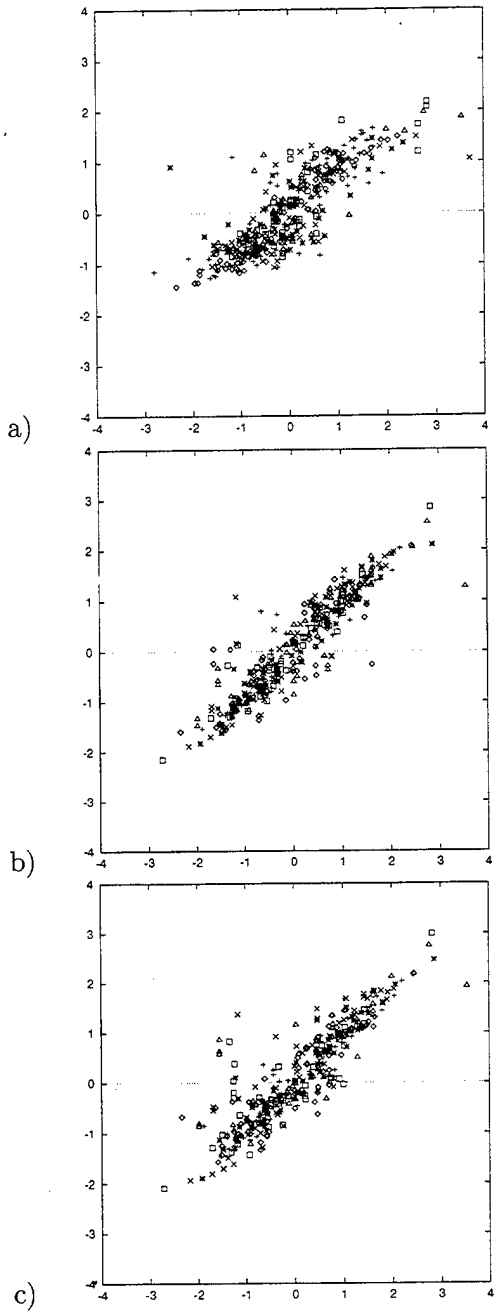


Figure 24: The SCRG-F model. Scattering plots for testing data. Model with (a) no PCA, (b) PCA, (c) reduced to 5 variables.

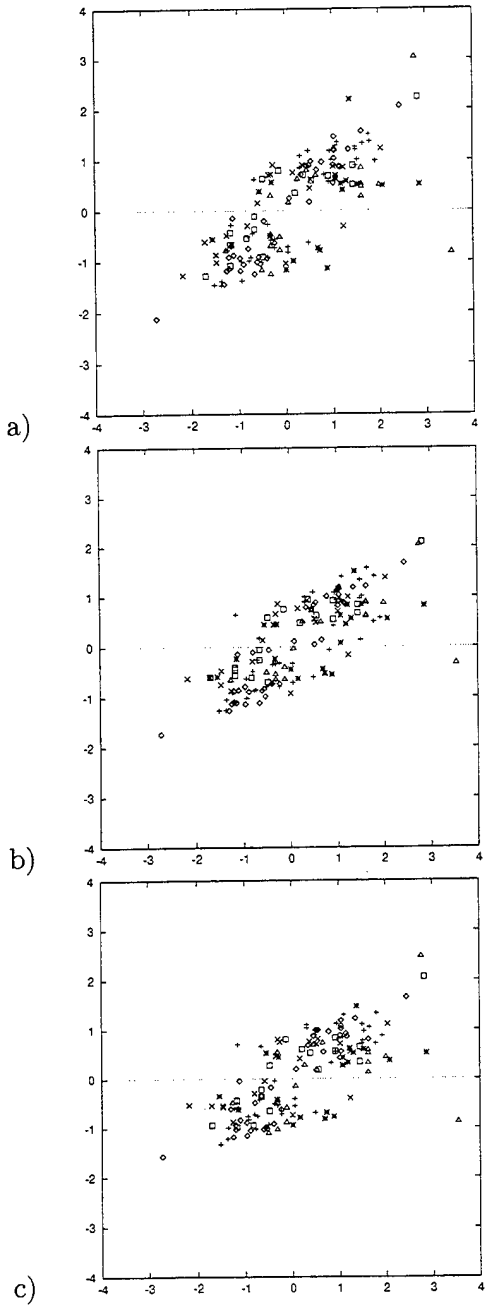


Figure 25: The S-F model. Scattering plots for testing data. Model with (a) no PCA, (b) PCA, (c) reduced to 5 variables.

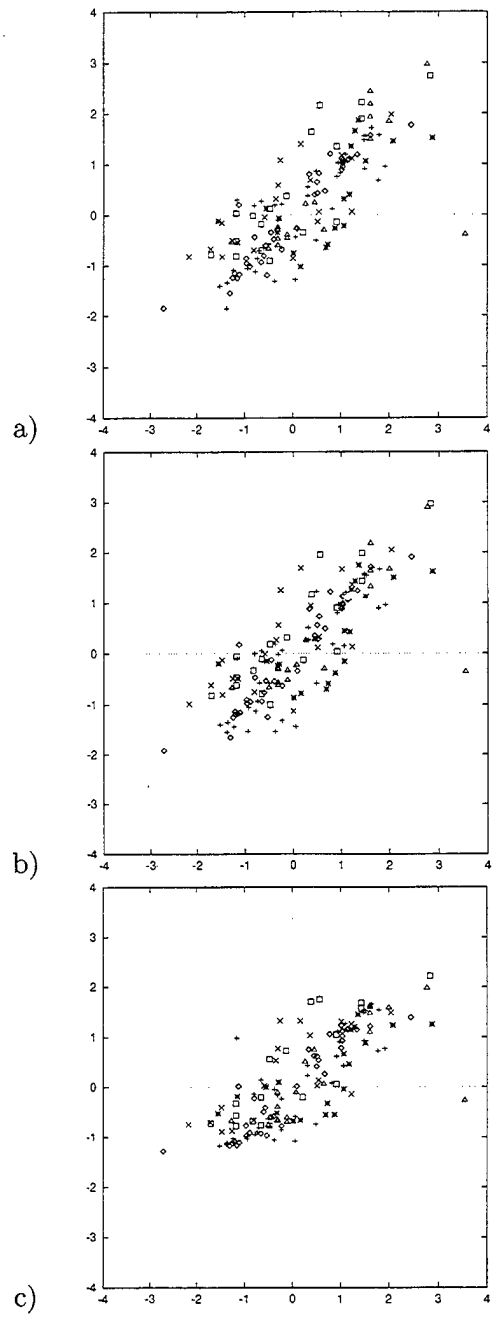


Figure 26: The CR-F model. Scattering plots for testing data. Model with (a) no PCA, (b) PCA, (c) reduced to 5 variables.

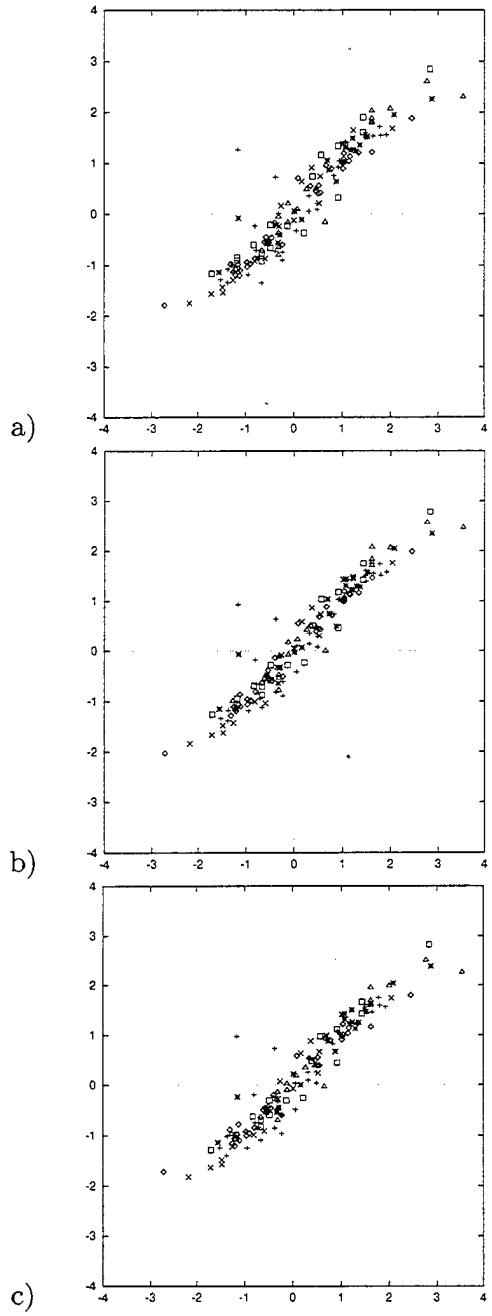


Figure 27: The G-F model. Scattering plots for testing data. Model with (a) no PCA, (b) PCA, (c) reduced to 5 variables.

Characteristics	Target value
F-Idss	224.0
F-Rds	2.6482
F-Rgs	3.458
F-Rs	0.844032
F-Rdg	3.720
F-Rd	1.10165
F-Vbdg	8.7719
F-Vbgs	7.769
F-Vpo	-1.495
F-Gm	201.1
G-Ids-pk	243.64
Lg	0.199696
C	5.93036
i-Rsh	0.004311
i-W	11.5838
p-Rsh	1.87114
p-W	11.3693
BH	1.33161
BG	36.5538

Table 6: Target F values for SCRG-F process.

(included in the entire package DESCENT). The optimal solutions for the input settings were found using the process models SCRG-F, S-F, CR-F, and G-F, assuming the target final characteristics values as listed in Tables 6 and 7. The results obtained are shown in Fig. 28 through 31 for each stage. Centered values for each of the stages for assumed tolerances are indicated by the cross. As the design center algorithm progresses, the initial point, $\boldsymbol{u}_0$, denoted as diamond, obtained by the inversion of the target point $\boldsymbol{y}_0$, is moved to the optimal location, which maximizes the yield of a respective fabrication process.

Upon completion of the algorithm, the center values of input characteristics are found with the PCA inverse operator (19) and the improved yield is estimated. The centered input characteristics values computed for the analyzed considered stages are summarized in Tables 8 and 9. In the Tables $\boldsymbol{x}_0$ is the process stage input inverse to the target $\boldsymbol{y}_0$. In other words, if setting $\boldsymbol{x}_0$ is chosen as the input of the process model, the model will respond with output $\boldsymbol{y}_0$. However, the actual process involves random fluctuations that affect the input settings and cause the yield loss. Then, assuming a specified tolerance level, another input

Characteristics	Target value
F-Idss	224.0
F-Rds	2.674
F-Rgs	3.514
F-Rs	0.8926
F-Rdg	3.678
F-Rd	1.053
F-Vpo	-1.495
F-Gm	201.1

Table 7: Target F values for S-F, CR-F, and G-F process.

center point x_c should be selected for the sake of the yield maximization.

As shown in Tables 8 and 9, final location of the center point x_c depends on the tolerance level selected. Roughly, large tolerances result in a large shift to the input center point x_c. To illustrate this variability the percentage change of the inverse solution x_0, evaluated as $(x_c - x_0)/x_0$, has been graphed for various tolerances in Figs. 32 through 35. Note that some of the input characteristics affect the yield significantly stronger than others. Obviously, the height of bars shown in the figures corresponds to the sensitivity of the process yield to particular input characteristics. This provides indications which process parameters are the most important for yield enhancement. For the processes investigated, the input characteristics can be referenced in Tables 1 and 2.

6.3 Yield Enhancement Test

The statistical yield estimation is implemented in the DESCENT package by a program named DES-TRY. This program performs verification of the centering data. The following procedure performs the fabrication process yield estimation: First, the input characteristics distribution is identified in order to enable generation of random points that conform to the distribution statistical parameters. The training data autocorrelation matrix is always prepared for PCA. The matrix can then be successfully used for identification of statistical parameters of the input characteristic distributions.

Secondly, a large number of random points from Gaussian distribution must be generated and then used for yield testing. The Gaussian distribution should have statistical parameters, such as variances and mutual correlations of characteristics, equal to the ones present in the original training data. In this approach the use of 10000 points was found to be sufficient for the yield estimation.

Successively, each of the generated points, treated as the model input, can be evaluated

	Characteristics	x_0	$x_c(5\%)$	$x_c(10\%)$	$x_c(20\%)$
1	OBSA	12.7371	12.1568	11.045	5.72798
2	EL2	1.2015e+16	1.20066e+16	1.20368e+16	1.17728e+16
3	OBSB	13.3056	12.5071	11.9152	0.804608
4	Rho	1.72557e+08	1.75338e+08	1.77475e+08	2.15856e+08
5	MuH	5659.08	5607.48	5565.15	4863.74
6	ns	6.45442e+06	6.41278e+06	6.38268e+06	5.80021e+06
7	Nd	1.77691e+18	1.74718e+18	1.75706e+18	1.21431e+18
8	ETA	657.323	646.326	649.98	449.201
9	Mu0	1524.63	1523.54	1522.15	1509.38
10	Mu1	26400.9	25473.3	25385.1	10053.6
11	C-Idss	925.17	919.348	915.339	833.106
12	C-Rds	1.75049	1.7637	1.78973	1.90798
13	C-Rc	15.8768	16.366	18.6594	17.6571
14	C-Rsh	16581.4	16663.2	16525.7	18464.1
15	O-Rsh	0.338449	0.34164	0.345169	0.384863
16	O-W	10.261	10.2572	10.2074	10.3451
17	R-Ids	642.938	635.591	631.212	524.66
18	R-Rds	2.32213	2.33644	2.37673	2.45576
19	G-Idss	220.691	213.373	211.83	94.3121
20	G-Rds	2.81938	2.87351	2.91545	3.66116
21	G-Rgs	3.72193	3.74162	3.79578	3.90967
22	G-Rs	1.09409	1.09919	1.10659	1.16292
23	G-Rdg	3.39089	3.41537	3.48194	3.62667
24	G-Rd	0.778185	0.789542	0.810896	0.916601
25	G-Vbdg	8.54138	9.27326	10.0964	19.1446
26	G-Vbgs	7.92374	7.81729	7.48703	7.02254
27	G-Vpo	-1.2877	-1.29316	-1.21426	-1.62586
28	G-Gm	204.103	202.879	201.863	185.259
29	G-Ids-pk	220.532	213.213	211.668	94.134
30	G-AL	1.57001	1.57125	1.58131	1.56171
31	G-Rsh	0.0579221	0.0583812	0.0589691	0.0643541
32	G-W	10.0864	10.0887	10.0782	10.1607

Table 8: Inverse and centered solutions for a given target and tolerances of 5, 10, 20%; SCRG stage.

Characteristics	x_0	$x_c(5\%)$	$x_c(10\%)$	$x_c(20\%)$
OBSA	10.2265	9.94636	4.44489	2.44095
EL2	1.26928e+16	1.24778e+16	1.48646e+16	1.30234e+16
OBSB	12.4846	11.1765	6.05006	-1.92648
Rho	1.7403e+08	1.79282e+08	1.8797e+08	2.23325e+08
MuH	5632.27	5535.22	5388.58	4731.32
ns	6.42992e+06	6.35175e+06	6.20051e+06	5.67978e+06
Nd	1.89656e+18	1.80194e+18	1.90554e+18	1.30441e+18
ETA	701.584	666.583	704.906	482.526
Mu0	1525.11	1522.61	1500.96	1494.24
Mu1	29056.5	26508.3	28172.7	12024.8
C-Idss	922.96	921.31	926.01	827.71
C-Rds	1.7652	1.7230	1.7823	1.9263
R-Idss	17.183	18.627	21.120	19.933
R-Rds	16501	17928	17061	18252
Rc	0.3404	0.3296	0.3607	0.3843
O-Rsh	10.232	10.358	10.234	10.318
O-W	640.54	647.01	625.01	520.77
R-Rds	2.3450	2.4383	2.1505	2.4991
G-Idss	240.53	226.34	220.925	166.523
G-Rds	2.7628	2.8279	2.88418	3.22121
G-Rgs	3.74395	3.76187	3.79176	3.89617
G-Rs	1.07237	1.07749	1.07304	1.14254
G-Rdg	3.45406	3.48515	3.54396	3.58107
G-Rd	0.809363	0.818058	0.845512	0.856502
G-Vpo	-1.11725	-1.12633	-1.07821	-1.35426
G-Gm	206.417	205.105	204.414	194.203

Table 9: Inverse and centered solutions for a given target and tolerances of 5, 10, 20%; S, CR, and G stages.

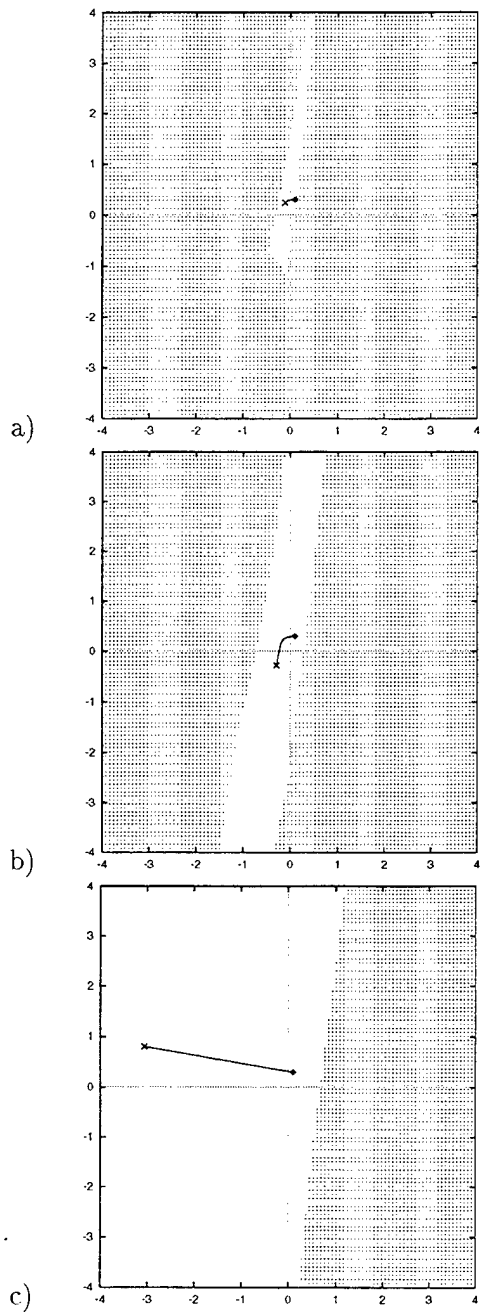


Figure 28: Design centering in SCRG-F fabrication stage. The diamond represents an inverse $\boldsymbol{u}_0$ to target $\boldsymbol{y}_0$ in the abstract coordinates u_1–u_2. Centered value $\boldsymbol{u}_c$, indicated by the cross, enables the yield maximization, within tolerances δ equal (a) 5%, (b) 10%, and (c) 20%.

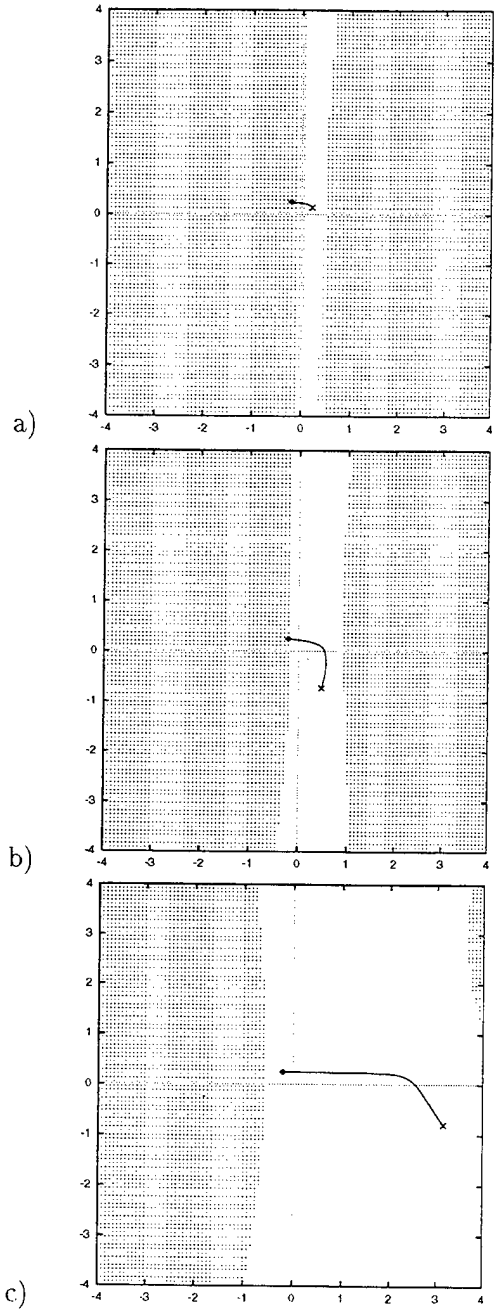


Figure 29: Design centering in S-F fabrication stage. The diamond represents an inverse $\boldsymbol{u}_0$ to target $\boldsymbol{y}_0$ in the abstract coordinates u_1–u_2. Centered value $\boldsymbol{u}_c$, indicated by the cross, enables the yield maximization, within tolerances δ equal (a) 5%, (b) 10%, and (c) 20%.

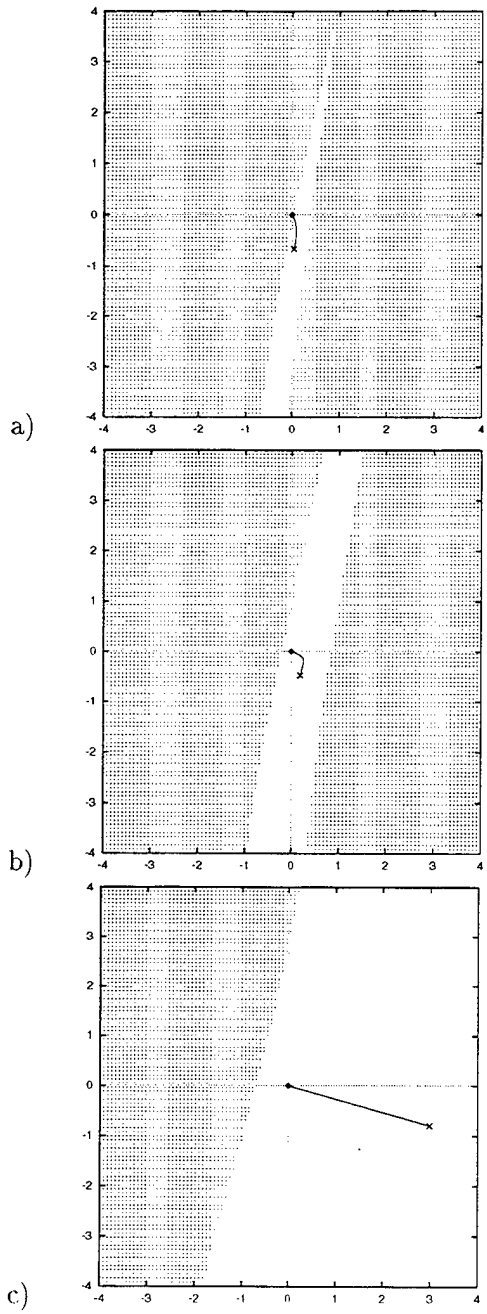


Figure 30: Design centering in CR-F fabrication stage. The diamond represents an inverse $\boldsymbol{u}_0$ to target $\boldsymbol{y}_0$ in the abstract coordinates u_1–u_2. Centered value $\boldsymbol{u}_c$, indicated by the cross, enables the yield maximization, within tolerances δ equal (a) 5%, (b) 10%, and (c) 20%.

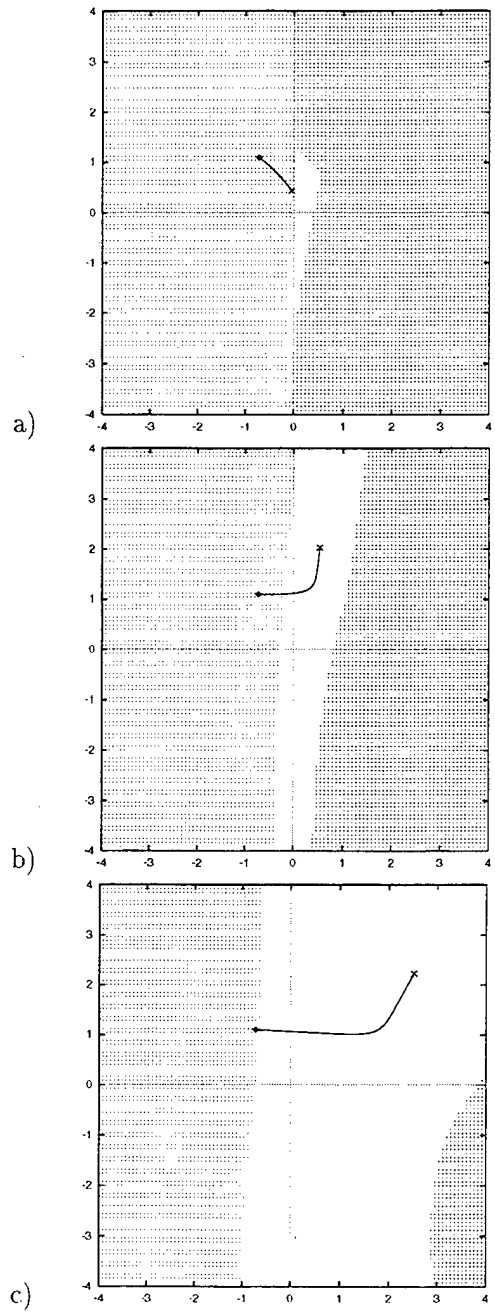


Figure 31: Design centering in G-F fabrication stage. The diamond represents an inverse $\boldsymbol{u}_0$ to target $\boldsymbol{y}_0$ in the abstract coordinates u_1–u_2. Centered value $\boldsymbol{u}_c$, indicated by the cross, enables for the yield maximization, within tolerances δ equal (a) 5%, (b) 10%, and (c) 20%.

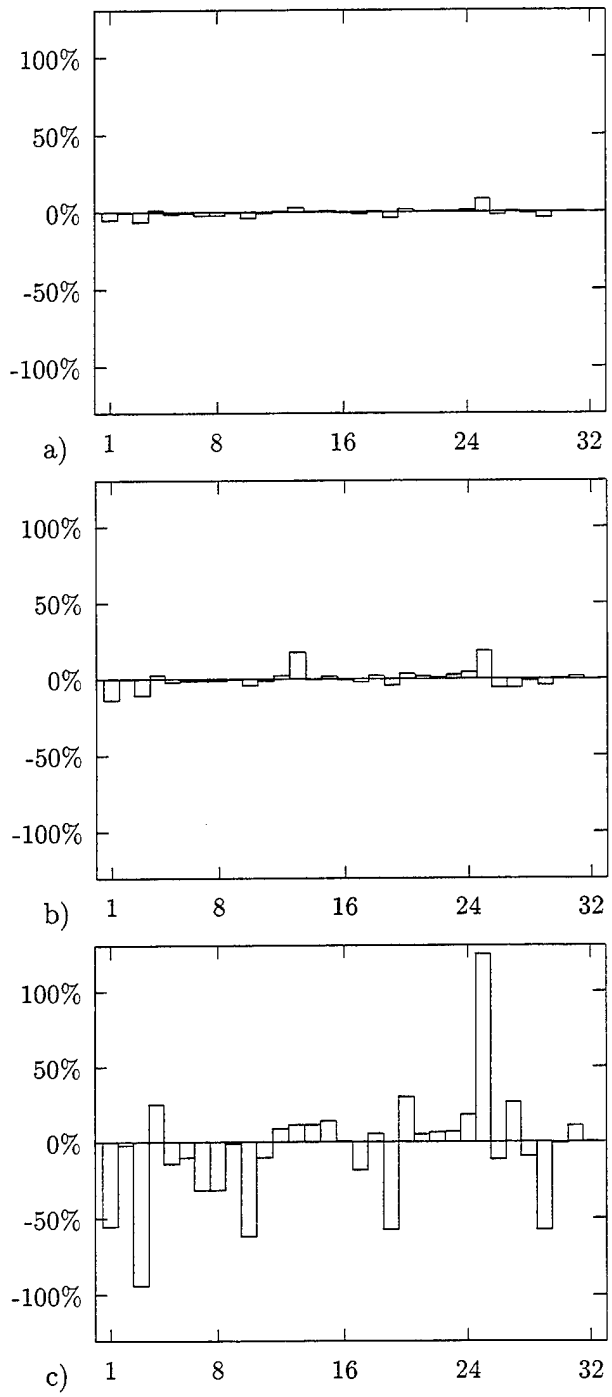


Figure 32: Percentage change to the inverse solution after centering for (a) 5%, (b) 10%, and (c) 20% tolerance; SCRG stage.

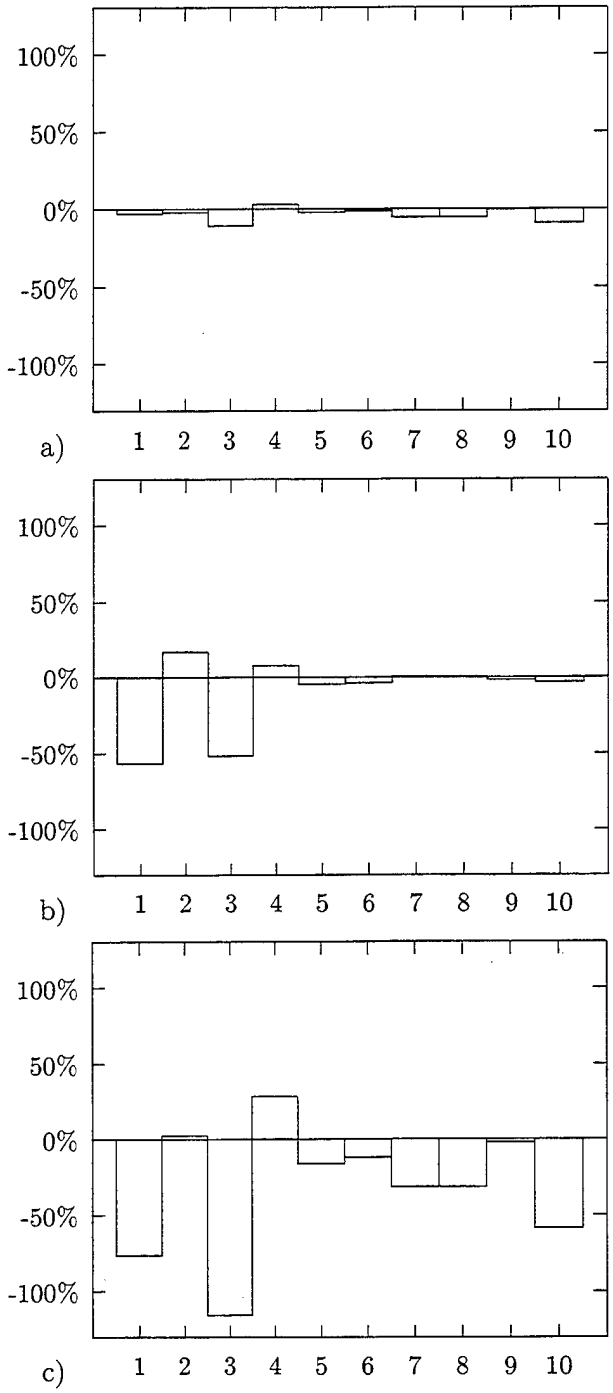


Figure 33: Percentage change to the inverse solution after centering for (a) 5%, (b) 10%, and (c) 20% tolerance; S stage.

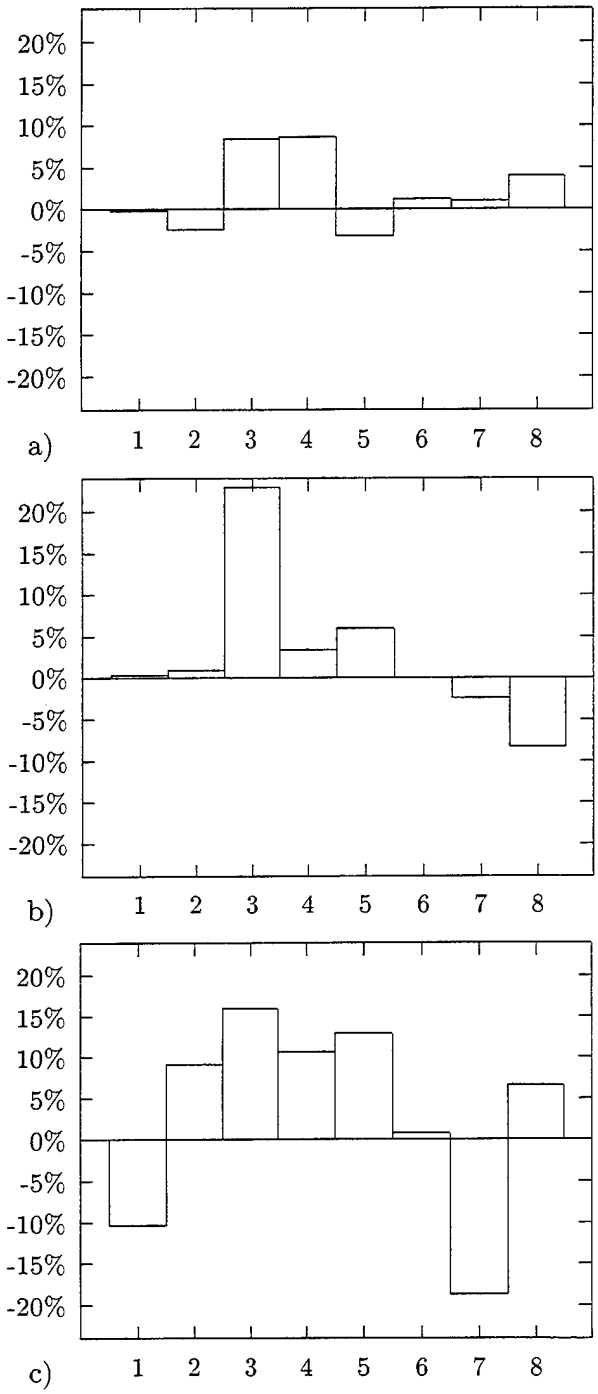


Figure 34: Percentage change to the inverse solution after centering for (a) 5%, (b) 10%, and (c) 20% tolerance; CR stage.

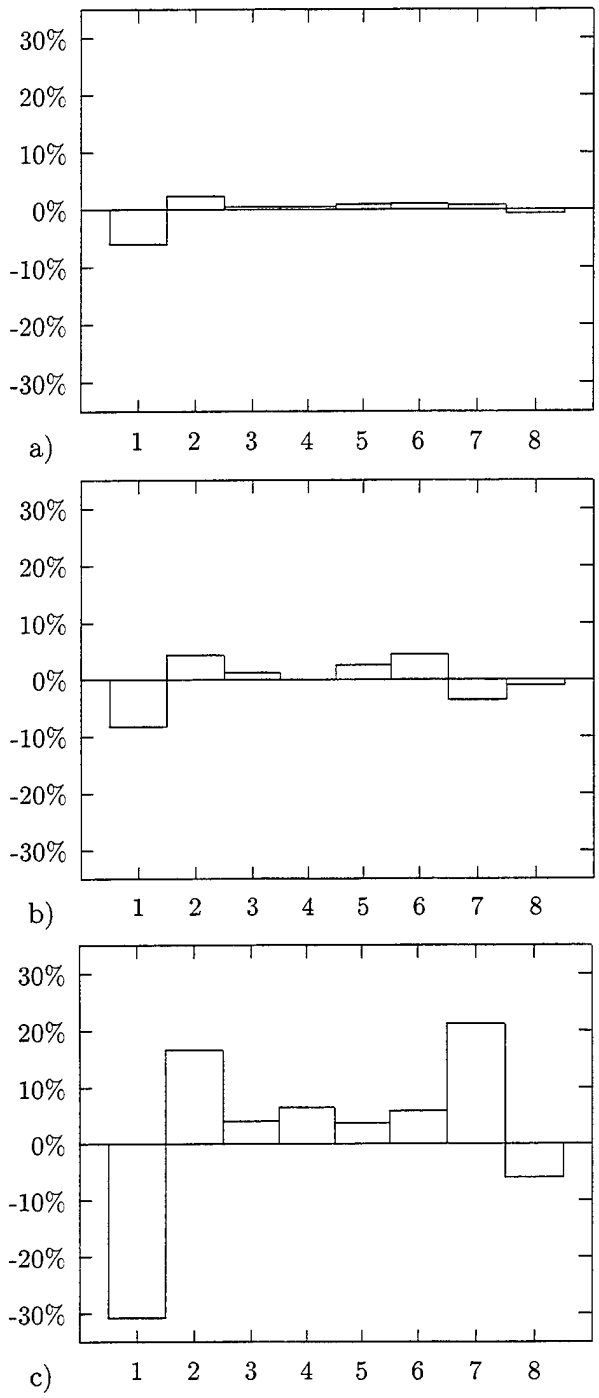


Figure 35: Percentage change to the inverse solution after centering for (a) 5%, (b) 10%, and (c) 20% tolerance; G stage.

Stage	δ	No centering	Centered
SCRG-F	5%	10.01%	10.78%
	10%	36.39%	38.71%
	20%	74.13%	99.99%
S-F	5%	13.38%	14.29%
	10%	38.18%	41.15%
	20%	73.54%	99.28%
CR-F	5%	12.04%	12.20%
	10%	29.48%	37.35%
	20%	55.68%	75.01%
G-f	5%	11.38%	12.78%
	10%	37.63%	39.94%
	20%	74.79%	98.62%

Table 10: Fabrication yield for inverse solution and centered solution with allowed tolerances δ.

for the corresponding output characteristics values, through the model. Simultaneously, its location in the abstract space can be visualized, as shown in Figures 36 through 39. There are 10000 small dots representing the points on each of the figures.

Once the output characteristics values are known, the points can be checked to see if they fall within the tolerance region. The percentage of points that pass the test is the computed process yield estimate. In Figures 36 to 39 the points which pass the tolerance test are represented by the bold points.

Each of the investigated fabrication processes was tested with the program DES-TRY for yield enhancement after the design centering was completed by the DESCENT package. The results are shown in Table 10. The yield was estimated for two cases: with no centering involved, and after centering. In the first column the input characteristic distributions have a mean value equal to $\boldsymbol{x}_0$ as obtained from the assumed target output by the model inversion. In the second case, the centered input $\boldsymbol{x}_c$ has been found using the design centering algorithm, and used as the input distribution mean, resulting with an improved yield. Each process stage was evaluated in this manner for three values of tolerances δ. As seen in the Table, the yield improvement becomes significant for large tolerances, which is reasonable due to the smoothness and nonlinearity of the process models.

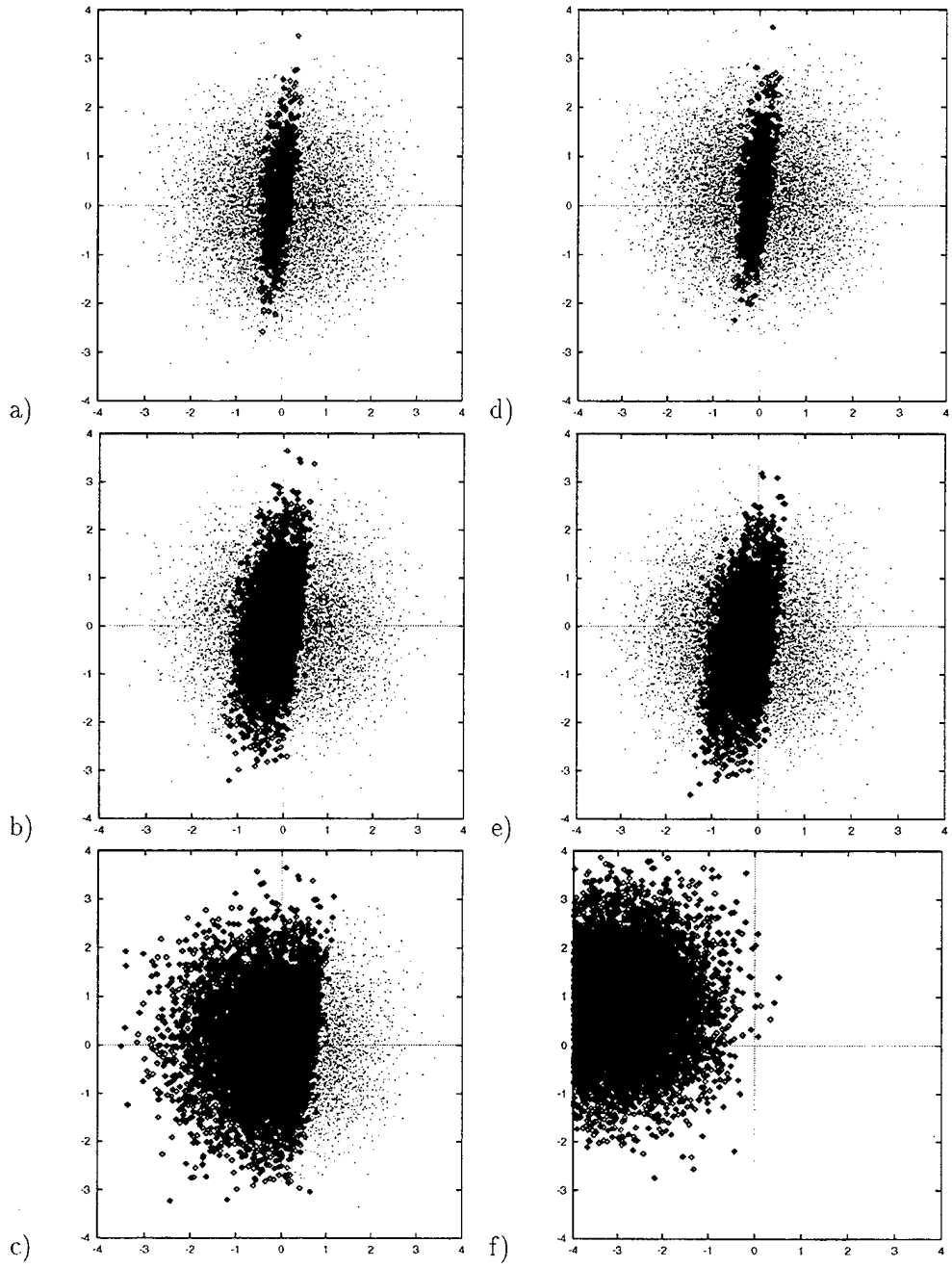


Figure 36: Yield test for the SCRG-F reduced model: before centering (a) 5% tolerance, (b) 10% tolerance, (c) 20% tolerance; and after centering (d) 5% tolerance, (e) 10% tolerance, (f) 20% tolerance.

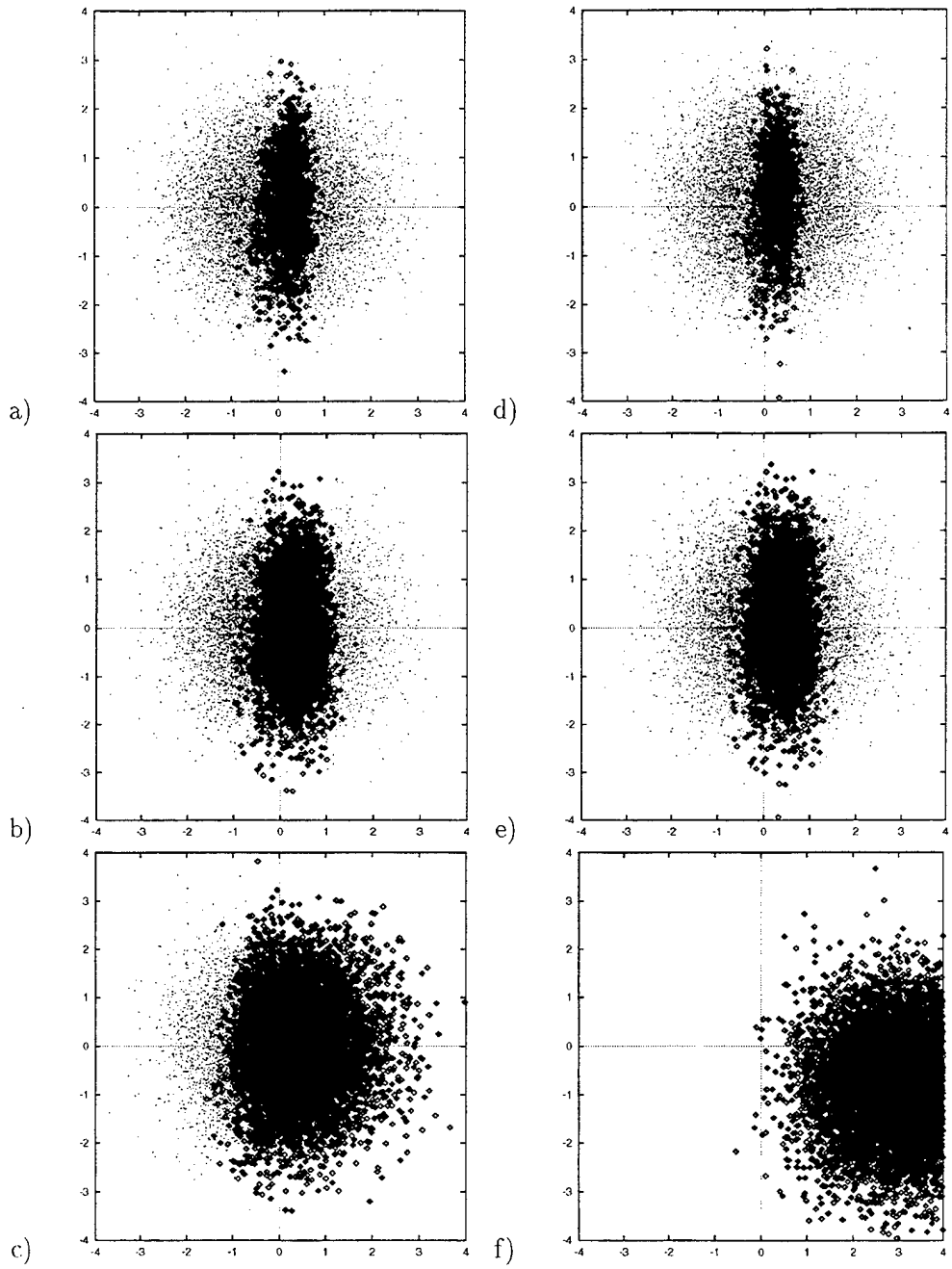


Figure 37: Yield test for the S-F reduced model: before centering (a) 5% tolerance, (b) 10% tolerance, (c) 20% tolerance; and after centering (d) 5% tolerance, (e) 10% tolerance, (f) 20% tolerance.

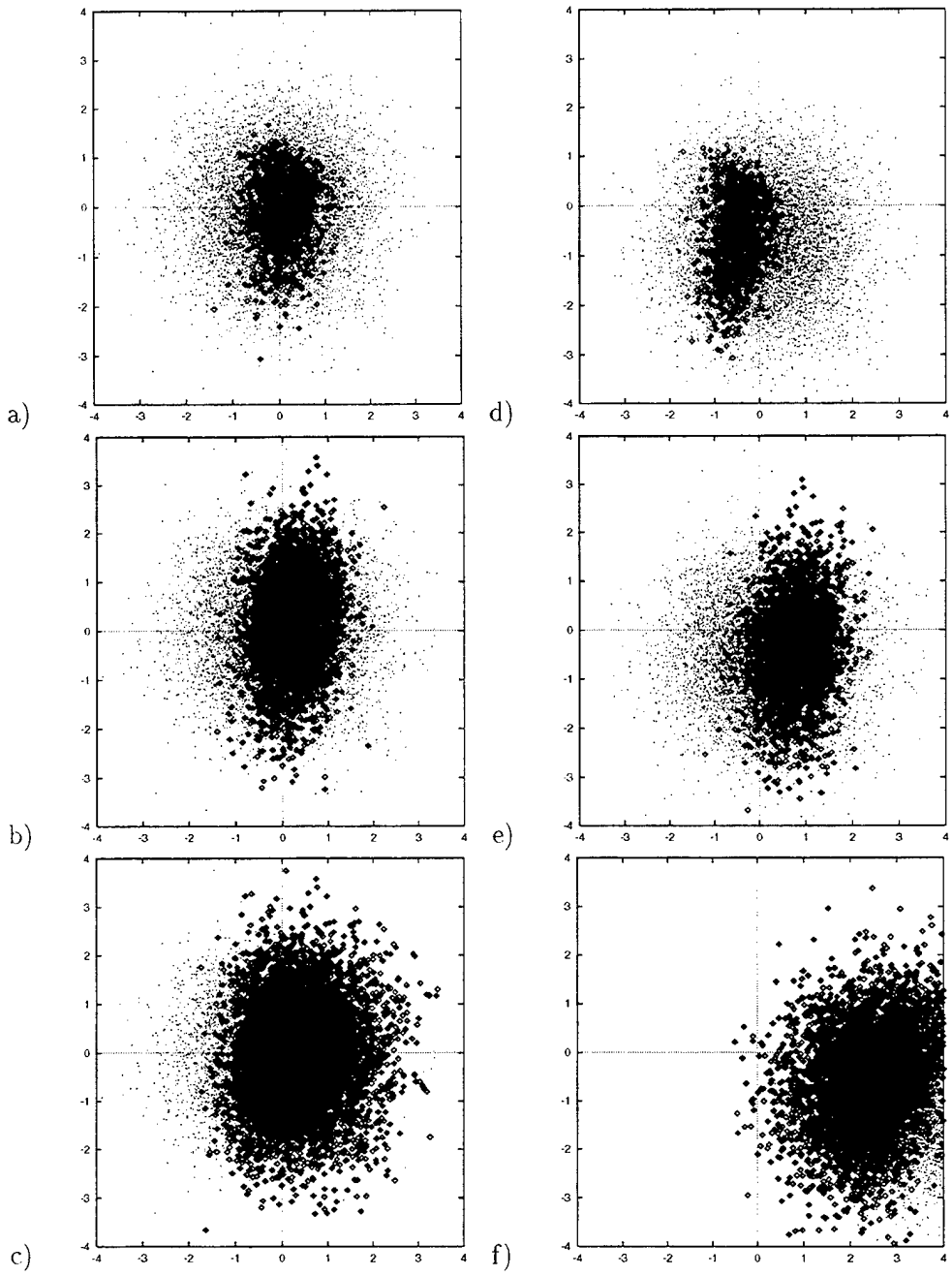


Figure 38: Yield test for the CR-F reduced model: before centering (a) 5% tolerance, (b) 10% tolerance, (c) 20% tolerance; and after centering (a) 5% tolerance, (b) 10% tolerance, (c) 20% tolerance.

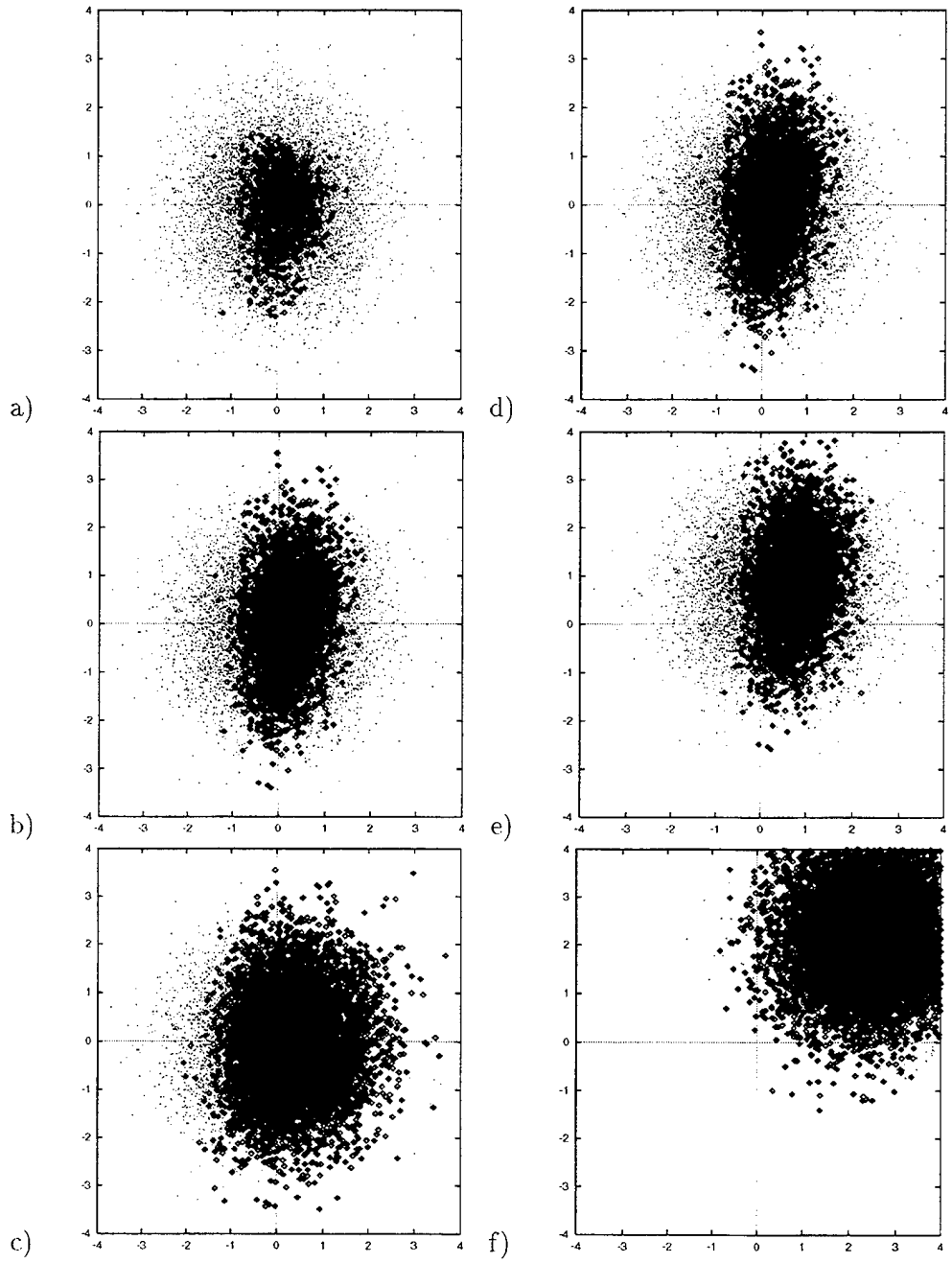


Figure 39: Yield test for the G-F reduced model: before centering (a) 5% tolerance, (b) 10% tolerance, (c) 20% tolerance; and after centering (d) 5% tolerance, (e) 10% tolerance, (f) 20% tolerance.

7 Conclusions

The presented design centering approach and related software package enabled yield maximization in fabrication processes described by numerical data taken from process measurements. The yield can be significantly improved, particularly when nonlinear relationships and larger tolerances are involved in the process characterization. This is the case for the manufacture of GaAs microelectronic devices. The design centering algorithm can work efficiently even with large measurements data sets since a Principal Component Analysis is performed on the raw data to reduce the problem size and thus avoid the "curse of dimensionality".

In addition, the package for design centering DESCENT offers a trade-off between the inverse modeling/design centering error and the computational complexity of the solution. Less accurate design centering solutions can be produced quickly through modeling with few principal components, while more precise solutions would require inclusion of more variables. In the latter case, optimization needs to be performed in multidimensional space and at a higher computational expense.

References

[1] G. L. Creech, J. M. Zurada, and P. Aronhime, "Feedforward neural networks for estimating IC parametric yield and device characterization," in *Proc. of the 1995 IEEE International Symposium on Circuits and Systems*, vol. 2, (Seattle, Washington), pp. 1520–1523, April 30–May 3 1995.

[2] G. L. Creech and J. M. Zurada, "Inverse modeling of MMIC fabrication material and device characteristics using feedforward neural networks," in *Proc. of the GaAs Manufacturing Technology Conference*, (New Orleans, Louisiana), pp. 20–23, May 9–12 1995.

[3] J. C. Zhang and M. A. Styblinski, *Yield and Variability Optimization of Integrated Circuits.* Kluwer Academic Publishers, 1995.

[4] G. L. Creech and J. M. Zurada, "GaAs MESFET DC characteristics and process modeling using neural networks," in *Proc. of the Artificial Neural Networks in Engineering*, (St. Louis, MO), Nov. 13–16, 1994.

[5] G. L. Creech, *Neural Network Modeling for GaAs IC Fabrication Process and Device Characteristics.* PhD thesis, University of Louisville, Louisville, KY, May 1996.

[6] J. M. Zurada and A. Malinowski, "Multilayer perceptron networks: Selected aspects of training optimization," *Applied Mathematics and Computer Science*, vol. 4, no. 3, pp. 281–307, 1994.

[7] A. Malinowski, J. M. Zurada, and P. B. Aronhime, "Minimal training set size estimation for neural network-based function approximation," in *Proc. of the IEEE International Conference on Neural Networks*, (Orlando, Florida), pp. 1125–1129, June 28–July 2 1994.

[8] J. H. Friedman, "An overview of predictive learning and function approximation," in *From Statistics to Neural Networks*, pp. 1–61, New York, NY: Springer-Verlag, 1991.

[9] D. E. Hocevar, M. R. Lightner, and T. N. Trick, "A study of variance reduction techniques for estimating circuit yields," *IEEE Trans. on Computer-Aided Design*, vol. 2, pp. 180–192, July 1983.

[10] C. Spanos and S. W. Director, "Parameter extraction for statistical process characterization," *IEEE Trans. on Computer-Aided Design of Integrated Circuit and Systems*, vol. 1, pp. 66–78, Jan 1986.

[11] K. K. Low and S. W. Director, "A new methodology for the design centering of IC fabrication processes," *Proc. of Int. Sym. on Circuits and Systems*, pp. 194–197, 1989.

[12] M. A. Styblinski and L. J. Opalski, "Algorithms and software tools for IC yield optimization based on fundamental fabrication parameters," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 1, pp. 66–78, Jan 1986.

[13] J. P. Spoto, W. T. Coston, and C. P. Hernandez, "Statistical integrated circuit design and characterization," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 5, pp. 90–103, Jan 1986.

[14] J. King *et al.*, "Overview of MIMIC Phase I materials/device correlation study," in *GOMAC Digest*, (Orlando, Florida), pp. 331–334, Nov. 5–7 1991.

[15] D. H. Rosenblatt, W. R. Hitchens, R. E. Anholt, and T. W. Sigmon, "GaAs MESFET device dependences on ion-implant tilt and rotation angles," *IEEE Electron Device Letters*, vol. 9, pp. 139–141, March 1988.

[16] P. Dobrilla, J. S. Blakemore, A. J. McCamant, K. R. Gleason, and R. Y. Koyama, "GaAs field effect transistor properties, as influenced by the local concentrations of midgap native donors and dislocations," *Applied Physics Letters*, vol. 47, no. 6, pp. 602–605, 1985.

[17] J. Purviance, M. D. Meehan, and D. M. Collins, "Properties of FET statistical data bases," in *IEEE MTT Symp. Digest*, (Dallas, Texas), pp. 567–570, May 8-10 1990.

[18] R. Anholt, "Process and device models of monolithic-microwave integrated-circuit uniformity," in *GOMAC Digest*, (Orlando, Florida), pp. 339–342, Nov 5-7 1991.

[19] S. Sze, *Physics of Semiconductor Devices*, ch. 6. New York: Wiley, 1981.

[20] R. Anholt, R. Worley, and R. Neidhard, "Statistical analysis of GaAs MESFET S-parameter equivalent-circuit models," *Int. Journal. Microwave, mm-Wave CAE*, vol. 1, no. 3, pp. 263–270, 1991.

[21] D. Khera, M. Zaghloul, L. Linholm, and C. Wilson, "A neural network approach for classifying test structure results," in *Proc. of the 1989 Int. Conf. on Microelectronic Test Structures*, vol. 2, pp. 200–204, March 1989.

[22] M. Zaghloul, D. Khera, L. Linholm, and C. Reeve, "A machine-learning classification approach for IC manufacturing control based on test structure measurements," *IEEE Trans. on Semiconductor Manufacturing*, vol. 2, pp. 47–53, May 1989.

[23] D. C. Look, D. C. Walters, R. T. Kemerley, J. M. King, M. G. Mier, J. S. Sewell, and J. S. Sizelove, "Wafer-level correlations of EL2, dislocation density, and FET saturation current at various processing stages," *J. Electronic Material*, vol. 18, no. 4, pp. 487–492, 1989.

[24] J. M. Zurada, *Introduction to Artificial Neural Systems.* Boston: PWS, 1992.

[25] J. M. Zurada, A. Malinowski, and I. Cloete, "Sensitivity analysis for minimization of input data dimension for feedforward neural network," in *Proc. of the IEEE International Symposium an Circuits and Systems*, (London, England), pp. 447–450, May 30–June 2, 1994.

[26] J. M. Zurada, A. Malinowski, and I. Cloete, "Sensitivity analysis for minimization of input data dimension for feedforward neural network," in *Proc. of the 1994 International IEEE Symposium on Circuits and Systems*, vol. 6, (London, England), pp. 447–450, May 29–June 1 1994.

[27] J. M. Zurada and A. Malinowski, "Sensitivity analysis for pruning of training data in feedforward neural networks," in *Proc. of Australian–New Zealand Conference on Intelligent Information Systems*, (Perth, Australia), pp. 288–292, Dec. 1–3, 1993.

[28] J. M. Zurada, A. Malinowski, and S. Usui, "Perturbation method for deleting redundant inputs of perceptron neural networks," *Neurocomputing*, no. 14, pp. 177–193, 1997.

[29] J. M. Zurada, A. Lozowski, and A. Malinowski, "Design centering in GaAs IC manufacturing," in *Proc. of the 1997 IEEE Aerospace Conference*, (Snowmass at Aspen, Colorado), pp. 435–447, February 1-8 1997.

[30] J. M. Zurada, A. Lozowski, and A. Malinowski, "Yield improvement in GaAs IC manufacturing using neural network inverse modeling." submitted for the 1997 International IEEE Conference on Neural Networks, Houston, Texas, June 9-12 1997.

[31] D. A. Hoskins, J. N. Hwang, and J. Vagners, "Iterative inversion of neural networks and its application to adaptive control," *IEEE Transactions on Neural Networks*, vol. 3, pp. 292–301, March 1992.

[32] H. Deif, J. M. Zurada, and W. Kuczborski, "Inverse mapping of continuous functions using feedforward neural networks." submitted for the 1997 International IEEE Conference on Neural Networks, Houston, Texas, June 9-12 1997.

[33] K. Fukunaga, *Introduction to Statistical Pettern Recognition.* Academic Press, New York, 1972.

[34] A. Cichocki and R. Unbehauen, *Neural Network for Optimization and Signal Processing.* John Wiley & Sons, 1993.

[35] A. Malinowski, *Reduced Perceptron Networks and Inverse Mapping Control.* PhD thesis, University of Louisville, Louisville, KY, June 1996.

8 Appendix

8.1 Wtab Program Description

Wtab is a program that tabulates the data from each reticle on a wafer. Several input options enable Wtab to organize the data in different forms. The first option lets the user specify the characteristics to be examined. Otherwise, Wtab will report every possible wafer characteristic, even if no measurements of that characteristic were made on that wafer. In addition to saving execution time, this feature will greatly reduce the size of the generated wafer table.

A second feature of Wtab is the reticle referencing option. By specifying a list of reticles, only those reticles will be listed in the wafer table. If a given reticle does not contain any characteristic data, or if the reticle does not contain data for all of the characteristics, Wtab will search for the reticle's nearest neighbor to find characteristic data values. Wtab's third main feature is very similar to the reticle referencing option, but it allows sub-reticle locations (test structure pad numbers) to be specified. This option must be used if the reticle referencing option is to be invoked.

Usage

Only one input value is required by Wtab, the wafer identification string (see the Program Operations section for more details on the wafer identification string). This string is the basis for all of the names of the data files that Wtab will tabularize. Wtab creates up to three output files (depending upon the options selected), with each output file name beginning with the wafer identification string. If the (-o) option is selected, the output file names will not use the wafer identification string as a base; instead, Wtab will invoke the reticle referencing option.Likewise, if (-w) is followed by a file name, the reticle locations contained in the file immediately following the (-w) will be used in the reticle referencing option.

Program Operation

When Wtab is invoked, the first item examined is the wafer identification string. This string is composed of eight alphanumeric symbols representing the company name (or source of the data), the process lot, the boule source, the boule number, and the wafer number. The characteristic data files should be located in a directory that is loosely based on the wafer identification string. These data files have names composed of the six symbol characteristic concatenated to the wafer identification string with a file extension of 'dat'. Each characteristic data file contains a variable number of measurements, with every measurement composed of two parts. The reticle/sub-reticle location, which consists of a reticle XXYY

location in addition to the sub-reticle xxyy location, is the first part. Next is the actual value measured for the characteristic of the location.

As the next step in the program operation, the command line options are parsed, and flags are set depending on the presence or absence of the options. If the characteristic option is given (-c), then Wtab opens the specified file and reads in the characteristics contained in the file into an array of strings. No attempt is made to check if these characteristics are valid; that is, if the characteristics are ones that are contained in the file 'characteristics.h'.

Next, Wtab searches through all the characteristic data files for data values. If a reticle location exists, the data for a specific characteristic is added to a linked list for that specific reticle location. Otherwise, a new reticle location is appended to a linked list of reticle locations for the wafer. Once all of the characteristics data files have been processed, a wafer table is written using a basename (either the wafer identification string or the string specified by the -o command line option) and the extension 'waf'. This file is then sorted by the reticle location.

If the sub-reticle reference option is used, Wtab reads the referenced sub-reticles from the file name supplied on the command line into an array. Wtab then locates the nearest reticle/sub-reticle with a characteristic data value for each referenced sub-reticle using Euclidean distances (note that the distance could be zero, thus the reference reticle already has a characteristic data value). After all of the referenced sub-reticles are processed and all of the characteristic data is assigned to them, an output file is written using the basename followed by the extension 'reticle'.

Finally, if the reticle option is utilized, the reticle locations contained in the file specified on the command line are read into an array. To conserve memory and increase execution speed, Wtab deallocates the memory used in the creation of the wafer table then allocates memory to hold the referenced sub-reticle table. For this reason, the sub-reticle reference option must be used if the reticle option is used. Again, using Euclidean distances, the nearest neighbor to the reference reticles are located and the characteristic data is copied from the nearest neighbor into the reference sub reticle. When all of the characteristics and referenced reticles have been processed, a file is written to disk using the basename and 'wafer' as an extension.

8.2 Inverse Mapping through Exhaustive Search Program

Program "inv" implements an algorithm introduced in [32].

Program command line:

inv ⟨weights-file⟩ ⟨solution-file⟩ ⟨desired-output-file⟩

Program configuration file "inv.ini":

The following are the program configuration parameters. They are assigned values present in the configuration file. Each of the parameters, except for the first in the table, will be assigned a default value, if no value is provided in the file.

input_compact_set	Input vector entries range. (Radius of the input domain hypercube)
samples	Total number of samples generated from the PDF at each relocation
max_iterations	Total number of program iterations
error_threshold	Maximum RMS error for a minimum to be considered global
explosion_threshold	Maximum trajectory speed for a minimum to be considered local
tracing	Set to 1 to create file "tracing.txt" with Maple 3D trajectory trace
minima_details	Set to 1 to create file "detail.txt" with detected minima
initial_ip	Initial input vector(s)

8.3 Process Stages Models

Weights of the developed models are listed in this paragraph. The first line contains number of inputs, number of hidden neurons, and number of outputs of the neural network. Note that the number of inputs, and the number of hiddens do not include a bias unit. Afterwards, two matrices are listed. Thus the number of columns in the corresponding matrices is increased by one because a bias neuron is added to the hidden and input layer. In the listings the matrices' columns are presented in succession if they exceed a page width. In this case, dashed lines indicate the fact that a matrix was wrapped up. Also PCA matrices are listed, if used.

The SCRG-F model weights. (See description in Chapter 3, and Fig. 7.)

```
32 22 19
1.6869e+00 5.4208e-01 2.8061e-01 4.9037e-01 -1.0935e+00 1.2227e-01 4.0896e-01 4.1035e-01
2.1359e-01 -2.3581e-01 1.5130e-01 8.5681e-01 -9.8346e-01 -3.9503e-01 3.7790e-01 3.7792e-01
-2.7058e-01 -3.4106e-01 -1.8894e-03 7.4523e-02 -4.2765e-01 7.1626e-02 1.8509e-01 1.8495e-01
3.6622e-01 -5.5526e-04 5.0624e-03 -3.6851e-01 6.4059e-03 7.8836e-01 -2.3402e-01 -2.3330e-01
-8.2443e-01 4.5483e-02 -1.8358e-01 6.2453e-01 -6.6952e-01 -2.4643e-01 -1.0638e+00 -1.0650e+00
-1.3340e-01 -5.2017e-02 5.9423e-02 8.7749e-01 -1.1085e+00 7.1370e-02 5.5972e-01 5.5936e-01
-1.7409e-01 -3.9046e-01 4.1157e-02 1.7419e-01 -2.0229e-02 -5.1311e-01 3.0282e-01 3.0173e-01
3.2058e-01 -2.6701e-02 2.7672e-01 1.2301e+00 2.0604e-01 -2.1857e+00 2.3384e-01 2.3351e-01
3.9931e-01 8.2334e-01 -2.7575e-01 -7.1957e-01 6.3962e-01 5.3313e-01 3.1222e-01 3.1393e-01
5.4664e-01 -3.0897e-01 2.7648e-01 1.4511e-02 -3.1813e-01 3.3442e-01 2.4820e-01 2.4744e-01
-4.1055e-01 8.7991e-01 -4.0441e-02 2.7410e-01 -1.6665e-01 -5.3288e-01 6.4686e-01 6.4753e-01
2.1118e-01 -8.5973e-03 -2.5865e-02 4.3084e-01 -5.4944e-01 -2.0469e-01 -2.7909e-01 -2.7838e-01
-1.3939e-01 1.2047e+00 3.8534e-01 -1.5250e-01 5.2691e-01 -3.1473e-02 4.0456e-01 4.0412e-01
-6.2166e-02 -2.1027e-01 1.1560e+00 -4.4107e-01 5.5290e-01 4.4806e-01 7.7686e-01 7.7737e-01
4.1411e-01 7.6331e-01 -4.8539e-01 -1.2025e-02 -3.8200e-01 5.9501e-01 -3.6446e-01 -3.6674e-01
8.4919e-02 -1.1094e-01 3.5742e-01 5.0102e-01 -1.1864e+00 1.0306e+00 -2.2357e+00 -2.2383e+00
7.1496e-01 -6.6459e-02 8.4225e-01 4.9803e-01 1.0294e-01 -7.9632e-01 -5.1525e-01 -5.1591e-01
-5.3660e-02 -1.7975e-01 -2.3745e-01 6.0509e-02 1.8797e-01 -7.5908e-01 2.1954e-01 2.1966e-01
-1.0683e-01 -1.9457e-01 4.9324e-01 -7.9145e-01 2.9812e-01 1.3550e+00 -7.4440e-02 -7.3708e-02
-2.4825e-01 3.3248e-01 9.2573e-02 1.8243e-01 2.0856e-01 -6.4470e-01 -2.7121e-01 -2.7164e-01
6.0487e-01 6.3782e-01 -1.9762e-01 3.3982e-01 -3.7436e-01 1.2810e-01 -7.3869e-01 -7.3784e-01
9.7252e-02 7.5301e-01 -6.6526e-01 2.5883e-01 6.0879e-01 -1.3961e+00 -2.5089e-01 -2.5064e-01
-----------------------------------------------------------------------------------------------
8.0060e-01  -4.9365e-02 -4.1070e-01 1.3226e+00 1.7005e-01 -7.1899e-01 1.4763e+00 1.1367e+00
-6.8149e-01 9.8955e-01 1.1193e-01 -6.8468e-01 2.1954e+00 -3.4843e-01 6.4253e-02 -1.3254e-02
6.9354e-01  -6.4109e-01 3.5795e-03 4.2916e-01 -7.7051e-01 -1.4373e+00 2.1248e-01 5.1655e-01
1.0902e+00  -7.2162e-02 6.2506e-02 -7.0412e-02 1.7267e-01 -6.1101e-01 -2.7058e-01 2.7399e-02
-7.3818e-01 6.0613e-01 4.8744e-01 -8.0093e-01 -8.8806e-01 1.3300e+00 -7.0584e-01 2.4769e+00
-1.6278e-01 1.5392e+00 -6.2428e-01 3.2247e-01 -1.3580e-01 -7.3359e-01 -2.1565e-01 -1.0557e+00
-3.2056e-01 -3.2967e-01 2.0236e-01 2.3536e-01 -1.7186e-02 1.2706e-01 6.4150e-01 -8.7301e-02
-5.3570e-03 -4.0125e-02 8.4369e-01 -5.6384e-01 -1.7019e+00 -7.6722e-01 -2.7979e-01 7.4878e-01
-8.9703e-01 3.6301e-01 7.3949e-02 -1.7804e-01 -9.2356e-01 -4.7546e-01 2.0267e+00 1.9646e-01
2.6111e-02  7.3827e-02 -1.6580e-01 2.1171e-01 4.7668e-01 4.4553e-01 9.0726e-01 -6.5148e-01
-1.9628e-02 -9.4641e-03 -1.1439e-01 6.7869e-01 5.3354e-01 -6.6755e-01 8.9658e-01 -4.6307e-01
-1.4861e+00 3.4981e-02 3.0038e-02 4.1641e-01 -1.3566e-01 1.0670e+00 2.9757e-01 -4.1017e-02
-8.2192e-02 5.4507e-01 -2.7420e-01 -5.2260e-01 1.3902e-01 -3.3008e-01 -9.1525e-01 -1.0612e+00
6.2735e-01  7.3931e-01 -3.3984e-01 -2.4899e-01 2.0476e-01 2.2887e-01 -8.4203e-01 -9.3192e-01
-8.1281e-01 6.9390e-01 -4.9834e-01 -6.8091e-01 3.3630e-01 8.7303e-01 -5.6708e-01 1.8204e+00
```

```
9.1543e-02  -1.5392e+00 -1.2499e+00 -2.9387e-01 1.8217e+00 4.2206e-01 5.8859e-01 2.5194e-01
-4.1785e-01 2.3760e-01 -1.8858e-01 -5.4097e-01 5.2739e-01 6.8573e-01 -1.0633e+00 -7.2299e-02
2.1298e-03  -5.9073e-01 9.2185e-01 7.6601e-01 1.4046e-01 5.7979e-01 9.1696e-01 1.4432e-01
-1.5428e+00 6.4546e-01 8.1707e-01 3.3920e-01 -2.5986e-01 -3.7365e-01 -9.3843e-01 -2.1942e+00
-2.8255e-01 -4.7093e-01 1.2867e+00 6.5018e-01 1.2542e-01 -7.5167e-01 -1.2078e-01 4.5300e-01
-7.1501e-01 -8.3269e-01 -3.9645e-02 6.0260e-01 9.1448e-01 7.4217e-01 -1.5481e+00 -1.0749e+00
7.4118e-01  -2.7693e-01 -1.3656e-01 -1.0155e+00 8.8738e-01 -1.0048e+00 -8.3020e-01 6.5586e-01
-----------------------------------------------------------------------------------------
-9.7412e-02 -3.0135e-01 -2.6439e-01 9.8869e-01 5.8387e-04 6.6127e-05 1.0883e-03 -1.8188e-04
-4.6448e-01 8.9346e-01  -1.0236e+00 -1.1360e+00 5.7539e-04 -1.0904e-03 2.2729e-04 -9.0469e-04
-3.3293e-01 2.7124e-01  5.7587e-01 1.1787e+00 -4.1546e-04 -1.1866e-03 -3.6368e-04 4.5809e-04
-4.5970e-02 2.0100e-01  4.9449e-01 -3.3064e-01 2.6893e-04 -1.8767e-04 8.6606e-04 4.0103e-04
7.8493e-01 -1.2462e-01  -4.5149e-01 -2.1224e-01 -7.3432e-04 7.8540e-04 -6.0949e-04 4.7620e-04
-3.3772e-01 -5.2802e-02 7.2678e-01 -3.0288e-01 3.8483e-04 -7.8119e-04 2.4875e-04 5.4426e-04
-5.7916e-02 -1.9160e-01 -1.6000e-01 4.4897e-01 6.0083e-04 8.3721e-04 5.8353e-04 1.4772e-04
-4.3976e-01 -7.3895e-01 6.6307e-01 -4.8898e-01 4.6568e-04 4.0894e-04 4.6105e-04 3.4990e-04
-1.9622e-01 6.6527e-03  -2.0837e+00 -9.2901e-01 5.9115e-04 1.0425e-03 -3.5435e-04 -5.9074e-04
1.0874e-03 -2.4716e-01  -3.7877e-01 1.9657e-01 9.3291e-04 2.4171e-04 4.0571e-04 1.1009e-03
-2.3554e-01 4.7320e-01  -2.8878e-01 1.0262e+00 1.0483e-03 1.8452e-04 -2.5464e-04 3.7683e-04
5.9049e-01 -1.6299e-01  -5.5015e-01 1.5388e-01 3.4493e-04 -1.2021e-03 1.0136e-03 5.6216e-04
-3.1744e-01 2.2132e-01  4.0406e-01 -5.2031e-02 9.7849e-04 2.4617e-04 6.4974e-06 3.0857e-04
-6.6465e-01 6.6031e-01  1.3307e+00 -3.0192e-01 1.0736e-03 6.4095e-04 -3.6521e-04 6.1561e-04
1.0413e-01 -1.0587e+00  1.7918e-01 -1.4624e-01 -9.7566e-04 4.2011e-04 -5.5135e-04 -1.4570e-05
-3.9245e-01 3.1014e-01  -6.3268e-01 6.5383e-01 3.9811e-04 5.4121e-05 7.6746e-04 5.9926e-04
5.1952e-01 2.8218e-01   -1.2104e+00 -2.6608e-01 -1.0053e-03 7.5434e-04 -1.2128e-03 1.1319e-03
3.1183e-01 -1.7244e-01  3.9656e-01 -6.7726e-01 3.4487e-04 -2.6877e-04 -2.0243e-04 6.3200e-04
1.5220e+00 -3.1759e-01  1.0440e+00 -5.4526e-01 3.5634e-04 1.2512e-04 1.8235e-04 -1.0077e-04
2.3832e-01 8.1243e-01   1.5107e+00 -8.6209e-01 2.6390e-05 6.6839e-04 -5.9947e-04 1.2249e-04
-3.4163e-01 -7.6899e-01 2.6865e+00 4.5899e-01 7.1968e-04 5.5802e-05 8.4026e-05 -7.2555e-04
4.5147e-02 -1.0409e+00  -1.5395e+00 9.5720e-02 -1.4034e-04 -5.5305e-04 -3.7554e-04 -5.7408e-04
-----------------------------------------------------------------------------------------
6.6973e-04 -6.8143e-04 -8.5457e-04  2.9360e-04 1.7410e-04 -8.5561e-05 1.0826e-03 1.2415e-03
-1.1760e-03 -7.8220e-04 1.1762e-03  1.0437e-03 9.3678e-05 9.9664e-04 8.3964e-04 3.6049e-04
1.4839e-04 3.1323e-04 -4.9370e-04   -6.4279e-04 8.0496e-04 6.0838e-04 9.6861e-05 -8.8997e-05
7.6359e-04 5.8056e-05 -1.1994e-03   2.1550e-04 5.6998e-04 -3.4840e-04 8.2494e-04 5.6167e-04
3.1529e-04 -4.9330e-04 3.5719e-04   1.0515e-03 -3.3386e-04 -3.0265e-04 7.4429e-04 9.3954e-04
9.4277e-04 6.9816e-04 -1.5538e-04   9.5709e-04 9.0448e-06 -7.1742e-04 4.5115e-04 -4.7820e-04
-3.1547e-04 -7.8787e-05 -5.5721e-04 1.1910e-04 7.1475e-04 1.4707e-04 3.0624e-04 -1.2442e-03
-1.0432e-03 4.9355e-04 -1.0228e-03  5.4319e-04 4.1286e-04 4.2312e-05 -4.8741e-04 -7.4261e-04
-1.0440e-03 2.8300e-04 -3.1212e-04  -3.7126e-04 -1.2614e-04 -6.7165e-04 1.0102e-03 -9.8510e-04
-1.1564e-03 9.4190e-04 2.7557e-04   -1.0181e-03 -6.7509e-04 -4.5096e-04 -6.8897e-05 9.0693e-04
1.1788e-03 -4.9613e-04 -1.6057e-04  -1.1287e-03 -1.0824e-03 8.3972e-04 2.0161e-04 -4.4488e-04
4.1551e-04 1.1715e-03 -8.8088e-04   1.0557e-03 2.7259e-04 -5.3373e-04 -7.5136e-05 -6.7889e-04
-1.2947e-04 1.4271e-04 -5.8079e-04  8.1645e-04 -2.8224e-04 -4.8467e-04 4.3951e-05 3.2075e-04
4.3123e-04 -8.6764e-04 1.2254e-04   4.3386e-04 -1.0552e-03 -5.8984e-04 5.1118e-04 -4.9870e-04
-7.5497e-05 2.8537e-04 -5.4700e-04  -1.1564e-03 1.1992e-03 -7.5312e-06 1.2412e-03 -7.3818e-05
-6.5941e-04 -1.1398e-03 -7.9845e-04 1.5594e-04 1.3088e-04 -4.4397e-04 1.0807e-03 -8.3476e-04
6.0932e-04 -2.4128e-05 4.9868e-04   6.8316e-04 7.7723e-04 -1.1417e-04 9.5760e-04 -3.0026e-04
-7.9232e-04 4.9293e-04 -5.2076e-04  1.1700e-04 1.0706e-03 -1.0494e-03 2.7661e-04 8.2322e-04
-6.6131e-04 -6.1973e-04 -4.0855e-04 -1.0849e-03 -8.8331e-04 7.2277e-04 -1.0773e-05 -9.0357e-04
-1.2378e-03 2.1157e-04 7.0973e-04   5.3553e-05 4.4590e-04 -8.1829e-04 6.3553e-04 -3.4286e-05
-3.0823e-04 -8.6768e-04 1.1667e-03  -1.1386e-03 4.7971e-04 -9.3218e-04 -1.2403e-03 7.4086e-05
-2.1042e-04 -1.0520e-03 3.3013e-04  -2.3715e-04 -3.5672e-04 -5.3040e-04 -1.8743e-04 -3.3861e-04
-----------------------------------------------------------------------------------------
7.2350e-04
```

-1.6223e-04
-1.0409e-03
-1.1718e-03
7.8242e-04
5.0056e-04
-1.9399e-04
-2.7626e-04
-1.1370e-03
9.6200e-04
7.3300e-04
2.8809e-04
-5.2053e-04
4.4933e-04
-1.1110e-03
-4.1404e-04
-1.1844e-03
2.4103e-04
1.1023e-03
2.7077e-04
5.8862e-04
-3.5806e-04

-3.1610e-01 2.2963e-01 -3.2061e-01 4.0139e-01 -2.5229e-01 6.7411e-02 -2.2470e-01 5.6737e-01
1.3135e-01 -1.9666e-01 3.1505e-01 -6.0291e-01 -1.5215e-01 1.5086e-01 3.3689e-01 -6.8649e-01
7.9777e-01 -9.9903e-01 8.5484e-01 1.2175e-02 3.7383e-03 3.8433e-02 3.0846e-01 -7.5029e-01
1.2810e+00 -1.0755e+00 9.5149e-01 1.2312e-01 1.8941e-01 -1.4497e-01 1.3363e-01 -6.1215e-01
-8.5333e-01 3.3466e-01 -1.5219e-01 -2.0831e-01 -2.6858e-01 2.8478e-01 1.6885e-01 -1.1032e-01
-1.2681e+00 9.0306e-01 -8.8854e-01 -4.8392e-01 -7.5349e-01 3.5040e-01 4.7886e-03 2.9949e-01
9.7834e-01 -4.9863e-01 -5.6525e-01 -2.2713e-01 -1.7508e+00 2.0260e-01 -1.2169e-01 -2.5621e-01
7.9633e-01 -1.7670e-01 3.4876e-02 -1.0170e-01 -1.5208e-01 3.2056e-01 2.1581e-01 -5.4535e-01
3.7222e-01 -1.8382e-02 3.2154e-02 -3.8812e-01 1.6331e-01 -1.6856e-01 1.8122e-01 -4.0491e-01
-9.5562e-01 9.7988e-01 -7.4455e-01 1.1386e-01 3.8388e-01 -3.5766e-02 -1.4971e-01 3.4665e-01
-3.1480e-01 2.2704e-01 -3.1918e-01 4.0026e-01 -2.5092e-01 6.7657e-02 -2.2274e-01 5.6299e-01
3.6725e-01 -4.4933e-03 -4.3548e-01 5.3322e-01 4.4322e-01 -6.7127e-01 -1.6592e-01 -4.2390e-01
-5.1607e-01 9.2353e-01 -6.4104e-01 1.1141e+00 1.4149e+00 1.7260e-02 -3.0108e-01 8.5930e-01
-4.4256e-02 1.1463e+00 -8.4722e-01 6.1102e-01 1.4761e+00 7.5569e-01 -7.9455e-01 -5.8935e-01
-7.4489e-01 1.0154e+00 -5.5986e-01 2.8074e-01 1.3081e+00 1.8150e+00 -7.8982e-01 -2.4418e-01
5.4123e-01 3.6655e-02 4.6218e-01 2.8735e-01 -1.0901e+00 -4.1853e-01 -1.4736e-01 5.3113e-01
1.0794e+00 -1.8807e-01 -1.5114e-01 -7.3961e-02 1.2625e+00 1.4337e+00 2.3027e-01 8.9496e-01
-3.4730e-01 -3.0735e-01 7.1268e-01 1.3236e-01 1.6822e-02 1.4332e-01 -9.4403e-03 2.2367e+00
4.3771e-01 8.0042e-01 -8.4259e-01 -6.4652e-01 8.8339e-01 -5.9256e-01 7.2228e-01 -1.0574e+00

-2.4211e-02 -1.7723e-01 -5.0308e-01 -4.5861e-01 5.8653e-02 2.4867e-02 1.4401e-01 1.5803e-01
1.9146e-01 4.2500e-01 5.7145e-01 5.6363e-01 -6.6025e-02 -4.0483e-01 -1.6846e-01 -3.8258e-01
-7.4413e-01 2.3815e-01 7.2254e-01 4.5308e-01 -5.2058e-01 3.3899e-01 5.4023e-01 3.8105e-01
-1.1928e+00 1.6432e-01 5.4844e-01 2.1778e-01 -2.6578e-01 5.0970e-01 4.9913e-01 -3.9100e-02
8.2849e-01 2.6594e-01 2.7497e-01 2.9955e-01 -1.2784e-01 -4.2418e-01 -2.9018e-01 3.9465e-01
1.4406e+00 3.8365e-01 4.6825e-02 1.7243e-01 1.2958e-01 -9.2550e-01 -5.6086e-01 1.0550e-02
-2.5275e-01 8.0770e-01 -6.0508e-01 3.6497e-01 6.0278e-01 -2.7589e-01 2.4771e-01 5.2497e-01
2.2693e-01 6.3736e-01 -2.5691e-01 6.1962e-01 -3.3720e-01 -2.0194e-01 -3.8488e-01 2.7576e-01
1.7786e-01 2.0608e-01 1.6368e-01 4.4903e-01 1.4422e-01 -1.8575e-01 -5.1231e-01 -1.5200e-01
1.2200e+00 -2.0331e-01 -9.9143e-01 7.1442e-02 5.2685e-01 -3.7756e-01 -1.2020e+00 -4.1543e-01
-2.4480e-02 -1.7607e-01 -5.0112e-01 -4.5485e-01 5.7572e-02 2.6017e-02 1.4456e-01 1.5599e-01
1.4171e-01 2.8960e-01 4.8353e-02 -4.5726e-01 -4.7434e-01 -4.6233e-01 -1.6653e-01 4.8056e-01
-4.4651e-01 3.7913e-01 5.2414e-01 -1.6590e+00 1.2439e+00 1.3095e+00 -3.9290e-01 1.8710e+00
-3.5848e-01 -4.7924e-02 -3.0058e-01 -1.8070e-01 -2.6931e-01 -9.1687e-02 1.3009e+00 1.9244e+00
-1.3693e-01 -2.1503e-01 -1.9159e-01 2.7247e-01 5.6103e-01 1.4549e-01 1.0297e+00 3.2429e+00
8.7150e-01 4.4298e-02 -1.1457e-01 3.8687e-01 -8.6415e-01 -9.9676e-01 -3.2860e-01 -2.5255e-01
1.8051e+00 8.0692e-01 -3.7233e-02 -2.1421e-01 -2.5387e-01 -9.9399e-01 1.5926e-01 4.4309e-01
-6.3913e-01 -3.0885e-01 1.2243e+00 -4.2634e-01 3.0760e-01 1.3396e-01 -9.6195e-01 -9.0875e-01
1.3658e-01 9.8995e-01 9.7253e-01 3.2051e-01 -5.5443e-01 -3.0433e-02 1.9349e-01 -1.9895e+00

-3.7880e-01 4.3797e-01 -1.0686e-01 5.8808e-01 2.0278e-01 -1.7582e-01 3.4097e-01
2.6075e-01 -6.2766e-01 4.3738e-02 -1.0136e+00 -9.9367e-02 1.1122e-01 2.8442e-02
-8.5714e-01 6.7266e-01 5.9531e-01 1.8179e-01 8.4424e-01 -1.6981e+00 -7.6197e-01
-3.4835e-01 1.1392e-01 1.8469e-01 2.0409e-01 1.2285e+00 -1.2752e+00 -3.6082e-01
-5.3837e-01 4.5388e-01 1.0411e-01 -4.9799e-01 -6.1167e-01 -1.4594e-01 8.6582e-01
-6.6804e-02 -2.5401e-01 -2.5732e-01 -1.0971e+00 -1.2740e+00 9.9041e-01 7.6413e-01
1.2432e+00 -5.7881e-01 9.4102e-01 -1.5105e+00 -8.0781e-01 1.1834e+00 1.1312e+00
4.6708e-01 1.8084e-01 7.4230e-01 -6.3049e-01 -2.4591e-01 -4.3504e-02 5.7615e-01
5.8110e-01 -3.6522e-01 8.9097e-02 -6.5411e-01 -4.5789e-01 4.8740e-01 -3.6973e-01
6.8521e-01 6.2807e-01 -3.8450e-01 2.1125e-01 -1.3170e+00 1.2852e+00 6.2896e-01
-3.7706e-01 4.3660e-01 -1.0591e-01 5.8604e-01 2.0224e-01 -1.7347e-01 3.4746e-01
-3.3006e-02 3.9988e-01 -4.0592e-01 6.2980e-03 -2.7776e-01 1.8713e-01 1.1528e-01
-9.2744e-01 -4.7695e-01 -1.2684e+00 -6.3585e-01 1.2271e+00 -7.4696e-01 -9.2228e-01
1.4432e-01 -6.5728e-01 3.0848e-01 -3.7142e-01 -9.6465e-02 -8.4528e-02 8.4407e-01
-1.0119e-01 -9.1115e-01 1.6482e+00 -4.0698e-01 3.0852e-01 -8.1419e-01 6.8100e-01

```
-3.7844e-01 2.7015e-01 4.3872e-01 2.6185e-01 -6.5742e-01 2.3042e-01 9.1138e-01
-1.2311e+00 4.2498e-01 1.1775e+00 -7.0913e-02 -1.5437e+00 -1.1355e+00 7.9473e-01
-6.3133e-01 -4.2571e-01 1.1527e+00 -1.4127e-01 1.4629e-01 3.9468e-01 5.9494e-01
-1.9926e-01 1.1836e+00 -9.1188e-01 -8.3045e-01 2.2074e-01 -1.7212e-01 -8.7871e-01
```

The S-F model weights. (See description in Chapter 3, and Fig. 9.)

```
10 22 8

-0.127055 7.49567 -0.634164 -1.11597 0.760231 1.38051 1.84404 1.85738 3.91031 -6.75478 -4.13089
-1.15991 4.21863 -2.07955 -1.2798 0.950087 1.46787 -1.31505 -1.33188 -0.286187 -4.02435 0.549858
0.60328 -5.38827 -0.630234 -0.134736 0.768164 -1.35755 1.68847 1.6799 -2.92909 0.0659604 -2.34425
2.65322 -3.19452 1.18069 0.162061 -0.0773701 -0.32567 0.891966 0.890681 0.839008 0.450627 -0.382486
0.247585 -1.69633 -2.93716 2.43155 -1.67145 -3.1637 3.74834 3.75101 -3.23694 6.38808 -1.09515
0.22658 4.68003 -3.00753 0.952964 -0.723596 -1.02982 1.43192 1.41785 1.23459 0.915274 -0.720082
-0.473219 1.02345 0.213825 -0.13452 0.180426 0.0407675 0.694132 0.693798 -1.04415 0.354356 0.499311
-0.520337 0.953973 2.62134 -0.668972 0.909709 -0.162397 3.16577 3.16585 0.460676 -4.30974 -2.22326
2.15827 -2.60255 0.563874 -0.827938 0.516267 1.11137 1.47518 1.47241 2.93692 -1.39766 0.0694464
2.30729 -2.00734 0.0245156 -0.352291 -0.163611 1.10656 -0.313612 -0.315663 -4.94008 0.962241 -0.508625
5.73512 4.04712 -3.36963 0.977273 -0.941597 -0.700782 -0.405551 -0.393342 -1.64023 0.434824 -2.66831
5.19099 2.78447 -0.362627 0.271568 -0.334927 -0.0836266 1.27092 1.27433 3.01069 -2.26078 -1.01064
0.523946 -2.45347 -4.01896 -1.72962 1.62482 1.40949 -1.15301 -1.15025 -1.7498 -3.34594 0.211599
2.31924 3.1309 -1.96561 2.04741 -1.44658 -2.38945 1.29298 1.29566 3.04474 4.93033 2.99175
2.61189 3.04501 -2.76416 0.21846 0.0243123 -0.547823 0.746666 0.740167 -2.27756 0.68891 -0.771956
0.207378 2.45468 2.56337 -0.687311 0.487981 0.677899 0.78158 0.777844 2.56758 -5.04673 -4.44225
0.251039 1.45833 1.54186 0.40725 0.168889 -1.17037 -0.719205 -0.729095 -0.687839 1.30444 -0.534128
2.0877 -0.142601 -4.52805 0.557334 -0.726439 -0.116063 -0.779162 -0.767457 -1.78103 9.09966 -0.226484
4.10106 -0.297321 -0.852439 -0.548436 0.535107 0.647142 0.203496 0.218735 2.69212 3.81538 -2.26511
-3.71711 -5.48733 0.978903 -1.95211 1.50749 2.09638 4.19544 4.1888 -2.32703 3.057 -3.17475
-1.76971 -5.0853 3.08085 0.765646 -0.843035 -0.190069 3.70928 3.71828 0.813826 5.12449 0.939194
-3.63085 2.93406 -3.98303 1.76724 -1.3841 -1.80379 2.20375 2.20716 -8.2196 10.0617 1.69674

-1.41056 -0.819631 0.0379508 0.110874 -2.01699 1.0654 0.11714 -0.737377
1.34006 1.42701 0.183514 -0.925341 4.05214 -0.98637 -0.247058 1.77803
0.562716 -0.629391 -1.2579 1.23881 1.14996 0.694119 -0.677164 -0.624838
1.34305 -0.0913979 -1.07282 0.41304 0.815762 0.191513 -0.642445 -1.12039
-0.594187 -0.0840592 0.159672 0.507175 2.14983 0.42006 0.284317 1.57252
-0.716556 1.21912 1.10391 -1.12946 3.55211 -0.307975 0.226648 3.08283
1.03531 0.701157 0.24615 0.235559 1.41968 -1.55727 0.650739 1.10568
-1.57538 -0.410763 1.13183 0.955036 -2.34565 -0.320394 -0.436531 1.24035
-----------------------------------------------------------------------
1.78814 0.191725 0.274019 -1.4969 -0.225139 1.21956 -0.23358 1.98136
-2.5284 1.10437 -0.642994 2.15314 0.317377 -1.34705 0.376865 -0.985477
-0.378008 0.449059 -0.366897 -0.74713 1.19745 -0.173506 1.44906 2.00464
-0.691011 0.428664 0.807154 0.298445 -0.0294774 -1.10634 0.278701 -0.436113
-0.695634 0.614727 -2.11233 -0.0921767 1.9475 1.00577 1.40695 2.7747
-1.48944 1.44021 -1.86725 1.23937 0.68504 0.444506 0.644304 1.70878
-2.4431 -0.833274 -0.573073 1.71236 -0.229447 -1.41252 0.703258 -2.10961
-0.959241 -2.03033 -0.956981 -1.27424 -0.487827 0.942531 1.63479 -0.307449
------------------------------------------------------------------------
-0.407778 0.474981 2.09619 -1.28273 -0.55461 1.62203 0.812927
0.0514394 -0.785429 -3.40052 2.07075 0.707884 -2.94254 1.02078
-0.541465 -2.47947 -1.3644 1.40248 0.806213 0.00203451 1.67669
-0.323574 -1.49327 -1.29357 1.16982 1.90257 -0.514515 0.75554
-0.285395 -1.50106 -1.13168 0.694205 -1.52813 -0.519244 2.5783
-0.456586 -0.0130976 -1.84245 0.767391 -1.72713 -2.12588 2.18016
-0.607895 -0.218026 -1.51166 0.386387 -0.0612476 -1.31173 -1.41075
0.14689 0.645187 3.2747 -1.46571 -1.55057 1.09223 -1.10574
```

The CR-F model weights. (See description in Chapter 3, and Fig. 11.)

```
8 22 8
-1.3085e+00 4.1527e-01 -4.9996e-01 4.8519e-01 8.5673e-01 8.2592e-01 -2.3364e-02 6.4216e-01 -1.2851e+00
-2.3064e-01 -2.9761e-01 4.3545e-02 -8.6807e-03 3.5919e-02 3.4539e-01 -2.6283e-01 1.2759e-01 1.0429e-01
-1.0123e+00 -2.5042e-01 4.6837e-02 -1.6593e-01 1.6497e-01 1.0754e-01 1.9853e-01 -1.1409e-01 -1.1305e-01
1.5470e+00 -2.8286e-01 2.2316e-03 3.9544e-01 -8.3460e-01 -1.0329e+00 2.8950e-01 -7.5500e-01 -2.8285e-04
-3.0221e-01 -4.8660e-01 7.1822e-02 5.8579e-02 -1.6121e-01 2.9125e-01 -5.5867e-01 -2.0067e-02 5.7372e-02
1.6665e-01 2.4806e-01 1.0430e-01 -1.4591e-01 1.0502e-01 2.4912e-01 4.1506e-01 1.6711e-01 2.2581e-01
-7.4664e-01 -3.8673e-01 1.9634e-01 -3.2986e-02 3.4573e-01 -2.3749e-01 9.1658e-02 -3.7710e-03 -9.2779e-02
5.5947e-01 1.0772e+00 1.0832e-01 -1.4477e-01 -5.4242e-01 -1.1024e-01 3.5523e-01 3.4899e-02 1.7341e-01
-3.3696e-01 -1.0416e-01 9.9488e-03 7.1025e-02 4.3416e-01 -4.2271e-01 2.1227e-01 7.2428e-02 -1.0793e-01
9.8408e-01 6.8099e-01 -2.3465e-01 -1.4657e-02 -6.5557e-01 4.0963e-01 2.1560e-03 -6.3293e-02 -2.9714e-02
1.0761e+00 1.1660e+00 9.2165e-01 -8.0447e-02 -8.3345e-01 -1.4926e+00 -2.0338e-01 -4.5279e-01 1.1001e+00
3.3478e-01 -4.8752e-02 7.4439e-01 -4.3029e-01 -4.0002e-01 2.3346e+00 -5.3189e-01 -1.7633e+00 -1.8205e+00
-1.1689e-01 -2.7517e-01 5.1753e-02 -6.9093e-03 -3.1421e-01 3.1552e-01 -4.2123e-01 -7.1079e-02 8.7639e-02
-9.4620e-01 -5.0233e-01 1.9645e-01 -1.1489e-01 2.7971e-02 -2.3217e-01 1.4053e-01 -2.7875e-01 -1.5328e-01
-3.1704e-02 1.1657e-01 1.1217e-01 2.8498e-02 -7.3922e-01 -1.7413e-01 -5.6170e-01 -4.7003e-01 -2.3646e-01
9.2502e-01 3.9601e-01 -6.2415e-02 1.7474e-01 -6.9695e-01 -2.6133e-01 -7.1535e-01 1.8393e-01 1.3352e-01
-2.8436e-01 -3.6855e-01 -2.0884e-01 3.4778e-01 6.7144e-01 -3.6958e-01 -2.1191e-01 2.4428e-01 6.3322e-02
1.4776e-01 1.8555e-01 1.2818e-01 -1.9532e-01 -8.4283e-02 3.0986e-01 3.2126e-01 7.6469e-02 1.9470e-01
1.1530e+00 1.7979e+00 -2.1409e+00 7.1231e-01 -1.3149e-01 -1.9106e+00 5.4954e-01 -9.8270e-03 4.8808e-01
-7.6099e-01 -3.5141e-01 2.7333e-01 -6.4724e-02 4.6243e-01 -2.6881e-01 1.2852e-01 1.1442e-01 6.0353e-03
7.8154e-03 1.0228e-01 3.1349e-02 8.3215e-02 -4.1447e-01 -3.0586e-01 -3.0943e-01 -3.4594e-01 -2.5241e-01
-8.7937e-01 -9.3131e-02 1.9187e-01 -3.3264e-01 6.8293e-01 4.3054e-01 1.0443e+00 -1.0816e-01 -5.8095e-03

-6.3846e-01 -5.4077e-02 -5.3975e-01 1.9549e-01 9.4009e-02 -1.8678e-01 -4.5271e-01 1.3990e-01
4.4009e-01 -1.8695e-02 9.0300e-01 -3.4669e-01 -4.2904e-02 -4.6317e-02 6.6820e-01 -2.3579e-01
1.1776e+00 -4.8028e-01 1.5082e-01 -8.3630e-01 -6.2157e-01 2.1350e-01 1.0207e-02 6.6404e-01
1.0252e+00 -2.6125e-01 2.9964e-01 -9.0724e-01 -3.7762e-01 1.4417e-01 -7.5815e-03 5.9115e-01
1.0569e-01 -3.4490e-01 1.5979e-01 -4.1644e-02 -2.7762e-01 -2.0522e-01 3.5368e-01 -1.3478e-01
-7.1237e-01 7.5612e-03 4.2368e-01 2.6330e-01 2.8225e-01 -5.3325e-01 4.2602e-01 -7.3880e-01
2.4794e-01 3.7074e-02 3.0844e-01 8.6107e-02 -1.0792e-01 9.0499e-02 4.9795e-01 -5.8888e-01
-3.1206e-01 -2.0041e-01 -4.5001e-01 1.2592e+00 -4.1624e-01 1.3544e-01 -2.1308e-01 8.2736e-04
-----------------------------------------------------------------------------------------
-2.8901e-01 5.9986e-01 7.5062e-01 2.0975e-01 1.5122e-01 -4.3764e-01 4.3482e-01 8.8574e-01
4.2177e-01 -7.6050e-01 -4.0726e-01 -9.2888e-01 -1.3110e-01 9.1335e-01 -8.0266e-02 -8.4468e-01
1.7783e-01 2.7868e-01 4.6807e-01 -3.6725e-01 -4.3836e-01 1.3128e-01 1.1341e-01 -1.5699e-01
4.9811e-02 1.7520e-01 8.2500e-01 7.3692e-01 -2.2781e-01 1.1888e-01 8.6072e-02 -1.6260e-01
4.7089e-01 -2.7729e-01 -2.2385e-01 -1.3451e+00 -3.6135e-01 3.2322e-01 1.8539e-01 9.5008e-02
2.8451e-01 -6.2107e-01 -4.8002e-01 -2.0400e+00 1.0213e-01 6.4880e-01 4.3402e-01 -2.7652e-01
4.3754e-01 -8.8115e-01 -9.8837e-01 -3.0553e-01 -2.1949e-01 4.1576e-01 -6.5217e-01 -9.9404e-01
2.7428e-01 2.2905e-01 -1.3175e+00 -6.5693e-01 -4.3262e-01 -5.3080e-01 -5.7849e-01 1.5647e-01
-----------------------------------------------------------------------------------------
-1.3548e-01 -1.0629e-01 -3.2037e-01 -5.2980e-01 2.9425e-01 -1.0945e+00 8.2669e-02
1.4783e-01 -5.1494e-02 3.2701e-01 6.7662e-01 -7.3155e-03 9.0304e-01 -2.8969e-01
-4.2594e-01 1.7678e-01 1.2825e+00 -1.5214e-02 2.0179e-01 4.1632e-01 -1.0381e-01
-3.0353e-01 1.4032e-01 1.0439e+00 -2.0296e-02 9.8248e-02 3.8299e-01 8.7224e-01
2.4910e-01 -2.8146e-01 -1.1657e-01 3.2828e-01 3.1558e-01 -2.1686e-01 -6.3865e-02
4.6385e-01 -4.8108e-01 -8.3500e-01 3.1995e-01 4.1195e-01 -9.4128e-02 -1.0561e+00
5.0057e-01 -2.6914e-02 -6.0055e-02 5.8598e-01 -3.6755e-01 9.8486e-01 -2.1816e-01
4.0895e-01 -6.0781e-02 -1.6532e+00 -1.7875e-01 -2.6827e-01 -1.8944e-01 1.7007e-01
```

The G-F model weights. (See description in Chapter 3, and Fig. 13.)

```
8 22 8
3.9635e-01 -1.1819e-01 1.1990e-01 4.7297e-02 6.8424e-01 1.6867e-01 4.0627e-02 1.7927e-01 -2.7389e-02
-2.1120e-01 2.6307e-02 2.1027e-01 8.3578e-01 3.4510e-01 -4.0099e-01 2.1977e-03 -2.4642e-02 1.1594e-03
2.1876e-01 3.8927e-02 -9.2947e-02 -5.6541e-01 -1.7004e-02 3.2886e-01 6.4114e-02 -1.0621e-01 -1.1613e-01
-3.9589e-02 -3.1215e-02 1.1050e-01 1.9587e-01 2.6939e-01 -1.3752e-01 -6.1360e-02 1.1957e-01 -7.8847e-02
9.7795e-02 -1.9530e-02 1.5257e-01 -1.2097e-01 5.7991e-01 1.0005e-01 -6.4069e-03 2.3650e-01 -1.8350e-01
5.9582e-01 -3.7518e-01 1.0197e-01 -7.7376e-01 1.3242e-02 3.4225e-01 6.5556e-02 3.5188e-01 1.2881e-01
-3.7926e-02 1.9661e-01 -3.2178e-01 7.3142e-01 4.3288e-02 -7.5840e-02 -1.0504e-01 -4.0203e-01 -1.5957e-01
5.8112e-01 -7.2778e-01 5.4264e-01 -2.0183e-01 -6.3314e-02 -5.6710e-01 1.7305e-01 5.0205e-01 3.5616e-01
-3.7490e-01 4.8161e-01 -3.4614e-01 1.0624e-01 2.2670e-01 4.1036e-01 -1.2606e-01 -9.0822e-02 -1.9868e-01
8.8955e-01 -6.3866e-01 2.4577e-01 -2.1785e-01 -4.8934e-02 -1.8940e-01 1.9147e-01 4.1616e-02 2.3956e-01
-3.3199e-01 -5.9074e-02 5.4182e-02 6.0052e-01 -3.6162e-01 -3.5437e-01 5.5149e-02 1.8272e-01 5.2306e-01
-1.1728e+00 2.5392e-01 -2.7886e-01 1.7219e-01 -8.8406e-01 -3.4586e-01 -7.4348e-03 3.1259e-01 1.3323e-01
-3.5564e-01 4.9990e-01 -7.3030e-01 3.0784e-02 -1.6989e-01 8.1115e-01 -1.5908e-01 -1.7856e-01 -3.5237e-06
1.7257e+00 -7.1754e-01 1.1330e+00 8.5317e-01 1.7983e+00 7.1216e-01 2.3707e-01 6.6065e-02 9.2922e-01
-3.8516e-01 4.2403e-01 -2.3156e-01 3.3791e-02 2.9117e-01 3.6979e-01 -1.1201e-01 6.1148e-02 -1.4508e-01
7.0369e-02 7.1846e-02 -9.8501e-02 -4.8885e-01 -2.5226e-01 3.0094e-01 1.1959e-01 -1.1636e-01 5.6178e-02
-8.0703e-01 1.4423e-01 4.9372e-01 -5.8113e-01 -1.7023e-01 -8.2778e-02 2.0693e-03 6.8994e-01 3.8636e-01
-1.7019e-01 2.8330e-01 -1.3074e-01 -1.2224e-01 5.3078e-01 4.4328e-01 -3.5626e-02 2.7384e-01 -6.5892e-02
1.6874e-01 5.7528e-02 -3.2322e-01 -7.7812e-01 -2.2891e-01 3.6952e-01 -6.9956e-02 -5.9895e-03 -1.6553e-01
2.0956e-01 -5.9177e-02 -1.9887e-01 -3.6298e-02 -5.5850e-01 -1.7990e-02 7.4707e-02 -4.3019e-01 1.9233e-02
5.8463e-01 -4.8865e-01 4.1125e-01 1.6040e-02 9.5876e-02 -5.9424e-01 1.6604e-01 -1.1970e-01 -4.9299e-02
-4.8285e-01 3.6944e-01 -1.3890e-01 2.0838e-01 2.3477e-01 1.2599e-01 -1.4574e-01 1.1193e-01 -1.1955e-01

1.4292e-02 -1.0295e-01 7.3342e-02 -3.1660e-02 -5.7749e-02 3.9225e-01 -5.6376e-02 4.1743e-01
1.0504e-01 1.7069e-01 -1.4951e-01 2.8328e-02 -2.8861e-02 -3.6544e-01 4.5119e-01 -5.4409e-01
6.3306e-01 -5.2888e-02 1.2855e-01 2.6707e-01 5.7824e-01 -1.3960e-01 1.3071e-01 -2.6518e-01
3.7042e-01 -4.1932e-01 3.1155e-01 1.4432e-02 2.5531e-01 2.5779e-01 9.0984e-02 -4.9789e-01
4.1041e-01 6.7901e-01 -2.8793e-01 2.8348e-01 3.1744e-01 -6.3809e-01 4.2669e-01 -6.8437e-02
1.9723e-02 6.2713e-01 -4.2148e-01 8.7752e-02 -1.9452e-01 -5.4004e-01 6.5917e-01 -2.6533e-01
-1.1651e-01 8.0553e-02 -1.2774e-01 -1.9135e-02 -9.4830e-02 -3.6467e-01 1.4181e-01 -5.1476e-01
-1.4919e-01 4.5215e-01 -5.0964e-01 1.7517e-01 -3.5293e-02 1.0411e-01 -2.9341e-01 8.4315e-01
-----------------------------------------------------------------------------------------
-3.5037e-01 6.3959e-01 -1.2006e-01 -3.6800e-01 -5.0586e-01 -3.0729e-01 -3.8044e-01 1.1753e-02
4.6687e-01 -4.6186e-01 1.7923e-01 1.5995e-01 8.2145e-01 9.8979e-01 4.1245e-01 -1.1242e-01
3.6393e-01 -4.7969e-01 -4.7636e-01 -8.2896e-01 9.6131e-03 1.0851e+00 4.1218e-01 -2.0845e-01
3.5227e-01 -2.7041e-01 -4.8293e-01 -7.2434e-01 5.9556e-01 6.0364e-01 3.1420e-01 1.0210e-01
1.7596e-01 -2.0772e-01 5.5982e-02 -3.0170e-01 -2.5560e-01 8.8961e-01 1.9216e-01 -4.3202e-01
8.6378e-02 1.4111e-02 4.1512e-01 2.0479e-01 2.6490e-01 9.6098e-01 -3.3355e-02 -3.5664e-01
 3.1280e-01 -5.5907e-01 2.4797e-01 4.8874e-01 6.8654e-01 2.9159e-01 2.9290e-01 -8.4852e-03
-3.6904e-01 1.7087e-01 8.1458e-01 1.1269e+00 -7.0844e-01 -1.0377e+00 -2.1543e-01 -3.4794e-01
-----------------------------------------------------------------------------------------
-5.8112e-01 -3.6364e-01 3.2398e-02 2.4435e-01 6.5241e-01 -3.8798e-01 2.2227e-01
-2.3613e-01 3.9065e-01 -6.9632e-02 -8.6898e-02 -7.0972e-01 3.6633e-01 -9.6389e-02
1.3383e-01 5.4889e-01 3.0631e-01 -6.1262e-01 -1.7773e-01 4.4494e-01 -2.3001e-01
-1.3344e-01 3.6893e-01 5.1465e-01 -1.7309e-01 -4.7198e-01 1.9743e-01 3.4804e-01
-3.4340e-01 2.7221e-01 -3.9601e-01 -3.8758e-01 2.0279e-01 2.8684e-01 4.1204e-01
-1.0733e+00 -1.1974e-01 -5.3997e-01 1.9194e-01 7.8757e-02 3.3267e-03 -2.7296e-01
3.4084e-01 2.1416e-01 -7.2174e-02 -7.0305e-02 -6.7878e-01 2.8966e-01 -2.4811e-01
6.5255e-01 -1.4509e-01 -6.3322e-01 -3.1334e-01 3.0072e-01 1.8011e-02 3.2765e-01
```

PCA matrix; SCRG stage. (Equation (16), $m = 5$, and Fig. 22a.)

```
0.159618 0.1654 -0.20805 -0.037133 0.133547
0.0274142 -0.066437 -0.156469 -0.221949 0.223136
0.198793 -0.0214514 -0.112075 0.0917318 0.0187815
-0.201743 0.0171548 0.299546 0.0636182 0.169901
0.199416 -0.00830103 -0.311838 -0.0612838 -0.167288
0.20092 -0.0247309 -0.272641 -0.0615542 -0.169591
0.216538 -0.192759 0.0468129 0.0770674 -0.023721
0.216535 -0.192758 0.0468053 0.0770583 -0.0237362
0.0673581 0.0251121 0.199947 0.241355 -0.568132
0.229409 -0.138192 0.0402738 0.0766758 -0.0266697
0.230351 -0.034788 -0.00656976 -0.10542 0.12013
-0.208198 -0.228492 -0.0417592 -0.0523944 -0.0670174
-0.0473709 -0.306077 -0.0189103 0.249304 0.192866
-0.147102 0.241517 0.019117 -0.249082 -0.0925428
-0.238652 -0.0498101 -0.0689535 -0.0944931 0.0384424
-0.0255072 0.452178 0.0659993 0.0776087 -0.101866
0.235879 -0.0533121 0.00997333 -0.024467 0.0635341
-0.1359 -0.302628 -0.0375417 0.163734 -0.22698
0.234158 -0.121609 0.0351651 -0.0805856 0.0623604
-0.241237 0.0192555 0.00707626 0.0802587 -0.0219575
-0.132543 -0.282246 -0.245222 -0.02386 -0.090954
-0.113967 -0.0613329 -0.257825 0.413421 0.145473
-0.145925 -0.303864 -0.0516858 -0.282828 -0.193167
-0.166508 -0.165764 0.124788 -0.236456 -0.126056
-0.223048 -0.0504161 -0.110932 0.121985 0.053571
0.0458048 0.123144 -0.235484 0.448119 0.0450694
0.0433314 -0.190453 0.123731 -0.0367641 0.473028
0.213087 -0.00731362 0.146798 -0.254939 -0.0786421
0.234207 -0.121608 0.0356842 -0.0800134 0.0622083
-0.000240707 -0.021803 -0.494732 -0.15192 -0.167482
-0.232829 -0.0866261 -0.0963052 -0.112733 0.0676589
-0.0983638 0.265998 -0.305274 -0.168465 0.177073
```

The SCRG-F model weights (reduced). Neural network. (See Fig. 24c.)

```
5 22 8

-0.212291 0.636242 0.405612 -0.102343 0.0270011 -0.369558
0.903363 0.12816 -0.0382012 -0.149195 0.214659 -0.0812161
-0.0743613 0.0324109 0.252557 0.30627 0.0546047 0.499319
-0.287316 0.559932 -0.227487 0.333479 -0.117488 -0.238845
-0.307986 0.538424 -0.226737 0.798282 -0.0476976 -0.710568
1.14593 -0.0404718 0.316596 -0.407218 0.000441096 0.375826
1.018 -0.0805636 0.145096 -0.112132 0.119646 0.152085
0.181288 0.0499232 0.268569 0.276859 0.492209 0.191066
0.104006 0.0885126 0.186664 0.16187 -0.0256453 0.473498
-0.096658 -0.0690513 0.188202 -0.0330042 0.224672 0.0736774
-0.395434 0.38333 -0.623982 0.472178 0.135439 -0.442659
-0.167439 0.164721 0.148204 0.337246 0.500054 0.241981
-0.102946 0.177708 0.301189 0.46795 0.436698 0.0395145
0.727314 -0.0738578 -0.0209484 0.06633 0.0216482 0.0743925
1.11485 -0.41716 -0.0247441 0.160373 -0.229027 0.37826
-0.358188 -0.39476 0.0710548 0.129131 -0.0376599 0.292263
0.00361767 0.349727 0.194663 0.0278302 -0.177939 -0.0587332
```

-0.125872 -0.216326 -0.463101 0.184691 -0.146161 0.184314
0.943456 0.0991569 -0.0711293 -0.0641769 0.0260406 -0.152439
-0.203919 -0.461441 -0.166517 0.100423 0.398138 0.440507
-0.316787 0.288533 0.407736 0.648976 0.102212 -0.235448
-0.00190544 0.380926 -0.449434 -0.00912081 0.126827 0.104384

-0.195977 0.435567 -0.0901558 -0.146616 0.0276031 0.536281 0.356496 0.120917
0.0339527 -0.466619 0.114645 0.288451 0.0155207 -0.325813 -0.420136 -0.305594
-0.323832 -0.250982 0.0559532 -0.067652 -0.350649 -0.0531015 -0.270405 -0.148937
-0.106578 -0.32596 -0.10491 -0.208191 -0.511 -0.245762 -0.446608 -0.249532
-0.545234 -0.262848 0.0276503 -0.167682 -0.203069 -0.418072 -0.155148 -0.0950647
-0.336341 -0.048784 0.0839886 0.0847236 0.110685 -0.388709 -0.0311192 0.0237574
0.16692 -0.365726 -4.33256e-05 0.253613 0.153416 -0.384201 -0.273329 -0.120136
0.339574 0.478448 0.368883 0.671932 0.867826 0.804138 0.627971 0.156716

0.0151154 0.0551714 -0.022073 0.146392 0.0635988 0.319245 0.645183 -0.110362
0.0972652 0.0585059 0.0896727 -0.0449502 -0.150915 -0.332349 -0.638215 0.397588
0.00326537 -0.0657499 -0.219861 -0.158518 -0.222998 -0.322487 -0.163951 0.0771703
-0.140745 0.0854018 -0.419018 -0.264365 -0.27886 -0.323662 -0.388028 0.234225
-0.114116 -0.00375573 0.18593 -0.0689891 -0.109548 -0.1502 -0.0827298 0.191427
0.105119 -0.0182831 0.534822 0.159157 0.110316 -0.0820985 -0.0778293 0.215018
-0.0707344 0.0968339 0.0422657 -0.0623928 -0.0259471 -0.352804 -0.56536 0.0866984
0.413761 -0.139295 0.463275 0.357941 0.312192 0.442253 0.516712 -0.0223553

-0.0475867 -0.00338068 0.390094 0.00325717 -0.151858 0.169337 0.546145
-0.0165077 0.279507 -0.454476 0.285402 0.0207274 -0.144152 0.266908
-0.198177 0.00671569 -0.405321 0.0704166 -0.252428 -0.23842 0.749672
-0.12581 -0.207311 -0.344315 0.0266495 -0.342227 -0.356034 -0.183315
-0.326307 0.433979 -0.310711 0.410361 -0.133302 -0.0690613 0.778983
-0.231717 0.522134 -0.135377 0.546072 0.116935 0.170891 0.497744
0.159382 -0.0138245 -0.17354 0.143467 0.229118 -0.0717348 -0.487123
0.458816 0.0866252 0.599321 -0.405217 0.669167 0.483067 0.717777

PCA matrix; S stage. (Equation (16), $m = 5$, and Fig. 22b.)

```
-0.264551 -0.380696 0.357033 0.0365137 -0.644538
-0.0721472 -0.368168 -0.815661 -0.29301 -0.307266
-0.336766 -0.0177291 0.181848 0.276045 -0.350562
0.371801 0.20131 -0.102648 0.249098 -0.296323
-0.366919 -0.210447 0.0946907 -0.249582 0.29127
-0.369562 -0.179984 0.115246 -0.242492 0.283724
-0.358147 0.280825 -0.225442 0.251732 0.00917964
-0.358147 0.280827 -0.225435 0.251699 0.00915198
-0.101235 0.622434 0.107375 -0.690701 -0.334694
-0.367651 0.238431 -0.167917 0.222887 0.0439469
```

The S-F model weights (reduced). Neural network. (See Fig. 25c.)

```
5 22 8
0.697504 -0.24782 0.307243 -0.718174 0.12272 0.000487883
0.122029 0.3598 0.0179291 0.0133144 0.221071 0.143659
1.03337 0.161341 -0.423966 -0.15342 0.633621 1.28447
0.568244 -0.163385 0.368944 -0.130907 0.294691 -0.0673799
0.974239 -0.0885513 -0.00176282 -0.0487995 0.266608 0.411273
0.342481 0.275051 -0.00893971 -0.00750058 0.15909 0.13086
0.706205 -0.126728 1.06392 -0.687881 0.942089 -0.0704915
0.354673 0.0498161 -0.086518 0.0504487 0.324221 0.520288
-0.854349 0.934866 -1.67027 1.20199 0.780992 1.45137
0.22385 0.33943 0.29262 0.468907 0.367417 0.418175
-0.121911 0.339476 0.229816 -0.05271 0.306521 0.564779
0.0128956 0.282351 0.501752 0.334301 0.476524 0.585289
0.301501 0.027116 0.0406484 -0.0446508 0.233116 0.192856
-0.0819414 0.296866 -0.103295 -0.0723208 0.436073 0.246568
0.362963 0.175641 0.206965 -0.019416 0.0936066 0.207839
0.681524 0.195154 0.305505 0.229221 0.0591768 0.251938
0.973396 0.0672387 -0.496074 0.386353 0.100236 0.262359
0.915677 0.191142 -0.58008 0.33615 0.465822 1.02115
0.0325922 0.347395 0.218526 0.232087 0.271079 0.265481
0.24539 0.472178 1.12563 -0.106211 -1.14426 0.269883
2.37044 -0.766526 0.0536006 -0.811114 -0.308816 -0.48794
-0.441332 -0.208276 -0.00286629 0.207217 0.419046 0.581569

-0.200506 -0.0574623 -0.120363 -0.171098 -0.243649 -0.186911 -0.335434 -0.0713263
 0.308289 0.0739632 0.364572  0.132314    0.18511 0.100939 0.260042 0.165147
 0.220789 -0.0476738 0.42208 0.036052     0.0703765 -0.0486274 0.36496 0.118075
 0.26718 -0.096129 0.425436 0.22008       0.189265 -0.0315244 0.518529 0.073002
 0.396751 -0.0312784 0.376982 -0.0480771 -0.0633065 -0.0450011 0.149057 0.0634867
 0.133049 0.13082 0.0694703 -0.097261    -0.0588035 -0.0595977 -0.221344 0.173279
 0.241547 0.0618139 0.0503931 0.164598    0.151097 0.0479629 0.180861 0.0607325
-0.0176619 0.292044 -0.57452 -0.334658   -0.40728 0.183023 -0.869271 -0.00414274
--------------------------------------------------------------------------------
 0.0753371 0.0970103 0.100841 0.253254   -0.121684 -0.0520827 -0.0793354 -0.156445
 0.0769915 -0.100714 -0.0550542 -0.143481 0.0906103 0.20446 0.0450487 -0.156413
 0.804288 -0.18739 0.0762514 -0.0981745   0.0627184 -0.0105436 0.0555074 -0.130914
 0.877441 -0.200726 -0.201867 -0.308425   0.0296315 -0.23739 0.0856514 -0.0649629
 0.712155 -0.0195168 0.0899889 0.165317   0.154508 0.311739 -0.104048 -0.283428
-0.0127152 0.187056 0.11761 0.194581      0.0130432 0.290985 -0.121896 -0.353706
-0.148171 -0.0271748 -0.016034 -0.186677  0.16929 -0.0306781 0.0340783 -0.0202074
-1.57432 0.29136 0.527125 0.41191         0.0381834 0.244978 0.159092 0.0732612
```

-0.402442	-0.339509	0.0884796	-0.398597	-0.431954	0.296948	0.79694
0.317519	0.454457	-0.123496	0.53532	0.65565	0.06173	-0.0811966
0.116018	0.158359	-0.109153	0.465186	0.306475	0.125104	0.439372
0.166311	0.27696	-0.137615	0.647967	0.0643326	-0.148442	-0.440816
0.0770605	0.185007	0.0442032	0.15663	0.824449	0.404757	0.469439
0.00139097	0.132863	0.105158	-0.0636119	0.990259	0.533075	0.394152
0.379375	0.364096	-0.0175316	0.428826	0.432948	-0.211912	-0.776385
-0.191997	-0.417126	0.287856	-0.861623	-0.44302	0.0338256	0.907776

PCA matrix; CR stage. (Equation (16), $m = 5$, and Fig. 23a.)

```
-0.448948 0.139323 0.240016 -0.172528 0.237394
0.445653 0.175543 0.212273 -0.116833 -0.153868
0.13128 0.511786 -0.60397 -0.304541 0.501535
0.245948 -0.475425 0.345467 -0.0921352 0.741082
0.451165 -0.0395878 0.0784338 -0.455918 -0.10307
-0.0230495 -0.562008 -0.604314 0.22882 0.0562235
-0.451868 0.168205 0.173525 0.0263903 0.228193
0.337414 0.340746 0.108387 0.770995 0.231025
```

The CR-F model weights (reduced). Neural network. (See Fig. 26c.)

```
5 22 8

0.389245 0.207724 0.0415545 -0.0657605 -0.0911199 0.177098
0.556514 0.246312 -0.0270832 0.14962 -0.0787796 0.374432
-0.11515 0.398386 0.327903 0.0850237 0.049606 0.434496
0.169985 0.137178 -0.0648788 -0.0749181 -0.0538662 0.455194
0.571153 -0.0275301 0.145062 -0.0638335 0.0575497 0.0258657
0.323891 0.0550081 0.328979 0.00858497 0.0238722 0.372199
0.432874 0.186945 0.378456 -0.0393837 0.0725278 0.0637182
0.745408 -0.0909882 -0.0141133 0.153646 0.205203 -0.0365793
-0.0433338 0.395188 0.393226 0.221382 0.258252 0.453381
0.257435 0.261831 0.237446 0.105636 0.223887 0.208202
-0.0181901 0.464977 0.350104 -0.0167638 -0.0296569 0.39578
0.3443 0.0267507 0.200247 -0.0928293 0.0582661 0.230738
0.351432 0.0828723 0.246301 0.286089 -0.0122994 0.026115
0.606732 0.0907523 0.128296 -0.087276 -0.06255 0.374406
0.869093 0.093621 0.0341711 -0.149775 -0.0712468 0.190945
-0.301866 0.441225 0.199308 0.227278 0.300515 0.460712
0.652242 0.261664 -0.0262619 0.0549472 -0.00879476 0.371632
0.713752 0.144663 0.0217355 0.122371 0.212572 -0.0244389
0.0720438 0.0963423 0.350154 0.221248 0.0564427 0.301451
0.915977 0.198816 -0.0248261 0.0970856 0.0118174 0.213779
-0.198608 0.447342 0.319463 0.351251 0.333672 0.420636
0.850964 -0.0815643 0.180848 -0.098453 -0.212108 0.285967

-0.0242737 -0.176137 0.32313 0.0715549     -0.265815 -0.0757016 -0.0490584 -0.283879
0.140564 0.274829 -0.124517 0.0979774       0.272972 0.116347 0.0751988 0.387362
0.250235 0.118432 0.197303 0.126138         0.070921 0.196187 0.0861898 0.109307
0.153559 0.181229 0.00845995 0.0807852      0.0967942 0.0696777 0.159259 0.108985
0.241963 0.215835 0.194328 0.291392         0.0459251 0.213035 0.141513 -0.0199025
0.0660216 0.203595 -0.00865076 0.250489     0.185261 0.0856701 -0.0182208 0.0500239
0.158535 0.000275177 -0.202597 0.0374932    0.176196 -0.0509921 0.0314442 0.396615
-0.191713 -0.133301 0.0604352 0.102431     -0.159806 -0.103244 -0.278701 -0.188997
--------------------------------------------------------------------------------
0.187427   0.005033 0.274622 -0.0127972    -0.0871016 -0.123943 -0.372055 0.426036
-0.127479 0.00439869 -0.115353 0.10462      0.163146 0.276823 0.452728 -0.309838
0.151064 0.0266176 0.101458 0.103773        0.0102432 0.16231 0.172464 0.0294783
-0.0584451 0.104179 0.0996144 0.194118      0.157008 0.143824 0.254601 -0.0548777
0.149129 0.0826895 0.0986011 0.237301      -0.0511785 0.315516 0.285489 0.0170466
-0.0658423 0.0500124 0.0597081 0.0531885    0.0351134 0.117671 0.207657 -0.00179809
-0.236127 -0.0604804 -0.217869 0.0841636    0.0554528 0.196704 0.22584 -0.258439
0.0605342 -0.0409829 -0.0981006 -0.0991666 -0.105949 -0.174397 -0.357512 0.0897234
--------------------------------------------------------------------------------
```

-0.247845	-0.281897	0.0838995	-0.420879	0.25089	-0.265533	0.581445
0.291785	0.267586	-0.03842	0.315605	-0.206467	0.427424	-0.0227673
0.195181	0.0167142	0.0702237	0.152829	0.098963	0.194167	0.552898
0.0750051	0.163271	-0.00346035	0.205992	-0.0170638	0.324778	-0.280575
0.214821	0.0812164	0.0300335	0.156272	0.0212183	0.285357	0.491474
0.119315	0.0993763	0.034942	0.189803	-0.180566	0.215189	0.341663
0.125933	0.315823	-0.0698111	0.305848	-0.345833	0.341529	-0.616239
-0.208361	-0.22047	0.0292588	-0.353433	0.142276	-0.304182	0.477759

PCA matrix; G stage. (Equation (16), $m = 5$, and Fig. 23b.)

```
-0.424773 0.175895 0.333093 -0.184241 0.356866
0.454216 -0.0411636 -0.240762 0.248895 -0.00692783
0.352203 0.108908 0.509722 -0.411096 -0.295344
0.276559 -0.517806 0.418611 -0.0259437 0.628244
0.341344 0.492898 0.163561 -0.323765 -0.0648916
0.321134 0.519782 -0.279289 0.0891611 0.582483
-0.0713927 0.324747 0.535914 0.756366 -0.0877131
-0.433572 0.262296 -0.0638189 -0.222644 0.198718
```

The G-F model weights (reduced). Neural network. (See Fig. 27c.)

```
5 22 8

0.352254 -0.0215603 -0.306115 0.218443 0.32653 0.0293625
0.184781 0.619208 -0.0831976 0.335191 0.34416 0.122681
0.0214977 -0.200823 0.398183 0.29667 0.0791597 0.130623
-0.081146 0.557309 0.0948687 0.0393736 0.329603 0.251947
0.036671 -0.0351811 0.485662 -0.024069 0.137647 0.111635
0.00396835 0.770082 0.152859 -0.293546 0.16043 0.424153
0.0103593 0.371252 0.517976 -0.112411 0.262395 0.134812
0.817603 0.0618705 -0.196402 -0.100552 0.0700276 0.294046
0.317632 -0.344212 0.451668 0.303968 0.159794 0.0794899
0.817632 -0.383377 0.10123 0.19705 0.319004 0.27046
0.562979 0.407069 -0.0238524 0.165003 0.11271 0.0998056
0.959109 0.280213 -0.473058 0.180773 -0.125764 -0.0376375
0.831392 0.0410714 -0.157449 0.0406729 -0.0094954 -0.0420811
0.653407 0.436003 0.00963968 0.167838 0.0876838 0.108159
0.306334 -0.164048 0.409516 -0.225334 -0.000655092 0.31307
0.785771 0.395003 -0.274237 0.0990463 0.0407331 -0.0390799
0.328589 0.0556629 0.494084 -0.256926 0.050106 0.218838
-0.191236 0.767578 0.304391 0.0210292 0.228535 0.286838
0.556873 -0.251106 0.212693 0.0478469 0.241821 0.435858
0.71165 0.335114 -0.0966616 0.0372082 0.292284 0.294004
0.610391 0.260725 0.266514 -0.0797969 0.187029 0.261124
0.496942 0.365262 0.00865438 -0.133312 -0.0556482 0.198533

-0.240816 -0.0312634 0.190091 0.164308 0.233149 0.313814 0.3027 -0.356604
0.260365 0.0699196 -0.0291448 -0.136068 -0.101222 -0.134498 -0.15901 0.424107
-0.0280397 0.026553 -0.0812633 0.0942142 0.00182564 0.394992 0.0450524 0.199105
0.241435 0.192335 -0.19861 0.189182 -0.0787769 0.398888 0.0634199 0.321024
-0.152842 -0.265184 0.0635499 -0.12698 0.244963 0.134784 0.211796 0.31134
0.0191641 -0.280692 0.252961 -0.253869 0.199426 -0.169754 0.00528301 0.245842
0.290719 -0.0759005 0.0149524 -0.23491 -0.0750058 -0.287336 -0.270718 0.326087
0.0384547 -0.417371 0.23761 -0.175097 0.0961122 -0.41915 -0.0919902 -0.259558
-------------------------------------------------------------------------------
-0.0821715 -0.399394 -0.217278 -0.509847 -0.379812 -0.231679 0.0795737 -0.452052
0.0863663 0.486521 0.176248 0.603208 0.372494 0.175164 0.117332 0.28227
-0.167149 -0.0715639 0.19001 0.166504 0.161366 0.113333 0.240016 0.14816
-0.328198 -0.151447 0.172146 0.479177 0.20885 0.259965 -0.0996201 0.335464
0.154391 0.190984 -0.0160404 0.130835 0.177699 0.065028 0.375638 0.0934839
0.392618 0.507473 -0.0105717 -0.0368717 0.16066 0.0013757 0.418643 -0.092062
0.0784083 0.315307 0.133701 0.493565 0.410948 0.156802 -0.0455716 0.355164
0.2413 -0.00638985 -0.3693 -0.496274 -0.26988 -0.393569 0.132571 -0.501397
-----------------------------------------------------------------------
```

0.0586728 0.366481 -0.103733 -0.297078 -0.0331113 -0.143032 0.776189
-0.0135813 -0.318338 0.306381 0.239489 0.178714 0.186107 -0.0122934
0.12445 0.0771469 0.0799623 0.109081 0.173061 0.267281 0.687942
0.0318706 0.265521 -0.138162 0.264527 0.0837886 0.255869 -0.291418
0.467576 -0.00623296 0.20984 0.09599 0.298499 0.293089 0.683646
0.346385 -0.300611 0.39118 0.0507155 0.19832 -0.0173745 0.407712
-0.18292 -0.355125 0.135098 0.15101 0.0638429 0.0546078 -0.72919
-0.036026 -0.285496 0.056554 -0.333959 -0.293954 -0.27836 0.723344

8.4 DESCENT Package

Software Package for Process Modeling and Design Centering *

Aleksander Malinowski, Jacek M. Zurada, Andrzej Lozowski
December 1996

Table of Contents, section numbers, figure numbers, and references of this section 8.4 of the Appendix are all local. This is intentional in order to keep this package description as an independent entity in case it is used as software manual or user's guide.

TABLE OF CONTENTS

1. Introduction
2. Overview of the package
2.1. Controlling the design centering algorithm with *DES_CFG*
2.2. Developing the model of the fabrication process with *DES_PREP*
2.3. Calculating the optimal design center with *DES_CENT*
2.4. Testing the new design center with *DES_TRY*
2.5. Evaluating the new design center with *DES_EVAL*
3. User controlled parameters
3.1. The training strategies
3.2. The termination conditions
3.3. The type of neuron
3.4. The learning methods
3.5. Iterative inverse mapping
3.6. Optimizing the design center
4. Example
4.1. Filter design
4.2. Filter modeling and design centering
5. Bibliography

* Sponsored by the ONR Grant N00014–93–1–0855

1. Introduction

DESCENT is a software package for enhancing yield through design centering. The package uses input/output data measured for manufacturing process and implements a design centering algorithm described in [18, 19]. Fusion of principal component analysis (PCA) and neural network model (NNM) allows for accurate modeling even for nonlinear and/or high dimensional processes with a relatively modest amount of data characterizing the process.

The package source code is written in C++. The source code can be compiled both under MS-DOS and Unix. It also includes ready to use programs compiled for DOS. Minimum system requirements for PC are 640KB RAM and 1MB disk space. The use of 486DX or faster CPU is strongly recommended especially for design centering cases of high dimensionality and large amount of data vectors. More amount of RAM and disk space may be necessary to develop large models using very large neural network structures or large data files.

2. Overview of the package

The package consists of the five following programs, each performing specific task involved in design centering:

- DES_CFG
- DES_PREP
- DES_CENT
- DES_TRY
- DES_EVAL

Program DES_CFG creates the customized configuration file process.ini for one of the chosen strategies for model development. This file is used by other parts of the package. The use of DES_CENT is optional because the default configuration file for a default strategy is created by other programs from the package. However, the use of the program is necessary if the user wants to manually customize the configuration before developing the first model.

In order to make this description of the software package clear, all variables affecting the final product as well as parameters of semi-products are called input settings. The output parameters of the manufactured products are called output parameters here. The dependence between input settings and output parameters is called the model of the process. Fig. 1 illustrates this terminology.

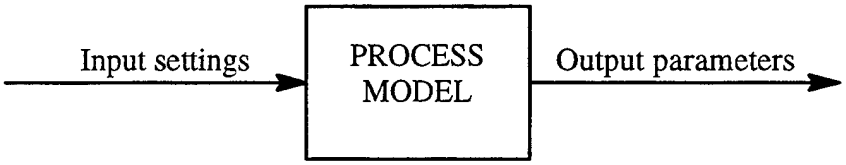


Fig. 1. The model of the process and its input and output variables.

Program DES_PREP develops the neural network model (NNM) of the process for the collected data. The model is later used for design centering and process center evaluation and is stored in two files: process.pca and process.net.

Program DES_CENT calculates input settings which produce the maximum yield of the process given the output parameter specifications and their tolerances.

Program DES_TRY estimates predicted process yield given the evaluated input settings and output parameters, both with their tolerances. It uses already developed NNM.

Program DES_EVAL evaluates the actual yield of the product from the data provided by the user, given the output parameter specifications and their tolerances.

Fig. 2 illustrates the data flow among the programs and the complete process of design centering.

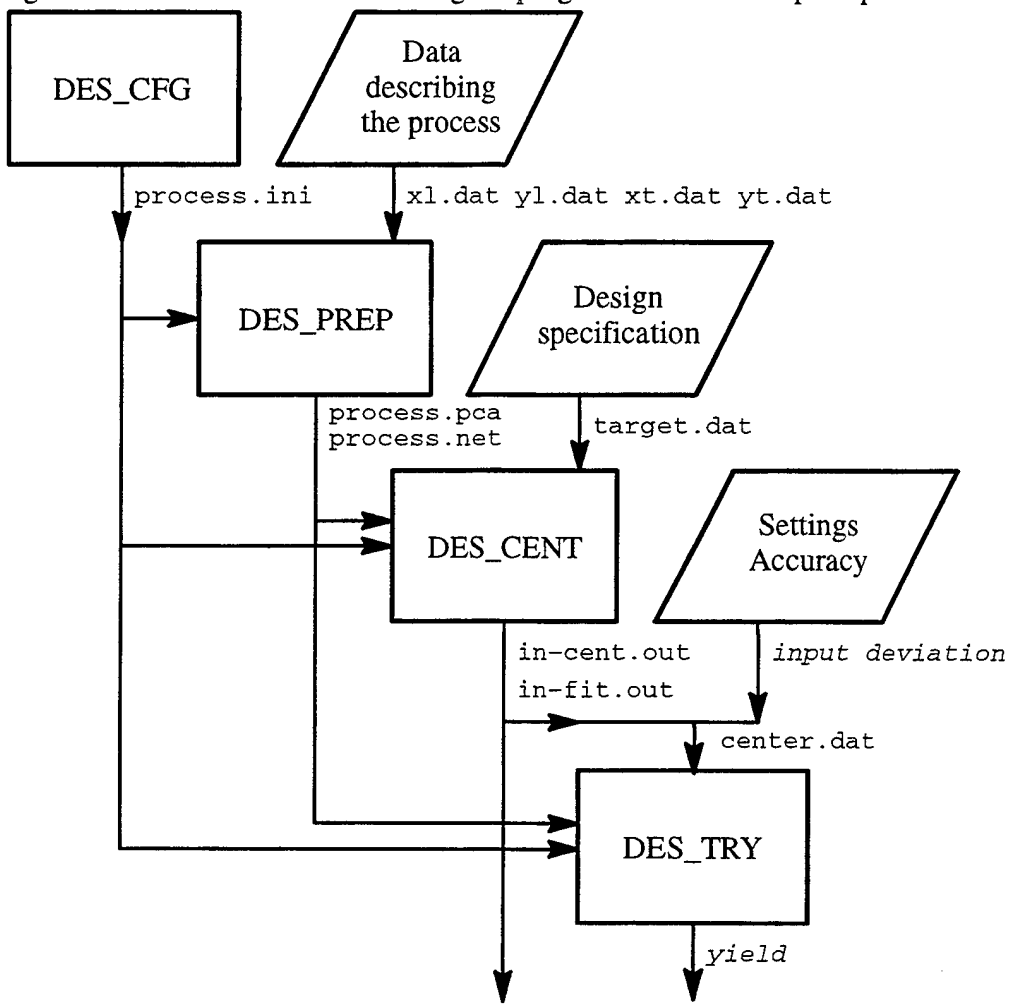


Fig. 2. The data flow in the design centering procedure.

The programs and data files will be explained in the following sections.

2.1. Controlling the design centering algorithm with DES_CFG

Program DES_CFG generates the customized control file process.ini which is described in detail in Section 3, and resets the model of the process in case it exists. The configuration file process.ini is an ASCII file and can be modified using any plain text editor.

When the user runs DES_CFG, he is asked only three questions about the model. Then the complete configuration file is generated with default values of the parameters for the following selected options:

A: Whether to use the PCA for input dimension reduction

B: Select the training strategy

C: Select the learning method

Re A: It is recommended to use the PCA analysis. This allows for dimensionality reduction of the NNM. A NNM with a smaller number of inputs requires less training data for good generalization. If you decide not to use PCA you can still try to reduce internal dimensionality. In that case inputs with the smallest correlation with outputs would be discarded.

Re B: The user can choose among the following training strategies:

A – best of all generalization

B – best generalization

F – best fit to the training data

P – smallest training error followed by network pruning

S – smallest training error.

Recommended selection is A. See Section 3.1 for details.

Re C: The package supports several training methods:

D – delta bar delta (varying localized learning constant and momentum)

L – lambda learning method (JMZ)

Q – quickprop (simplified version)

S – standard error-backpropagation (including pruning methods)

Recommended selection is S for standard learning. See Section 3.4 for details.

2.2. Developing the model of the fabrication process with DES_PREP

Program DES_PREP generates the NNM of the manufacturing process necessary for design centering. This step is the most time consuming operation. To develop the model you will need some data describing it. To run the program type:

```
des_prep xl.dat yl.dat xt.dat yt.dat R H
```

where:

xl.dat	is the file containing input vectors of the learning data set;
yl.dat	is the file containing output vectors of the learning data set;
xt.dat	is the file containing input vectors of the testing data set;
yt.dat	is the file containing input vectors of the testing data set;
R	is the number of inputs of the NNM after PCA input reduction;
H	is the number of hidden neurons in the NNM.

The program reads its configuration from the file process.ini. If this file is not found, one with default settings is created. See Section 3 for detailed description of the configuration file.

Let us denote the original input data dimension as I and output data dimension as K. It is obvious that the PCA-reduced number of inputs in the new coordinates R should be greater than zero and not higher than I. The order of entries in the input vector is changed inside the model, so that defining

R<I results in discarding the least meaningful dimensions. There are two methods used for the task of ranking the inputs: either values of eigenvalues from PCA if PCA is set to be active, or the input-output correlation if PCA is disabled. In the latter case the PCA transformation between inputs and PCA-reduced inputs is repaced by the transformation which only changes the order of entries in the input vector.

The number of hidden neurons H in NNM depends on the complexity of the data relationship, and can not yet be solved analytically [3]. The best way for finding H is to try to use different numbers and choose one which allows the development of the most accurate model. Often, for our size of models H between 5 and 10 is sufficient and therefore could be used on the first try.

When the model of the process is successfully created, two files containing model description are produced: process.pca and process.net. Those files are then used by other programs from the package. They remain unchanged until a new model is created.

The structure of data for the process model development

To generate data for DES_PREP, the measured data needs to be split into two sets: training data set and testing data set. Each set should consist of two files: first with input vectors; and second with output vectors. The following example shows the data files for the two dimensional function approximator for mapping $y = x_1^2 + 0.6x_2^2$ (number of inputs I=2, number of outputs K=1). Although this particular example is not relevant for this modeling effort, it illustrates the structure of the data files. The following Table 1 shows the contents of the data files.

TABLE 1. An example of learning and testing data set.

learning data set		testing data set	
xl.dat - inputs	yl.dat - outputs	xt.dat - inputs	yt.dat - outputs
-10.0 -10.0	160.0	-7.5 -7.5	90.0
-10.0 -5.0	115.0	-7.5 -2.5	60.0
-10.0 0.0	100.0	-7.5 2.5	60.0
-10.0 5.0	115.0	-7.5 7.5	90.0
--10.0 10.0	160.0	-2.5 -7.5	40.0
-5.0 -10.0	85.0	-2.5 -2.5	10.0
-5.0 -5.0	40.0	-2.5 2.5	10.0
-5.0 0.0	25.0	-2.5 7.5	40.0
-5.0 5.0	40.0	2.5 -7.5	40.0
-5.0 10.0	85.0	2.5 -2.5	10.0
0.0 -10.0	60.0	2.5 2.5	10.0
0.0 -5.0	15.0	2.5 7.5	40.0
0.0 0.0	0.0	7.5 -7.5	90.0
0.0 5.0	15.0	7.5 -2.5	60.0
0.0 10.0	60.0	7.5 2.5	60.0
5.0 -10.0	85.0	7.5 7.5	90.0
5.0 -5.0	40.0		
5.0 0.0	25.0		
5.0 5.0	40.0		
5.0 10.0	85.0		
10.0 -10.0	160.0		
10.0 -5.0	115.0		
10.0 0.0	100.0		
10.0 5.0	115.0		
10.0 10.0	160.0		

The dimensionalities of data files are checked by programs during reading of files.

2.3. Calculating the optimal design center with DES_CENT

Program DES_CENT is the central part of the package. It implements the process centering algorithm described in [18, 19]. The model developed using DES_PREP program is used to find the process input settings which optimize the yield of the final product. The calculations are performed for the user-specified desired output parameters and acceptable tolerances. To run the program type:

```
des_cent target.dat
```

where:

target.dat — is the file with desired output parameters. This file contains a vector of target output values and a vector of **relative** (percent) tolerances for each component. See an example in the following subsection.

The program also reads the process model from files process.pca and process.net, and configuration from process.ini.

The results of design centering calculations are written to four new files:

in–cent.out — containing input settings producing the output parameters closest to the specified in target.dat.

out–cent.out — containing output parameters corresponding to input setting from in–cent.out.

in–fit.out — containing input settings optimized for maximum yield of the product.

out–fit.out — containing output parameters corresponding to settings from in–fit.out.

Note that files with these names would be overwritten if they existed before running DES_CENT.

Sometimes it is not possible to find a point which satisfies all of the output parameters specified in the design specification file at the same time. In such case the closest possible input settings are found and reported in in–cent.out instead of exact solution. These settings are then used as an initial point in the yield maximization procedure. The yield is maximized using the specification from target.dat, output parameters and their tolerances. The result settings of this iterative procedure are stored in in–fit.out.

The structure of the design target specification file target.dat

The structure of the design specification file is similar to one of files containing data for modeling. The file consists of two lines. The vector of desired output parameters is stored in the first line. A vector in which the entries are relative tolerances for each parameter from the previous line is stored in the next line. The numbers within each vector are white space delimited.

For example:

```
100.0
  0.05
```

indicates that we desire the first and only one output parameter in our example to be close to 100.0 within the 5% range (95.0..105.0).

2.4. Testing the new design center with DES_TRY

After the optimum input settings are found by DES_CENT, they can be checked against the estimated improvement in yield. Program DES_TRY can statistically evaluate the fabrication yield us-

ing an already developed process model for the given input settings and their standard deviations, and for specified output parameters and their tolerances. To perform a test using DES_TRY type:

```
des_try center.dat target.dat C
```

where:

center.dat	is the file containing input settings and their tolerances. Files in–cent.out or in–fit.out can be used to build center.dat if vector of **absolute** setting tolerances is appended. See an example in the following subsection.
target.dat	is the file with output parameter specifications as described in Section 2.3.
C	is the number of random drawings requested to estimate the yield.

Each tolerance is treated as a variance of the input setting under the assumption of Gaussian distribution. Because of the unknown actual probability distribution of the input data, the actual results from the process may differ to some extent from simulations.

The structure of the design center specification file center.dat

The design center specification file center.dat has the same structure as the design target specification file center.dat with the exception that absolute deviation of each setting is used instead of relative tolerances.

For example:

```
8.00 -7.76
0.2   0.1
```

indicates that we desire the first input setting to be close to 8.0 within the range of 0.2 (7.8..8.2), and the second input setting to be close to –7.76 within the range of 0.1 (–7.86..–7.66).

2.5. Evaluating the new design center with DES_EVAL

Program DES_EVAL evaluates the fabrication yield for actual data. Instead of simulating results of different randomly distributed input settings, the actual results are used. To perform the evaluation using DES_EVAL type:

```
des_eval output.dat target.dat results.dat
```

where:

output.dat	is the file containing the output parameters, similar to yl.dat.
target.dat	is the output target file, same as in Section 2.4.
results.dat	is the file to which the yield results are written.

3. User controlled parameters

All pieces of software use a common profile file process.ini. This file is an ASCII text file and can be modified using plain text editor. Modification of data stored in that file would affect the model development and design centering. The following sample profile file contains default parameters and their brief description:

```
;PROCESS.INI
[MODEL]
pca=1                    ; 0 - disables,  1 - enables
learning-mode=B best     ;  A, B - best gen., F - best fit, S - smallest error
input-scaling-type=1     ; -1 - reserved, 0 - no scaling, 1 - normalization,
```

```
output-scaling-type=9     ;  2 - into -1..+1 range, 9 - into -0.9..+0.9 range
debug=0                   ; debug=1 allows creation of intermediate files *.dbg
[TERMINATION]
;                           Logic AND is performed on all conditions
mse=0.01                  ; Stop training when specified MSE is achieved
max=0.05                  ; Stop when the maximum error is less than specified
misc=0                    ; Used only with NN classifiers, leave unchanged (=0)
iter=10000                ; Maximum number of iterations, 0=no restrictions
[HIDDEN NEURON]
type=0                    ; The hidden neuron type
zero=0.05                 ; Minimum value of the activation fn. derivative
max-net=50.0              ; To prevent overflow in exp()
[OUTPUT NEURON]
type=0                    ; The output neuron type
zero=0.05                 ; Minimum value of the activation fn. derivative
max-net=50.0              ; To prevent overflow in exp()
[LEARNING]
constant=0.1              ; Learning constant 0<η<1
momentum=0.8              ; Momentum constant 0<=μ<1
suppression=0.0           ; Used only in CSDF, otherwise ε=0
facilitation=0.0          ; Used only in CSDF, otherwise γ=0
decay=0.0                 ; Used only in Structural Learning, otherwise γ=0
decaysquare=0.0           ; Used only in quickprop, =1E-5, otherwise =0.
damon=0.0                 ; Left for future development
small-weight=0.001        ; Not used in this package, otherwise =0.001
type=S standard           ; Learning type, e.g. S - EBP, D - DBD
[INVERSION]
constant=0.1              ; Inversion constant 0<ξ<1
end-error=1E-03           ; Inversion accuracy, (=1E-03)
end-derivative=1E-06      ; Local minima detection (=1E-06)
end-iteration=1000        ; Maximum number of iterations (0=no restriction)
[DBD LEARNING]
                          ; Parameters from this section are used only by
                          ; Delta Bar Delta Learning Algorithm
con-vex=0.70              ; Convex constant in Delta Bar Delta training, (θ=0.7)
min-eta=1E-6              ; Minimum learning constant
max-eta=0.50              ; Maximum learning constant to prevent "wild jumps"
inc-eta=0.10              ; Linear increase of learning constant,     (αη=0.1)
dec-eta=0.90              ; Geometric decrease of learning constant, (ψη=0.9)
min-mom=0.00              ; Minimum momentum
max-mom=0.80              ; Maximum momentum to prevent "wild jumps,"(<1)
inc-mom=0.10              ; Linear increase of momentum,              (αμ=0.1)
dec-mom=0.90              ; Geometric decrease of momentum,           (ψμ=0.9)
[OPTIMIZE CENTER]
max-pts-iter=100000       ; Maximum number of iterations during data collection
max-bad-pts=5000          ; Maximum number of "bad" points for centering
min-bad-pts=100           ; Minimum numbers of "bad" points to run centering
min-sigma=0.01            : Initial variance (sigma) during centering
inc-sigma=1.01            ; Geometric increase factor for centering
max-sigma=1               ; Final (maximum) variance during centering
opt-constant=1.0          ; Center optimization constant
opt-end-field=1E-07       ; Optimization termination condition
opt-iter-max=20000        ; Maximum number of iterations during centering
debug=0                   ; debug=1 allows creation of intermediate files *.dbg
```

3.1. The training strategies

The strategy of model development can be controlled by parameters stored in sections [MODEL] and section [TERMINATION]. You can choose among four training strategies:

A best of all generalization

B best generalization (also called "stopped training")
F best fit to the training data
P smallest training error followed by network pruning
S smallest training error.

Recommended selection is B for the best generalization.

The best of all generalization

In the best of all generalization strategy, while the model is developed using the training data set it is also tested at the same time using testing data set. The NNM with the minimum MSE error over the testing data set during the training is stored and then retrieved when the learning is finished. The search for the NNM is continued until the termination condition is reached for the learning data set.

```
[MODEL]
learning-mode=A best of all generalization
```

The best generalization

In the best generalization strategy, while the model is developed using the training data set it is also tested at the same time using testing data set. The NNM training is iterative, but it stops when the minimum MSE error is reached over the testing data set. If the training would have been continued beyond that point, the NNM would start memorizing data instead of trying to generalize them. However, there is danger that the best generalization can not be reached if learning step is too large. To prevent this, do not set the learning constant larger than 0.1 and do not use the delta bar delta learning rule, or use the best of all generalization strategy.

```
[MODEL]
learning-mode=B best generalization
```

The best fit to the training data

When no accurate NNM can be developed due to complex data relationship and data points are collected without noise, the best fit strategy may be a solution. In such case, the learning data set is selected to be very small at the beginning, and then increased by including more data entries which produce largest error as it is described in [12]. This strategy promotes a uniform approximation because data entries with larger error are selected as more important during the training. The search for the NNM is continued until the termination condition is reached for the learning data set.

```
[MODEL]
learning-mode=F best fit
```

Smallest training error followed by pruning

When learning with subsequent pruning is chosen as described in Section 3.4 below, the developed NNM is characterized by very small values of redundant weights. Those weights can be removed without harm or with negligible deterioration of performance, what in turn can enable removal of unconnected hidden neurons [7, 10]. Thus, the size of the NNM can be reduced. The pruning methods in this package are used together with the smallest training error strategy.

```
[MODEL]
learning-mode=P pruning
```

Smallest training error

When there are no separate learning and testing data sets, select the smallest learning error strategy – S. This is the standard approach to NNM training. This strategy works with all learning methods. The search for the NNM is continued until the termination condition is reached for the learning data set.

```
[MODEL]
learning-mode=S standard
```

Input and output scaling

The MFNN inside the NNM may require data scaling during the process of model development. Input data to be processed by PCA are always rescaled so that their mean value is zero, and standard deviation equal to one (`input-scaling-type=1`). Output data, however, can be rescaled in several ways:

- Scaling disabled: `output-scaling-type=-1`
- No scaling performed: `output-scaling-type=0`
- Statistical scaling: `output-scaling-type=1`
- Scaling into –1..+1 range: `output-scaling-type=2`
- Scaling into –.9.+.9 range: `output-scaling-type=9`

The recommended output scaling type is "9" for bipolar continuous output neurons, and "1" for fully linear output neurons. See section 3.3 for details about types of neurons.

```
[MODEL]
input-scaling-type=1
output-scaling-type=9
```

3.2. The termination conditions

All three implemented methods also use additional termination criteria based on the training error and stored in the profile file section [TERMINATION]. The NNM training will be finished if all of the following criteria are satisfied:

- Mean Square Error per pattern per dimension (mse) not exceeded;
- Maximum error over all patterns and all dimensions (max) not exceeded;
- Number of patterns with maximum error larger than 0.1 (misc) not exceeded;
- Number of training cycles (iter) reached;

Logic AND is performed on all conditions. The errors are calculated for the rescaled NNM output. If scaling is performed the error is calculated for the scaled outputs directly. Number of iterations set to 0 means no maximum number of iterations will be enforced.

```
[TERMINATION]
mse=0.01
max=0.05
misc=0
iter=10000
```

3.3. The type of neuron

Data in sections [HIDDEN NEURON] and [OUTPUT NEURON] control the neuron type and its minimum derivative used during training. The neurons in the NNM are hidden (non-output) and output neurons. Four types of neuron activation functions are implemented:

- Continuous bipolar: `type=0` $o = \frac{2}{1 + e^{-net}} - 1$
- Hyperbolic tangent: `type=1` $o = \frac{2}{1 + e^{-2net}} - 1$
- Linear with saturation: `type=2` $o = \begin{cases} -1, & net \leq 1 \\ +1, & net \geq 1 \\ net, & otherwise \end{cases}$
- Fully linear: `type=3` $o = net$

To speed up the process of training, neuron's derivative smaller than that specified by "zero" are replaced by that value. Large activation values may cause numeric overflow in the neuron's activation function. To prevent occurrence of this effect, the maximum activation value is defined by "max–net". Both negative and positive activations are truncated.

```
[NEURON]
type=0
zero=0.05
max-net=50.0
```

3.4. The learning methods

The package supports several training methods:

S standard error-backpropagation (including pruning methods)

Q quickprop (simplified version)

L lambda learning method

D delta bar delta (varying localized learning constant and momentum)

Recommended selection is S – standard learning.

Section [LEARNING] contains the learning constants used by error-backpropagation (EBP) algorithm and some of its modifications. Four main methods of EBP learning are available in the package: standard EBP, EBP with pruning, Lambda learning rule and Delta Bar Delta training. The last of the mentioned methods has its parameters stored in an additional section in the profile file ([DELTA BAR DELTA]).

Standard EBP

Standard error-backpropagation with momentum is the fundamental method for MFNN training. It is described in many sources, for example [1, 16]. We recommend learning constant equal to 0.1 and momentum equal to 0.8.

```
[LEARNING]
constant=0.1
momentum=0.8
type=S standard
```

EBP with pruning

Two pruning methods are supported by the package: Structural Learning (SL) [4, 5, 10], and Convergence Suppression and Divergence Facilitation (CSDF) [13, 14, 15, 10]. To activate these methods, change appropriate constants in the [LEARNING] section of the profile file to non-zero value.

Discussion how to set values of these constants is beyond the scope of this work and can be found in [7, 10]. The example values for CSDF are:

```
[LEARNING]
; Convergence Suppression and Divergence Facilitation defaults
constant=0.1
momentum=0.8
suppression=0.9E-4
facilitation=0.3
decay=0.0
decaysquare=0.0
small-weight=0.001
type=S standard
```

and for SL:

```
[LEARNING]
; Structural Learning defaults
constant=0.1
momentum=0.8
suppression=0.0
facilitation=0.0
decay=1E-4
decaysquare=0.0
small-weight=0.001
type=S standard
```

Lambda learning rule

The lambda learning rule was developed to speed up learning by varying the gain of the neuron. Each neuron has its own gain. The method is described in detail in [17]. It should not be combined with the pruning methods. To activate this method, set the learning type in the profile file to L.

```
[LEARNING]
; Lambda Learning Rule
constant=0.1
momentum=0.8
suppression=0.0
facilitation=0.0
decay=0.0
decaysquare=0.0
small-weight=0.001
type=L Lambda learning
```

Extended Delta Bar Delta learning

In general, the EBP algorithm converges slowly. Delta Bar Delta learning is one of the algorithms where the goal is to speed up the standard EBP training by adapting the values of the learning constant and momentum. It is based on Jacob's algorithm introduced in [6]. The method is discussed in detail in [11].

```
[DBD LEARNING]
con-vex=0.70          ; Convex constant in Delta Bar Delta training, (θ=0.7)
min-eta=1E-6          ; Minimum learning constant
max-eta=0.50          ; Maximum learning constant to prevent "wild jumps"
inc-eta=0.10          ; Linear increase of learning constant,     (αη=0.1)
dec-eta=0.90          ; Geometric decrease of learning constant,  (ψη=0.9)
min-mom=0.00          ; Minimum momentum
max-mom=0.80          ; Maximum momentum to prevent "wild jumps,"(<1)
inc-mom=0.10          ; Linear increase of momentum,              (αμ=0.1)
dec-mom=0.90          ; Geometric decrease of momentum,           (ψμ=0.9)
```

Allowing for large maximum learning constant can cause sudden "jumps" in the weight space and increase the training error. Therefore this training method is not recommended with the best generalization training strategy, and the maximum learning constant should have reasonable value, e.g. 0.5. The minimum learning constant was introduced in this package as an additional feature. It may be useful in the case of data sets that are difficult to train.

3.5. Iterative inverse mapping

The iterative mapping inversion of the model is controlled by parameters stored in section [INVERSION]. The iterative inversion algorithm is used to find NNM inputs which correspond to the desired model output. The search is similar to EBP with the difference that input vector is optimized in place of weights of the NNM which are kept constant. The algorithm is described with details in [2, 8, 9, 10].

```
[INVERSION]
constant=0.1
end-error=1E-06
end-derivative=1E-12
end-iteration=1000
```

3.6. Optimizing the design center

Design centering algorithm used in this package is described in detail in [18, 19]. The initial center of the design is moved in the PCA-reduced space to optimize the process yield. That process is controlled by a few parameters. Usually there is no need to change the default conditions which are listed below:

```
[OPTIMIZE CENTER]
max-pts-iter=100000      ; Maximum number of iterations during data collection
max-bad-pts=5000         ; Maximum number of "bad" points for centering
min-bad-pts=100          ; Minimum numbers of "bad" points to run centering
min-sigma=0.01           : Initial variance (sigma) during centering
inc-sigma=1.01           ; Geometric increase factor for centering
max-sigma=1              ; Final (maximum) variance during centering
opt-constant=1.0         ; Center optimization constant
opt-end-field=1E-07      ; Optimization termination condition
opt-iter-max=20000       ; Maximum number of iterations during centering
debug=0                  ; debug=1 allows creation of intermediate files *.dbg
```

4. Example

In order to illustrate the design centering procedure, a filter fabrication yield will be considered. Assume that a passive high-order band-pass filter is to be designed and then manufactured in a large series of circuits. The filter should have a center frequency equal to ω_0 and a bandwidth of B_{3dB}. Also, attenuation ζ at the center frequency needs to be maintained. The filter is to be built from passive RLC elements. Due to the spread in component parameter values, the filter specifications will differ from those desired but should remain within given tolerances δ_{B3dB}, $\delta_{\omega 0}$ and δ_{ζ}. Required nominal component parameters RLC can be easily calculated from the filter specifications by using appropriate equations. To meet the requirements, some of the components can be tuned using measurements taken at the circuit nodes. Since not every component of the filter is tunable, deviation of the RLC component parameters will result in some of the circuits in the series not meeting the

tolerance requirements and thus being rejected. The goal of the proposed design centering technique is to provide tuning criteria that will minimize the number of rejected circuits and hence maximize the fabrication yield.

4.1. Filter design

Consider a fourth-order band pass filter shown in Fig. 3. The filter magnitude transfer function,

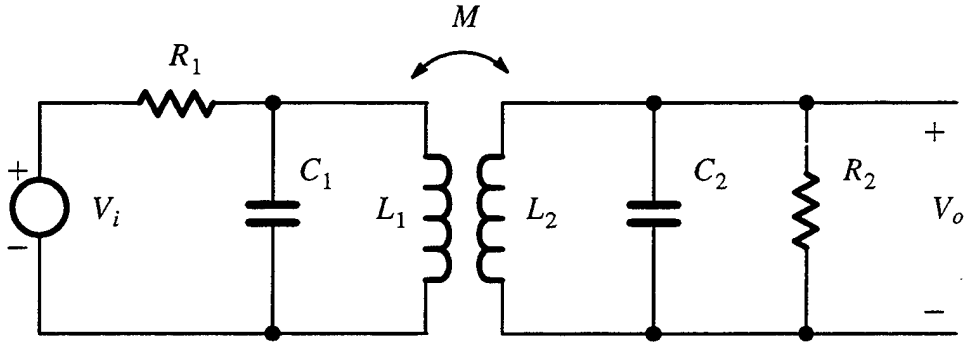


Fig. 3. The fourth-order band pass filter used in the design centering example.

shown in Fig. 4, depends on seven RLC parameters. Center angular frequency

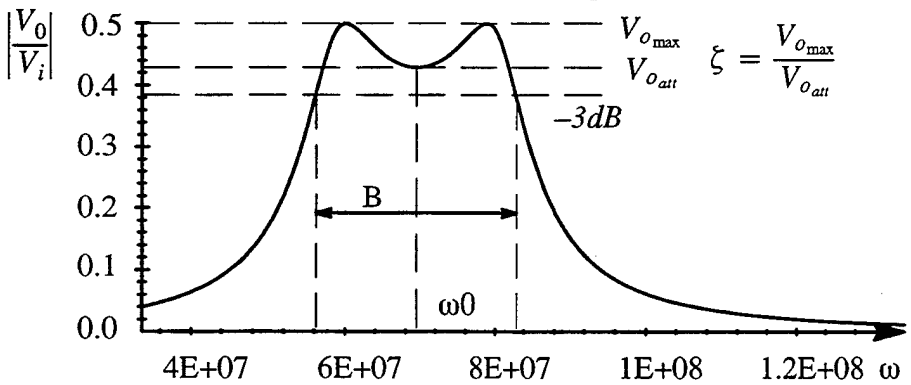


Fig. 3. The transfer function of the fourth-order band pass filter from Fig. 2.

ω_0=2π10.7MHz=67.23E+06, bandwidth B_{3dB}=2π2.3MHz=14.45E+06 and center attenuation coefficient ζ=1 are the specifications that the transfer function should fulfill. Tolerances $\delta_{\omega 0}$, δ_{B3dB} and δ_ζ are criteria for an acceptance or rejection of a given circuit. RLC parameters are calculated from the filter specifications and used as nominal values for the circuit components. Mutually dependent parameters M, L_1 and L_2 are used for tuning. Numeric values of the components are as follows: R_1=R_2=100Ω, L_1=L_2=22.1mH, M=2.21mH, C_1=C_2=1nF.

After a filter circuit is assembled, it needs to be tuned using the following tune-up procedure: set V_i to 1V amplitude and inspect voltage V_0 at five frequencies ω_1 ...ω_5. Tune the inductances until the voltages $V_o(\omega_1)$..$V_o(\omega_5)$ are as close to the optimized input values of the design center as possible. Section 4.2 demonstrates how to calculate the optimized design center using DESCENT software package. If the components used in the circuits were ideal, each assembled filter would precisely match the requirements, and hence implement the transfer function from Fig. 4. However, the

components parameters are distributed around their nominal values and after the tune-up is finished, the resulting ω_0, B_{3dB} and ζ are going to to be off the specs. If ω_0, B_{3dB} and ζ fail to fall into the tolerance range represented by δ, the circuit is rejected.

4.2. Filter modeling and design centering

A set of 2000 circuits has been numerically simulated in order to develop the model of the process. The element values were distorted within 20% on nominal values using uniform distribution. Data were divided into two subsets of equal size: training data set and testing data set, and stored in the files mentioned in the first six rows in Table 2.

TABLE 2. The list of input files for filter modeling and design centering.

file name	description	in Section 3
pa–lrn.dat	values of elements (R_1, R_2, L_1, L_2, M, C_1, C_2) for filters used for model training, not used in the process of design centering	–
pa–tst.dat	values of elements (R_1, R_2, L_1, L_2, M, C_1, C_2) for filters used for model training, not used in the process of design centering	–
vo–lrn.dat	measured voltages V_o's for filters used for training	xl.dat
vo–tst.dat	measured voltages V_o's for filters used for testing	xt.dat
fr–lrn.dat	measured filter output parameters ω_0, B_{3dB} and ζ	yl.dat
fr–tst.dat	measured filter output parameters ω_0, B_{3dB} and ζ	yt.dat
fr–targ.dat	desired output parameters (specification and tolerances)	target.dat
vo–tol.dat	standard deviations of input settings	–

TABLE 3. The list of files created during design centering.

in–cent.out	calculated input settings producing the output parameters closest to the specified in fr–targ.dat	in–cent.out
out–cent.out	output parameters corresponding to the input setting from in–cent.out	out–cent.out
in–fit.out	input settings optimized for maximum yield of the product	in–fit.out
out–fit.out	output parameters corresponding to the settings from in–fit.out	out–fit.out
in–cent.dat	input settings and their tolerances created from file in–cent.out by adding a vector of standard deviations vo–tol.dat	center.dat
in–fit.dat	input settings and their tolerances created from file in–fit.out by adding a vector of standard deviations vo–tol.dat	center.dat

In addition to the files listed in Tables 2 and 3 process.ini, process.pca and process.net are used.

To prepare the configuration file, DES_CFG was executed by typing:

```
des_cfg
```

The program responded:

```
des_cfg: Setup for Design Centering package

Create new Environment for des-cent? (y/n) y

Use PCA input dimension reduction (y/n) y

Choose the training strategy:
A - training for the best of all generalization
B - training for the best generalization
F - training for the best fitting of learning data
P - training for the smallest error followed by pruning
S - training for the smallest learning error

Select now: (b/f/s) b

Which learning method would you like to use:
D - delta bar delta (requires more memory and is
    not recommended with best generalization strategy)
P - error backpropagation with structural learning
L - lambda learning rule
Q - quickprop                          (simplified version)
S - standard error backpropagation with momentum (default)

Select now: (d l p q s) s
Configuration file "process.ini" has been created.
To customize settings further, please edit that file.
```

Then DES_PREP was executed by typing:

```
des_prep vo-lrn.dat fr-lrn.dat vo-tst.dat fr-tst.dat 5 5
```

The program evaluated the PCA components for vo–lrn.dat and produced the following printout:

```
The model dimensionality is: 5-5-5-3.
Training set has 1000 data entries.
Testing  set has 1000 data entries.
Constructing model of the process ...
Reading profile ...
Sorted Principal Components:
L[1] =        2.6124
L[2] =        2.16616
L[3] =        0.171451
L[4] =        0.038635
L[5] =        0.0113584

Preparing data ...
Normalizing ...
Training ...
Training for the smallest error ...

^C
```

At this moment the program was terminated by user (^C) and rerun with a different number of components used for internal NNM. Based on the calculated PCA eigenvalues (L[.]), only two reduced dimensions were selected with corresponding eigenvalues L[1] and L[2]. DES_PREP was executed once again with smaller PCA-reduced dimension specified by typing:

```
des_prep vo-lrn.dat fr-lrn.dat vo-tst.dat fr-tst.dat 2 5
```

The program evaluated the PCA components for `vo-lrn.dat`, and then NNM training was started by typing:

```
des_prep vo-lrn.dat fr-lrn.dat vo-tst.dat fr-tst.dat 2 5
```

The program responded:

```
The model dimensionality is: 5-2-5-3.
Training set has 1000 data entries.
Testing  set has 1000 data entries.
Constructing model of the process ...
Reading profile ...
Sorted Principal Components:
IMP[1] =        2.6124
IMP[2] =        2.16616
=== cut is set here ===
IMP[3] =        0.171451
IMP[4] =        0.038635
IMP[5] =        0.0113584

Preparing data ...
Normalizing ...
Training ...
Training for the best generalization ...
IT =      10 L-ERR = 0.108968 T-ERR = 0.112958
...
```

After training, two new files are created: process.pca and process.net. They contain the complete description of the model.

A new file fr–targ.dat (target.dat) with target output parameter values (ω_0, B_{3dB} and ζ) with the following contents corresponding to the specifications (δ_{B3dB}, $\delta_{\omega 0}$ and δ_ζ) from Section 3.1 needs to be created by user:

```
67.23E+06 14.45E+06 1.0
 0.05 0.1 0.2
```

Then the voltages $V_o(\omega_1)..V_o(\omega_5)$ were calculated according to the design centering algorithm using DES_CENT program. DES_CENT was executed by typing:

```
des_cent fr-targ.dat
```

This produced:

```
Retrieving model of the process from process.pca and process.net ...
The model dimensionality is: 5-2-5-3, PCA active
Reading specifications from fr-targ.dat ...

Design specification ...
 67.23e+06 14.45e+06 1
 0.05 0.1 0.2

Inverse mapping ...
NOTE: The best local minimum found
IN:   0.224642 0.288539 0.314294 0.28986 0.230704
OUT:  6.7197e+07 1.39769e+07 0.98046

Design centering ...
Looking for initial center ...
NOTE: The best local minimum found
Looking around the initial design center ...
NOTE: 10000 bad data collected
Optimizing the design center ...
```

```
NOTE: Optimization force field is down to 9.99178e-08 in 4505 iterations
IN:   0.230324 0.286644 0.304525 0.278411 0.221964
OUT:  6.75473e+07 1.45612e+07 0.974033

Done!
```

The program calculated two centers: one corresponding to the specified desired (target) filter parameters (in–cent.out), and the second one (in–fit.out) corresponding to the maximum yield of manufactured filters for the given specification – parameters and their tolerances (in–fit.out). The numbers in these files correspond to voltages $V_o(\omega_1)..V_o(\omega_5)$. Estimated output parameters for those specifications were also stored in out–cent.out and out–fit.out. The numbers in these files correspond to ω_0, B_{3dB} and ζ. Note that your results for this example may differ slightly due to the selection of different random points used for center optimization.

in–cent.out:

```
 0.224642 0.288539 0.314294 0.28986 0.230704
```

out–cent.out:

```
 6.7197e+07 1.39769e+07 0.98046
```

in–fit.out:

```
 0.230324 0.286644 0.304525 0.278411 0.221964
```

out–fit.out:

```
 6.75473e+07 1.45612e+07 0.974033
```

The performed optimization results of the design center can be tested and improvement evaluated using DES_TRY. For that purpose both files have to be supplied with the process parameter deviations. In the case of our example the voltages can be tuned only with the accuracy of 0.05V due to the small number of tunable elements.:

```
 0.05  0.05  0.05  0.05  0.05
```

Adding these deviations to in–cent.out and in–fit.out yields two new files:

in–cent.dat:

```
 0.224642 0.288539 0.314294 0.28986 0.230704
 0.05  0.05  0.05  0.05  0.05
```

in–fit.dat:

```
 0.230324 0.286644 0.304525 0.278411 0.221964
 0.05  0.05  0.05  0.05  0.05
```

The process yield can be estimated using developed model both for both centers by typing:

```
des_try  in-cent.dat  fr-targ.dat  10000
```

and

```
des_try  in-fit.dat  fr-targ.dat  10000
```

For the given design centers and the model the results were following:

For the first center the program responded with:

```
Retrieving model of the process ...
The model dimensionality is: 5-2-5-3, PCA active
Reading design center and absolute tolerance from in-cent.dat ...
Reading specifications from fr-targ.dat ...
Calculating 10000 sample cases ...
 IN-TOL #1 OUT-TOL #1 PREDICTED YIELD IS 5319 out of 10000
Done!
```

For the latter center the program responded with:

```
Retrieving model of the process ...
The model dimensionality is: 5-2-5-3, PCA active
Reading design center and absolute tolerance from in-fit.dat ...
Reading specifications from fr-targ.dat ...
Calculating 10000 sample cases ...
 IN-TOL #1 OUT-TOL #1 PREDICTED YIELD IS  5880 out of 10000
Done!
```

Now the decision can be made about the voltage values for filter tuning. Moving the process center from the initial value (in–cent.out) to the optimized design center (in–fit.out) would increase the filter manufacturing yield by about 5.5% with no additional changes in the manufacturing process.

5. BIBLIOGRAPHY

1. J. Hertz, A. Krogh, R. Palmer, *Introduction to the Theory of Neural Computation.* Addison–Wesley 1991.

2. D. A. Hoskins, J. N. Hwang, J. Vagners, "Iterative Inversion of Neural Networks and Its Application to Adaptive Control," *IEEE Transactions on Neural Networks,* vol. 3, no. 2, March 1992, pp.292–301.

3. S–C. Huang, Y–F. Huang, "Bounds on the Number of Hidden Neurons in Multilayer Perceptrons," *IEEE Transactions on Neural Networks,* vol. 2, No 1, January 1991, pp. 47–55.

4. M. Ishikawa, "A Structural Learning Algorithm with Forgetting of Link Weights," *Proceedings of International Joint Conference on Neural Networks*, Washington. DC, 1989, vol. 2, pp. 626.

5. M. Ishikawa,"Structural Learning in Neural Networks," *Proceedings of the 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing,* Iizuka, Japan, August 1–7, 1994, pp. 37–44.

6. R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, No. 1, 1988, pp.295–307.

7. A. Malinowski, D. A. Miller and J. M. Zurada, "Reconciling Training and Weight Suppression: New Guidelines for Pruning–Efficient Training," *Proc. of the World Congress on Neural Networks,* Washington, DC, July 17-21, 1995, vol. 1, pp. 724–728.

8. A. Malinowski, J. M. Zurada, J. H. Lilly "Inverse Control of Nonlinear Systems Using Neural Network Observer and Inverse Mapping Approach," *Proc. of IEEE International Conference on Neural Networks,* Perth, Western Australia, November 27–December 1, 1995, vol. 5, pp. 2513–2518.

9. A. Malinowski, T. J. Cholewo, J. M. Zurada, P. B. Aronhime, "Inverse Mapping with Neural Network for Control of Nonlinear Systems," *Proc. of IEEE International Symposium on Circuits and Systems,* Atlanta, GA, May 12–15, 1996, vol. 3, pp. 453–456.

10. A. Malinowski, *Reduced Perceptron Networks and Inverse Dynamics Control,* University of Louisville, Louisville, Kentucky, August 1996, Ph.D. Dissertation.

11. K. Ochiai, S. Usui, "Improved Kick Out Learning Algorithm with Delta-Bar-Delta-Bar Rule," *Proc. of the IEEE International Conference on Neural Networks*, San Francisco, California, March 28–April 1, 1993, vol. 1, pp.269–274.

12. M. Plutowski, H. White, "Selecting Concise Training Sets from Clean Data," *IEEE Transactions on Neural Networks*, vol. 4, No. 2, March 1993, pp. 305–318.

13. S. Yasui, "A new method to remove redundant connections in backpropagation neural networks: Introduction of 'parametric lateral inhibition fields,'" *Proc. IEEE INNS International Joint Conference on Neural Networks*, Beijing, 1992, vol. 2, pp. 360–367.

14. S. Yasui "Convergence suppression and divergence facilitation for pruning multi–output backpropagation networks," *Proc. 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing,* Iizuka, August 1–7, 1994, pp. 137–139.

15. S. Yasui, A. Malinowski, J. M. Zurada, "Convergence Suppression and Divergence Facilitation: New Approach to Prune Hidden Layer and Weights in Feedforward Neural Networks," *Proc. of IEEE International Symposium on Circuits and Systems,* Seattle, WA, April 29–May 3, 1995, vol. 1, pp. 121–124.

16. J. M. Zurada, *Introduction to Artificial Neural Systems*, PWS Publishing Company, March 1992.

17. J. M. Zurada, "Lambda Learning Rule for Feedforward Neural Networks," *Proc. of the International IEEE Conference on Neural Networks*, San Francisco, California, March 28–April 1, 1993, pp. 1808–1811.

18. J. M. Zurada, A. Lozowski, A. Malinowski, "Design Centering in GaAs IC Manufacturing" Accepted to *Proc. of the IEEE Aerospace Conference*, Snowmass, Colorado, USA, February 1–8, 1997.

19. J. M. Zurada, A. Lozowski, A. Malinowski, "Yield Improvement in GaAs IC Manufacturing using Neural Network Inverse Modeling," submitted to *Proc. of the International Joint Conference on Neural Networks (IJCNN'97)*, Houston, Texas, June 9–12, 1997.

8.5 List of Published Papers

GRANT REPORT
N00014-93-1-0855
July 1, 1993 - December 31, 1996
List of Publications

"Neural Network Models for Yield Enhancement in Semiconductor Manufacturing" and "Neural Networks for Inverse Parameter Modeling of IC Fabrication Stages"

Date: February 1997

Place: Department of Electrical Engineering

Dr. Jacek M. Zurada, PI

List of papers published supported by the Grant:

Zurada, J.M., Malinowski, A., Usui, S., **Perturbation Method for Deleting Redundant Inputs of Perceptron Networks**, Neurocomputing 14 (1997), pp. 177-193

Deif, H.M., Zurada, J.M., Kuczborski, W., **Inverse Mapping of Continuous Functions Using Feedforward Neural Networks**, accepted for the 1997 IEEE International Conference on Neural Networks, Houston, Texas, June 9-12, 1997

Zurada, J.M., Lozowski, A., Malinowski, A., **Yield Improvement in GaAs IC Manufacturing Using Neural Network Inverse Modeling**, accepted for the 1997 International IEEE Conference on Neural Networks, Houston, Texas, June 9-12, 1997

Zurada, J.M., Lozowski, A., Malinowski, A., **Design Centering in GaAs IC Manufacturing**, Proc. of the 1997 IEEE Aerospace Conference, Snowmass at Aspen, Colorado, February 1-8, 1997, Vol. II, pp. 435-447

Creech, G.L., Zurada, J.M., **Neural Network Modeling of GaAs IC Material and MESFET Device Characteristic**, Proc. of the OAI Neural Networks Symposium and Workshop, Athens, Ohio, August 21-22, 1995, pp. 257-266

Creech, G.L., Zurada, J.M., **Inverse Modeling of MMIC Fabrication Material and Device Characteristics Using Feed-Forward Neural Networks**, Proc. of the GaAs Manufacturing Technology Conference, New Orleans, Louisiana, May 9-12, 1995, pp. 20-23

Creech, G.L., Zurada, J.M., Aronhime, P.B., **Feedforward Neural Networks for Estimating IC Parametric Yield and Device Characterization**, Proc. of the 1995 IEEE International Symposium on Circuits and Systems, Seattle, Washington, April 29 - May 3, 1995, vol. 2, pp. 1520-1523

Creech, G.L., Zurada, J.M., **GaAs Mesfet DC Characteristics and Process Modeling Using Neural Networks**, Proceedings of ANNIE94, St. Louis, Missouri, November 13-16, 1994, pp. 1005-1013

Zurada, J.M., Malinowski, A., Cloete, I., **Sensitivity Analysis for Minimization of Input Data Dimension for Feedforward Neural Networks**, Proc. of the 1994 International IEEE Symposium on Circuits and Systems, London, England, May 28-June 1, 1994, vol. 6, pp. 447-450

Zurada, J.M., Malinowski, A., **Minimal Training Set Size Estimation for Neural Network-Based Function Approximation**, Proc. of the 1994 International IEEE Symposium on Circuits and Systems, London, England, May 28 - June 1, 1994, vol. 6, pp. 403-406

Zurada, J.M., **Multilayer Perceptron Networks: Selected Aspects of Training Optimization**, Invited Paper, Proc. of the First National Polish Conference on Neural Networks and Applications, Kule, April 12 - 15, 1994, vol. 1, pp. 82-103 (later reprinted in Appl. Math. and Comp. Sci., 1994, vol. 4, No. 3, pp. 281-307)

NEUROCOMPUTING

Volume 14, Number 2 — An International Journal — ISSN 0925-231

5 February 1997 — 121–20

Editor-in-Chief
V. David Sanchez A.

now also including:

NEUROCOMPUTING LETTERS

Now available:

Electronic Features

http://www.elsevier.nl/locate/neucom

ELSEVIER

Neurocomputing 14 (1997) 177–193

NEUROCOMPUTING

Perturbation method for deleting redundant inputs of perceptron networks [1]

Jacek M. Zurada [a,*], Aleksander Malinowski [a], Shiro Usui [b]

[a] *Department of Electrical Engineering, University of Louisville, Louisville, KY 40292, USA*
[b] *Toyohashi University of Technology, Toyohashi, Japan*

Received 9 May 1995; accepted 31 January 1996

Abstract

Multilayer feedforward networks are often used for modeling complex functional relationships between data sets. Should a measurable redundancy in training data exist, deleting unimportant data components in the training sets could lead to smallest networks due to reduced-size data vectors. This reduction can be achieved by analyzing the total disturbance of network outputs due to perturbed inputs. The search for redundant input data components proposed in the paper is based on the concept of sensitivity in linearized models. The mappings considered are $\mathbb{R}^I \to \mathbb{R}^K$ with continuous and differentiable outputs. Criteria and algorithm for inputs' pruning are formulated and illustrated with examples.

Keywords: Perceptron networks; Sensitivity to inputs; Input layer pruning; Feature elimination; Saliency measures; Continuous mapping

1. Introduction

Neural networks are often used to model complex functional relationships between sets of experimental data. Such a modeling approach proves useful when analytical models of processes do not exist or are not known, but when sufficient data is available for embedding existing relationships into neural network structures. Multilayer feedforward neural networks (MFNN) consisting of continuous neurons have been found particularly useful for such model building [1–3]. Representative training data are used in such cases for supervised training of a suitable user-selected MFNN architecture.

* Corresponding author. Email: jmzura02@starbase.spd.louisville.edu.

[1] This work was supported in part by the ONR Grant N00014-93-1-0855.

0925-2312/97/$17.00 Copyright © 1997 Elsevier Science B.V. All rights reserved.
PII S0925-2312(96)00031-8

Minimization of potential redundancy in data used for supervised training can take different forms [4]. Duplicative data pairs are essentially removable from the training sets without a loss of accuracy. In contrast, special attention should be paid to data that carry conflicting information. Such data do not normally allow for unique mapping and should be eliminated. Our concern in this paper is to explore potential redundancy in input vector dimensionality. As such, this concern has only little in common with the widely used notion of network pruning. By deleting superfluous inputs, if such inputs exist, the number of input nodes is reduced. The resulting network is still pruned as it contains no weights fanning out of deleted inputs.

A popular objective of network pruning is to detect irrelevant weights and neurons. This can be achieved through evaluation of sensitivities of the error function to the weights which are the learning parameters [5,6]. Errors other than quadratic are often used to achieve identification of insensitive weights. Statistical moments of neural networks-built mappings, including sensitivities to inputs, are discussed in [7]. Our focus in the paper is mainly to develop clear and practical measures of sensitivities to inputs rather than to weights or neurons. Then, a systematic algorithmic approach has been developed to utilize these measures towards deletion of redundant inputs.

To determine which inputs are necessary for the satisfactory neural network performance a metric known as saliency was introduced in [8]. Belue and Bauer developed an algorithm extending the saliency metric over the entire input space [9]. The approach involves multiple neural network training and superposition of noise on the training patterns to reduce the dependence of results on local minima. This method, however, is computationally intensive due to the required multiple training sessions and exhaustive coverage of the input space.

The saliency method was developed to determine the irrelevant features for neural network classifiers [9]. The sensitivities of MFNN outputs with respect to inputs are calculated and used along with various metrics to evaluate importance of features. Such classifier networks in general are characterized by small sensitivities when fully trained. Therefore, saliency can be applied only with the addition of noise to the training patterns and with sampling of the input space over the whole domain. Multiple training is necessary to average the results and prevent dependence on local minima achieved during training.

This paper focuses on the concept of sensitivity, or perturbation method, for pruning unimportant inputs for neural networks providing continuous mapping. This assumption and the proposed new sensitivity summation metrics allow application of the method directly to trained MFNNs without adding noise to the training patterns or multiple trainings. In fact, in case of continuous mapping the problem of local minima reached during training is not important if sufficient approximation accuracy is achieved. As a result the presence of local minima does not affect the Jacobian matrix used by this method. The Jacobian matrix is derived from the approximate neural network mapping over the training data set. This eliminates the need for computationally intensive repetitive training. In addition to mappings with continuous outputs the sensitivity method can be applied to the classification problems. However, in such cases an additional neural network has to be trained as described in one of the examples.

Let us consider an MFNN with a single hidden layer. The network is assumed to

perform a nonlinear, differentiable mapping $\Gamma:\mathbb{R}^I \to \mathbb{R}^K$, $\mathbf{o}=\Gamma(\mathbf{x})$, where $\mathbf{o}$ $(K\times 1)$, and $\mathbf{x}$ $(I\times 1)$ are output and input vectors, respectively. In further discussion it is assumed that certain inputs bear none, or little statistical or deterministic relationships to output vectors, and are therefore removable. The objective here is to reduce the original dimensionality of the input vector, $\mathbf{x}$, so that a smaller network can be used as a model without loss of accuracy. Initial considerations published in [10–12] are extended below along with a formal framework for the perturbation approach as applied to the neural network models.

Let $\mathbf{o}:\mathbb{R}^I \to \mathbb{R}^K$ with component functions $o_1, o_2, \ldots, o_K$. Suppose $\mathbf{x}^{(n)} \in \Omega$, where Ω is an open set. Since $\mathbf{o}$ is differentiable at $\mathbf{x}^{(n)}$ we have

$$\mathbf{o}(\mathbf{x}+\Delta\mathbf{x}) = \mathbf{o}(\mathbf{x}^{(n)}) + \mathbf{J}(\mathbf{x}^{(n)})\Delta\mathbf{x} + \mathbf{g}(\Delta\mathbf{x}), \tag{1}$$

where

$$\mathbf{J}(\mathbf{x}^{(n)}) = \begin{bmatrix} \dfrac{\partial o_1}{\partial x_1} & \dfrac{\partial o_1}{\partial x_2} & \cdots & \dfrac{\partial o_1}{\partial x_I} \\ \dfrac{\partial o_2}{\partial x_1} & \dfrac{\partial o_2}{\partial x_2} & \cdots & \dfrac{\partial o_2}{\partial x_I} \\ \cdots & \cdots & \cdots & \cdots \\ \dfrac{\partial o_K}{\partial x_1} & \dfrac{\partial o_K}{\partial x_2} & \cdots & \dfrac{\partial o_K}{\partial x_I} \end{bmatrix}_{\mathbf{x}^{(n)}}$$

is the Jacobian matrix and

$$\lim_{\Delta\mathbf{x}\to 0} \frac{\mathbf{g}(\Delta\mathbf{x})}{|\Delta\mathbf{x}|} = 0. \tag{2}$$

Fig. 1 provides geometrical interpretation of relationship (1) in space $\mathbb{R}^K$. Point $\mathbf{o}(\mathbf{x}^{(n)})$ represents the nominal response of the MFNN for the n-th element of the training set, $\mathbf{x}^{(n)}$. The disturbance $\Delta\mathbf{x}$ of the input vector causes the perturbed response at $\mathbf{o}(\mathbf{x}^{(n)}+\Delta\mathbf{x})$. This response can be expressed as a combination of three vectors as indicated in Eq. (1).

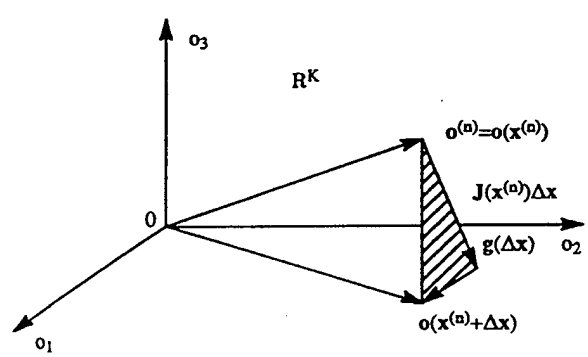


Fig. 1. Geometrical interpretation of output disturbance due to the input disturbance $\Delta\mathbf{x}$.

Assuming now that the network with input $\mathbf{x}$ is perturbed by applying $\Delta\mathbf{x} \to \mathbf{0}$, then the only relevant component in Eq. (1) becomes $\mathbf{J}(\mathbf{x}^{(n)})\Delta\mathbf{x}$ due to the fact that the first term of Eq. (1) is fixed, and the third one vanishes accordingly to Eq. (2). This corresponds to the shaded triangle vanishing due to the vanishing $\mathbf{g}(\Delta\mathbf{x})$ side, but also due to the vanishing multiplier $\Delta\mathbf{x}$ of the Jacobian matrix. Still, matrix $\mathbf{J}(\mathbf{x}^{(n)})$ provides the crucial first-order directional information about the non-zero displacement $\mathbf{o}(\mathbf{x}^{(n)} + \Delta\mathbf{x}) - \mathbf{o}(\mathbf{x}^{(n)})$.

The proposed input perturbation approach has proven rather useful for function approximation cases studied in context of input pruning. However, in case of pattern classifiers output neurons are near saturation and the method needs to be modified as discussed in one of the later sections.

2. Statement of the problem

Our purpose is to evaluate the displacements due to the perturbed inputs over the entire training set $\mathscr{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$. For an example of several training vectors $\mathbf{x}^{(n)} \in \mathscr{X}$ disturbed by vector $\Delta\mathbf{x}$ each output relationship is depicted in Fig. 2(a). Depicted displacements are for identical and small values of $\Delta\mathbf{x}$ for $i = 1,2,\ldots,N$ ($N = 18$ in this example).

These changes can be projected back to the input space $\mathbb{R}^I$. The question is whether or not all I dimensions of input vectors are relevant for having caused the displacements

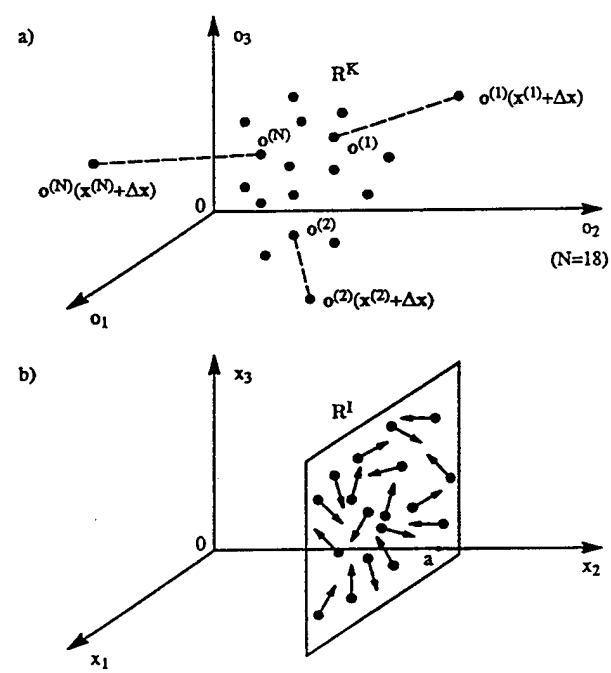


Fig. 2. Perturbation impact in (a) output space, and (b) input space when all output changes are constant in x_2.

of outputs. Fig. 2(b) illustrates an example of respective input changes which are causing output perturbations of Fig. 2(a). It can be seen that the variable x_2 does not participate in output changes $\mathbf{o}^{(k)}(\mathbf{x}^{(i)} + \Delta\mathbf{x})$, or each of the K output functions measured on the training set $\mathscr{X}$ is constant in x_2.

In general, should all outputs be insensitive to the i-th variable, then the entire i-th column of the Jacobian matrix would vanish. Note that the vanishing column property would have to hold for the entire training set $\mathscr{X}$, thus zeroing the i-th column for $\mathbf{J}(\mathbf{x}^{(n)})$, $n = 1,2,\ldots,N$. In real life situations, however, qualitative measures other than zero need to be developed to compare the relative significance of each particular input over the training set. Following sections of the paper are aimed at formulating such measures and the related input variable pruning algorithm.

3. Sensitivity matrix

It can be easily noticed that the entries of the Jacobian matrix defined in Eq. (1) can be considered as sensitivity coefficients. Specially, sensitivity of an output o_k with respect to its input x_i is

$$S_{x_i}^{o_k} \doteq \frac{\partial o_k}{\partial x_i} \tag{3}$$

which can be written succinctly as

$$S_{ki} \doteq S_{x_i}^{o_k}.$$

By using the standard notation of an error backpropagation approach [3], the derivative of Eq. (3) can be readily expressed in terms of network weights as follows:

$$\frac{\partial o_k}{\partial x_i} = o_k' \sum_{j=1}^{J-1} w_{kj} \frac{\partial y_j}{\partial x_i}, \tag{4}$$

where y_j denotes the output of the j-th neuron of the hidden layer, and o_k' is the value of derivative of the activation function $o = f(net)$ taken at the k-th output neuron. This further yields

$$\frac{\partial o_k}{\partial x_i} = o_k' \sum_{j=1}^{J-1} w_{kj} y_j' v_{ji}, \tag{5}$$

where y_j' is the value of derivative of the activation function $y = f(net)$ of the j-th hidden neuron ($y_J' = 0$ since the J-th neuron is a dummy one, i.e. it serves as a bias input to the output layer). The sensitivity matrix $\mathbf{S}$ $(K \times I)$ consisting of entries as in Eq. (5) or Eq. (3) can now be expressed using array notation as

$$\mathbf{S} = \mathbf{O}' \times \mathbf{W} \times \mathbf{Y}' \times \mathbf{V}, \tag{6}$$

where $\mathbf{W}$ $(K \times J)$ and $\mathbf{V}$ $(J \times I)$ are output and hidden layer weight matrices, respectively, and $\mathbf{O}'$ $(K \times K)$ and $\mathbf{Y}'$ $(J \times J)$ are diagonal matrices defined as follows

$$\begin{aligned} \mathbf{O}' &\doteq diag(o_1', o_2', \ldots, o_k'), \\ \mathbf{Y}' &\doteq diag(y_1', y_2', \ldots, y_J'). \end{aligned} \tag{7}$$

Matrix S contains entries S_{ki} which are ratios of absolute increments of output k due to the input i as defined in Eq. (3). This matrix depends only upon the network weights as well as slopes of the activation functions of all neurons. Each training vector $\mathbf{x}^{(n)} \in \mathscr{X}$ produces different sensitivity matrix $\mathbf{S}^{(n)}$ even for a fixed network. This is due to the fact that although weights of a trained network remain constant, the activation values of neurons change across the set of training vectors $\mathbf{x}^{(n)}$, $n = 1,2,\ldots,N$. This, in turn, produces different diagonal matrices of derivatives $\mathbf{O}'$ and $\mathbf{Y}'$, which strongly depend upon the neurons' operating points determined by their activation values. These matrices contain linearized activation functions at their quiescent points.

4. Measures of sensitivity to inputs over training set

In order to evaluate the option of dimensionality reduction of input vectors, the sensitivity matrix as in Eq. (6) needs to be evaluated over the entire training set $\mathscr{X}$. Let us define the sensitivity matrix for the pattern x_n as $\mathbf{S}^{(n)}$. There are several ways to define the overall sensitivity matrix, each relating to the different objective function which needs to be minimized.

The *mean square average* (MSA) *sensitivities*, $S_{ki,\mathrm{avg}}$, over the set $\mathscr{X}$ can be computed as

$$S_{ki,\mathrm{avg}} \doteq \sqrt{\frac{\sum_{n=1}^{N} \left(S_{ki}^{(n)}\right)^2}{N}}\,. \tag{8}$$

Matrix $\mathbf{S}_{\mathrm{avg}}$ $(K \times I)$ is defined as $[\mathbf{S}_{\mathrm{avg}}] = S_{ki,\mathrm{avg}}$. This method of sensitivity averaging is coherent with the goal of network training which minimizes the mean square error over all outputs and all patterns in the training set.

The *absolute value average sensitivities*, $S_{ki,\mathrm{abs}}$, over the set $\mathscr{X}$ can be computed as

$$S_{ki,\mathrm{abs}} \doteq \sqrt{\frac{\sum_{n=1}^{N} |S_{ki}^{(n)}|}{N}}\,. \tag{9}$$

Matrix $\mathbf{S}_{\mathrm{abs}}$ $(K \times I)$ is defined as $[\mathbf{S}_{\mathrm{abs}}] = S_{ki,\mathrm{abs}}$. Note that summing sensitivities across the training set requires taking their absolute values due to the possibility of cancelations of their taking negative and positive values. This method of averaging may be more advantageous than Eq. (8) when sensitivities $S_{ki}^{(n)}$, $n = 1,\ldots,N$, are of disparate values.

The *maximum sensitivities*, $S_{ki,\max}$, over the set $\mathscr{X}$ can be computed as

$$S_{ki,\max} \doteq \max_{n=1\ldots N} \left\{S_{ki}^{(n)}\right\}. \tag{10}$$

Matrix $\mathbf{S}_{\max}$ $(K \times I)$ is defined as $[\mathbf{S}_{\max}] = S_{ki,\max}$. This sensitivity definition allows to prevent pruning inputs which are rather relevant for the network, but relevance occurs only in a small percentage of input vectors of the whole training set. Infrequent but relevant relationships in the training set are masked due to the averaging in Eq. (8) and

Eq. (9), but remain distinguishable for the measure (10). The drawback of this measure is that the significance of inputs can be overestimated and some unimportant inputs may remain after shrinking the input vector.

Any of the sensitivity measure matrices proposed in Eqs. (8)–(10) can provide useful information as to the relative significance of each of the inputs in $\mathscr{X}$ to each of the outputs. For the sake of brevity, however, mainly the MSA sensitivity matrix defined in Eq. (8) will be used in further discussion but other sensitivity measures will be used for comparison purposes. The cumulative statistical information resulting from Eq. (8) will be used along with criteria for reducing the number of inputs to the smallest number of them sufficient for accurate learning. These criteria are formulated in the next section.

Other useful measure of sensitivity used for the evaluation of input saliency was introduced in [9]. Instead of summarizing S_{ki} over the data set as in Eq. (9), the set of points $\mathscr{X}$ created by uniform sampling of the input space $\mathscr{D} \subset \mathbb{R}^I$ is used

$$S_{ki,\mathrm{sal}} \doteq \frac{\sum_{\mathscr{X}} |S_{ki}^{(n)}|}{N_{\mathscr{X}}}, \tag{11}$$

where $N_{\mathscr{X}}$ is the number of samples in $\mathscr{X}$. The saliency measure $[\mathrm{S}_{\mathrm{sal}}] = S_{ki,\mathrm{sal}}$ allows for better estimation of the sensitivity over the entire input space. However, in highly dimensional space, and when $\mathscr{D}$ does not have the shape of hypercube, the summation (11) may be difficult to perform due to the problems with generating the set $\mathscr{X}$. It would be computationally intensive to sample uniformly high-dimensional hypercube. Furthermore, training patterns in $\mathscr{D}$ may not cover the domain uniformly and/or some of the samples generated may not represent the desired properties of the network. Our proposed measures do not suffer from these potential limitations.

5. Criteria for pruning inputs

Inspection of the MSA sensitivity matrix $\mathrm{S}_{\mathrm{avg}}$ allows to determine which inputs affect outputs least. A small value of $S_{ki,\mathrm{avg}}$ in comparison to others means that for the particular k-th output of the network, the i-th input does not significantly contribute per average to output k, and therefore could be possibly disregarded. This reasoning and results of experiments allow to formulate the following practical rule: *The sensitivity matrices for a trained neural network can be evaluated for both training and testing data sets; the norms of MSA sensitivity matrix columns can be used for ranking inputs according to their significance and for reducing the size of network accordingly through pruning less relevant inputs.*

When one or more of the inputs have relatively small sensitivity in comparison to others, the dimension of neural network can be reduced by removing them, and a smaller-size neural network can be successfully retrained in most cases. The criterion used in this paper for an algorithm determining which inputs can be removed is based on the so called largest gap method.

Suppose two inputs are providing important data for a neural network. One of them has much larger relative change than the other one. In such case the sensitivity of the

second output would be much larger than the first one due to the necessity of an additional amplification of the input by network weights. In the extreme the first of those two inputs may even be selected for pruning. To prevent such cases additional scaling of matrices defined in Eqs. (8)–(10) is necessary or additional data preprocessing is another solution. In fact the latter seems to be better due to the fact that it prevents hidden layer neuron saturation at the beginning of the training when all their weights remain random, and therefore speed up the training. Formulas proposed in Eq. (12) allow to scale inputs into the range of $[-1;1]$. They were used in examples presented in this paper.

$$\begin{aligned}\hat{x}_i^{(m)} &\doteq \frac{x_i^{(m)} - \left(\left(\max\limits_{n=1\ldots N}\{x_i^{(n)}\} + \min\limits_{n=1\ldots N}\{x_i^{(n)}\}\right)/2\right)}{\left(\max\limits_{n=1\ldots N}\{x_i^{(n)}\} - \min\limits_{n=1\ldots N}\{x_i^{(n)}\}\right)},\\ \hat{o}_k^{(m)} &\doteq \frac{o_k^{(m)} - \left(\left(\max\limits_{n=1\ldots N}\{o_k^{(n)}\} + \min\limits_{n=1\ldots N}\{o_k^{(n)}\}\right)/2\right)}{\left(\max\limits_{n=1\ldots N}\{o_k^{(n)}\} - \min\limits_{n=1\ldots N}\{o_k^{(n)}\}\right)},\end{aligned} \tag{12}$$

where ^ denotes the normalized variable, or parameter.

Experiments were performed also for scaling inputs into range [0;1]. Similar results were achieved for the same learning conditions. The first scaling seems to accelerate slightly the convergence while accuracy and relations among sensitivities remain unchanged. If input and output data scaling (12) has been performed before network training, no additional operations on S_{ki} is required and we have

$$\hat{S}_{ki,\mathrm{avg}} \doteq S_{ki,\mathrm{avg}}. \tag{13}$$

Note that scaling can be performed either on entries of S or S_{avg}.

In case when network original inputs and outputs are not scaled to the same level as in Eq. (12), additional scaling (14) is necessary to allow for accurate comparison among inputs.

$$\hat{S}_{ki,\mathrm{avg}} \doteq S_{ki}\frac{\left(\max\limits_{n=1\ldots N}\{x_i^{(n)}\} - \min\limits_{n=1\ldots N}\{x_i^{(n)}\}\right)}{\left(\max\limits_{n=1\ldots N}\{o_k^{(n)}\} - \min\limits_{n=1\ldots N}\{o_k^{(n)}\}\right)}. \tag{14}$$

The significance measure of i-th input, Φ_i, across the entire set $\mathscr{X}$ is now defined as:

$$\Phi_{i,\mathrm{avg}} \doteq \max_{k=1\ldots K}\left\{\hat{S}_{ki,\mathrm{avg}}\right\}, \quad i = 1,\ldots,I-1. \tag{15}$$

Obviously, Φ_{abs} and Φ_{max} can be evaluated similarly to Φ_{avg} defined in Eq. (15) if other sensitivity measures are used. Note that searching for entries Φ_i, $i = 1,2,\ldots,I-1$, as in Eq. (15) corresponds to finding norms of column vectors of the normalized MSA sensitivity matrix $\hat{S}_{avg}$. This can be denoted as

$$\|S_i\|_\infty = \max_{k=1\ldots K}|\hat{S}_{ki,\mathrm{avg}}|, \quad i = 1,2,\ldots I-1. \tag{16}$$

In order to distinguish inputs with relative high and low importance and rank them properly, entries of Φ_i have to be sorted in descending order so that:

$$\Phi_{i_{m+1}} \leq \Phi_{i_m}, \quad m = 1, \ldots, I-2 \tag{17}$$

where $\{i_m\}$ is a sequence of sorted inputs. Note that the sensitivity measures S_{ki} and respective input significance measures, Φ_i, are abstract quantities. Practical heuristic algorithms need now be outlined based on which subsequent input pruning decisions can be made. One of such algorithms is outlined below based on the ratios of two neighboring terms of the sequence Φ_{i_m}.

Let us define the measure of gap as Eq. (18)

$$g_m \doteq \frac{\phi_{i_m}}{\Phi_{i_{m+1}}} \tag{18}$$

and then find the largest gap using the formula (19).

$$g_{\max} \doteq \max_m \{g_m\} \text{ and } m_{\text{cut}} \doteq m \text{ such that } g_m = g_{\max}. \tag{19}$$

Determining the largest gap, however, does not imply, that all inputs with coefficients lower than $\Phi_{m_{\text{cut}}}$ can be pruned. Whether or not inputs selected for pruning are actually contributing much to the neural network performance an additional criterion is necessary. An example of such criterion is given by Eq. (20) stating that second largest gap, $g_{\max\,\text{II}}$, has to be much smaller than that given by the formula (19). If condition (20) is valid, then the gap found between m_{cut} and $m_{\text{cut}}+1$ is large enough.

$$Cg_{\max} > g_{\max\,\text{II}}, \text{ where } g_{\max\,\text{II}} \doteq \max_{i_m \neq i_{m_{\text{cut}}}} \{g_m\}. \tag{20}$$

Constant C from Eq. (20) is chosen arbitrarily within the reasonable range (e.g. $C = 0.5$. The smaller C, the stronger is the condition for existence of the acceptable gap). All inputs with index $\{i_{m+1} \ldots i_{I-1}\}$ can be pruned with the smallest loss of information to the MFNN.

The gap method can be also applied for comparison among sensitivities of inputs to each output separately. For this purpose, a set containing candidates for pruning can be created for every output. Final pruning is performed by removing these inputs which can be found in every set determined previously for each output independently.

Obviously, S_{avg} can be meaningfully evaluated only for well trained neural networks. Despite this disadvantage, proposed criteria can still save computational effort when initial training is performed on smaller, but still representative subset of data. S_{avg} can then be evaluated based either on the data set used for initial training or on complete data set. Subsequently, newly developed neural networks with appropriate inputs can be retrained using the full set of training patterns with reduced dimension.

The importance of input i can be also determined statistically. The input saliency method referred to earlier [9] uses formula (21) and the averaging of the results over multiple training sessions.

$$\phi_{i,\text{sal}} \doteq \sum_{k=1}^{K} |\hat{S}_{ki,\text{sal}}|, \quad i = 1, \ldots, I-1. \tag{21}$$

The measure $\Phi_{i,\mathrm{sal}}$ is called input saliency. The i-th input is considered to be salient if the average saliency calculated over many training sessions is above the upper confidence boundary as described in more detail in [9]. Such statistical approach, although computationally intensive in comparison with proposed heuristic solution (19) allows for formulation of the theoretical criterion for unimportant feature removal.

6. Numerical examples

A series of numerical simulations was performed in order to verify the proposed perturbation-based approach and the pruning criteria. In the first experiment a training set for a neural network was produced using four inputs $x_1 \ldots x_4$ and two outputs o_1 and o_2. Values of output o_1 were correlated with x_1 and x_2, and of output o_2 with x_2 and x_3. Input vectors $\mathbf{x}$ (4×1) were produced using a random number generator. The expected values of vector $\mathbf{d}$ (2×1) for the output vector $\mathbf{o}$ (2×1) were evaluated for each $\mathbf{x}$ using a known relationship $\mathbf{d} = \mathbf{F}(\mathbf{x})$ where $\mathbf{d}$ is the desired (target) output vector for supervised training. The training set $\mathscr{X}$ consisted of $N = 81$ patterns. A neural network with 4 inputs, 2 outputs and 6 hidden neurons ($I = 5$, $J = 7$, $K = 2$) has been trained for the mean square error defined as in Eq. (22)

$$\mathrm{MSE} \doteq \sqrt{\frac{\sum_{n=1}^{N} \sum_{k=1}^{K} \left(d_k^{(n)} - o_k^{(n)}\right)^2}{N}} \tag{22}$$

equal to 0.001 per input vector. Matrices of sensitivities were subsequently evaluated and $\mathbf{S}_{\mathrm{avg}}$ produced at the end of training over the entire input data set $\mathscr{X}$.

The changes of MSA sensitivity entries during learning are presented in Fig. 3. It can be seen that initial sensitivities are low and apparently random positive numbers. During the training some of the average sensitivities $S_{ki,\mathrm{avg}}$ increase, while others converge towards low values. An obvious property can be seen that an untrained neural network in the example has per average smaller sensitivities than after the training. Final values

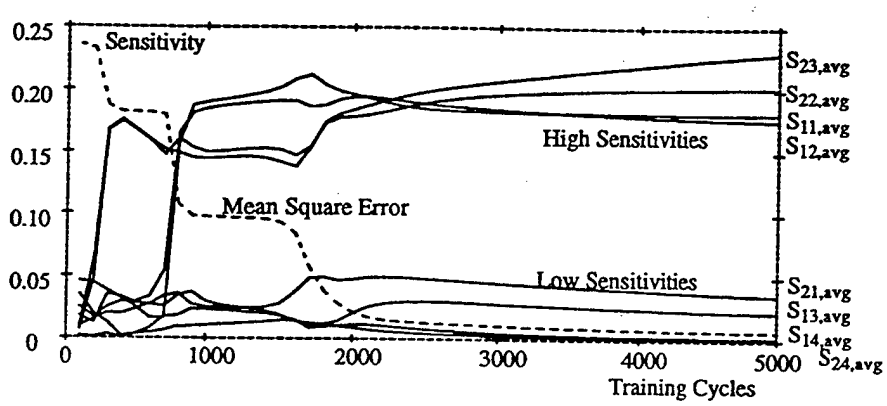


Fig. 3. Sensitivity profile during training for the full training set.

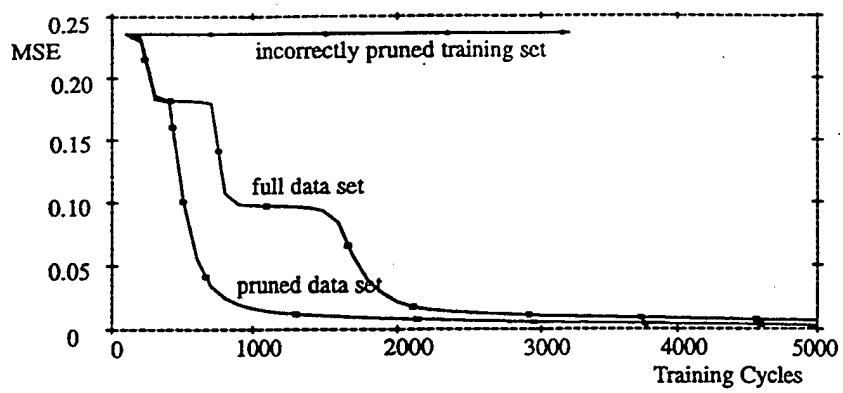


Fig. 4. Learning profile for full and pruned training sets.

of sensitivities of the first output offer hints for deleting x_3 and x_4, and these for the second output indicate that x_1 and x_4 could be deleted. The only input which then shows up in both sensitivity sets candidates for deletion is x_4. Therefore, the fourth input to the network can be eliminated and its dimension reduced to 3 inputs plus bias ($I = 4$).

The new network with three inputs was trained successfully after deleting x_4 from the training data set with the same accuracy. The learning profiles for full and reduced input sets for the same learning conditions are compared in Fig. 4. Not only the network with three inputs trains within a smaller number of cycles, but each learning cycle is performed faster due to the reduced input layer size.

If an input not recommended for pruning is erroneously deleted, the network is not able to learn the data sets. In our example the MSE value remains at the level of approximately 0.24 as it is shown in Fig. 4. Most entries of the sensitivity matrix remain low as shown in Fig. 5, which is indicative of poor network performance. A network erroneously trimmed is not able to learn accurately because some important relationships have been lost after pruning.

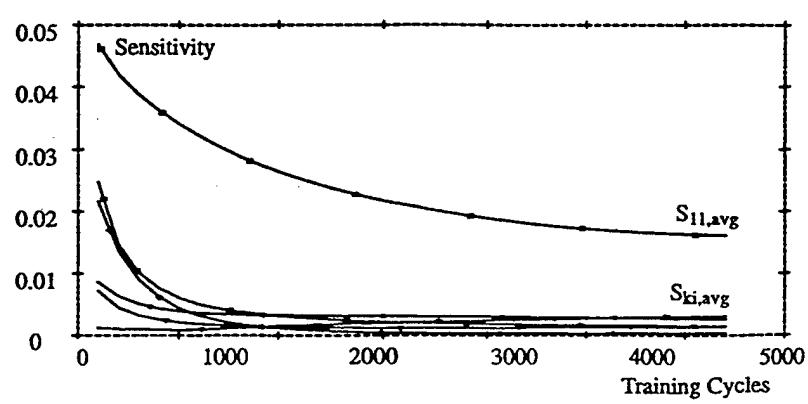


Fig. 5. Sensitivity profile during training for incorrectly trimmed training set.

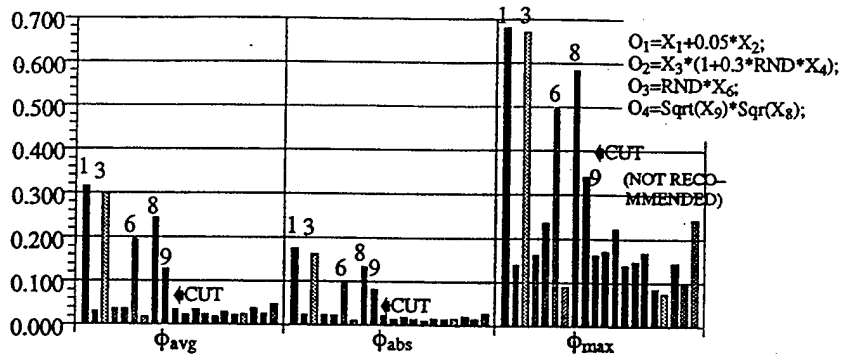


Fig. 6. Input significance ϕ evaluated using different overall sensitivities (8)–(10) and pruning criterion (20). (RND is a random function in $[-1;1]$.)

The second experiment was performed using a larger network and random data. MFNN had 20 inputs ($I=21$), 10 hidden neurons ($J=11$) and 4 outputs ($K=4$). There were $N=500$ patterns in the training set. Several additional data sets of the same size have been generated according to the same rule as the training set for performance evaluation. The network was successfully trained to the MSE of 0.15. However, due to the randomness of the data, MSE for additional sets remained at the level of $0.2\bar{0}$. All outputs were strongly correlated with inputs x_1, x_2, x_3, x_4, x_6, x_8, and x_9. Input x_6 during data generation was multiplied by random number, while the influence of x_2 and x_4 on outputs can be seen as scaled down in comparison to other inputs.

Table 1
Intermediate results of the pruning algorithm (refer to Fig. 6, Fig. 7 and Fig. 9)

$\{i_m\}$	$\phi_{avg,im}$	g_{im}	$\{i_m\}$	$\phi_{abs,im}$	g_{im}	$\{i_m\}$	$\phi_{max,im}$	g_{im}	$\{i_m\}$	$\phi_{cor,im}$	g_{im}
1	0.314	1.047	1	0.175	1.071	1	0.679	1.015	1	0.998	1.006
3	0.299	1.227	3	0.164	1.223	3	0.669	1.146	3	0.992	1.153
8	0.244	1.371	8	0.134	1.329	8	0.584	1.181	8	0.861	1.310
6	0.194	1.497	6	0.101	1.218	6	0.494	**1.454**	6	0.657	1.333
9	0.129	**3.013**	9	0.082	**2.979**	9	0.340	1.402	9	0.493	**1.681**
20	0.043	1.132	20	0.027	1.118	20	0.242	1.030	4	0.293	1.081
18	0.038	1.034	4	0.024	1.040	5	0.235	1.061	7	0.271	1.207
5	0.036	1.023	2	0.023	1.029	12	0.222	1.284	2	0.224	1.476
4	0.036	1.037	5	0.023	1.029	11	0.172	1.026	14	0.152	1.027
10	0.034	1.001	10	0.022	1.100	15	0.168	1.030	12	0.148	1.088
12	0.034	1.110	18	0.020	1.10	4	0.163	1.000	11	0.136	1.038
2	0.031	1.022	12	0.018	1.130	10	0.163	1.105	15	0.131	1.147
15	0.030	1.134	15	0.016	1.007	14	0.148	1.018	5	0.114	1.225
19	0.026	1.106	19	0.016	1.02	18	0.145	1.029	16	0.093	1.024
13	0.024	1.025	17	0.015	1.034	2	0.41	1.013	13	0.091	1.108
17	0.023	1.054	16	0.015	1.121	13	0.129	1.447	18	0.082	1.027
16	0.022	1.005	11	0.013	1.003	19	0.096	1.046	17	0.080	1.236
11	0.022	1.121	13	0.013	1.167	7	0.9092	1.078	10	0.064	1.186
14	0.020	1.108	7	0.011	1.034	16	0.085	1,137	19	0.054	1.147
7	0.018	–	14	0.011	–	17	0.075	–	20	0.047	–
$m_{cut}=5$			$m_{cut}=5$			$m_{cut}=$ none			$m_{cut}=$ none		
$g_{max}=3.013$ at $m=5$ $g_{max\ II}=1.497$ at $m=4$ $g_{max\ II}/g_{max}=0.497<C$			$g_{max}=2.979$ at $m=5$ $g_{max\ II}=1.329$ at $m=4$ $g_{max\ II}/g_{max}=0.446<C$			$g_{max}=1.454$ at $m=4$ $g_{max\ II}=1.447$ at $m=16$ $g_{max\ II}/g_{max}=0.995>C$			$g_{max}=1.681$ at $m=5$ $g_{max\ II}=1.476$ at $m=8$ $g_{max\ II}/g_{max}=0.878>C$		

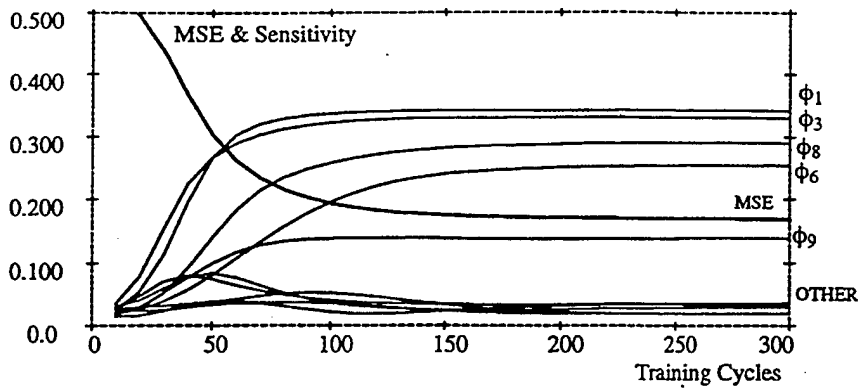


Fig. 7. Input significance ϕ_{avg} changes during training for the full training set.

The input significance measures calculated using formulas (8)–(10) are shown in Fig. 6. After sorting inputs x_2 and x_4 are ranked as even less important than other inputs which are not correlated at all. This occurred because of their low correlation with outputs, and they can be ignored as well as other inputs which show as uncorrelated for given MSE value as a final condition for training. The sequence of significance measures Φ_{avg}, Φ_{abs}, and Φ_{max}, are the same for all proposed coefficients, however, the size of gaps are different in each case.

Table 1 summarizes numerical results. It lists the sequence $\{i_m\}$ and values g_{im}. Note that $m_{cut} = 5$ and $g_{max} = 3.013$. C was selected arbitrarily as 0.5. Note that value $C = 0.5$ would prevent pruning using ϕ_{max} definition. Also note that the maximum method does not provide a clear clue where to locate the gap for purging due to fuzziness of the training data.

The result of initial training is shown in Fig. 7. It can be determined from this figure which inputs should remain active after pruning. The network performance after pruning is shown in Fig. 8. No additional dimension reduction is advisable because no large gap in input importance is found. The speed of training has increased mostly because of the

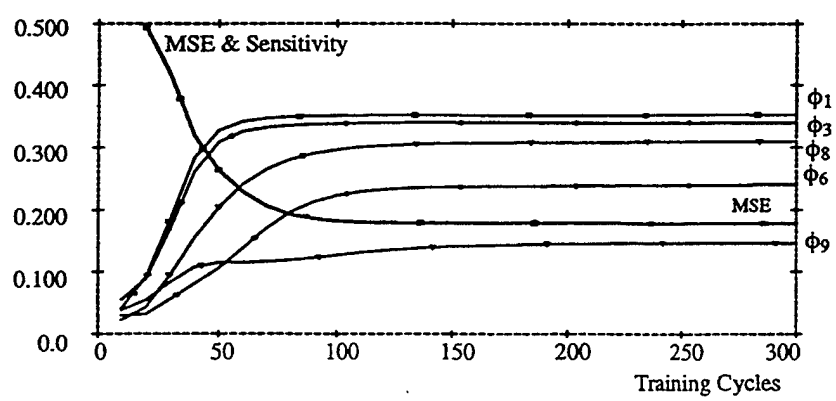


Fig. 8. Input significance ϕ_{avg} changes during training for the pruned training set.

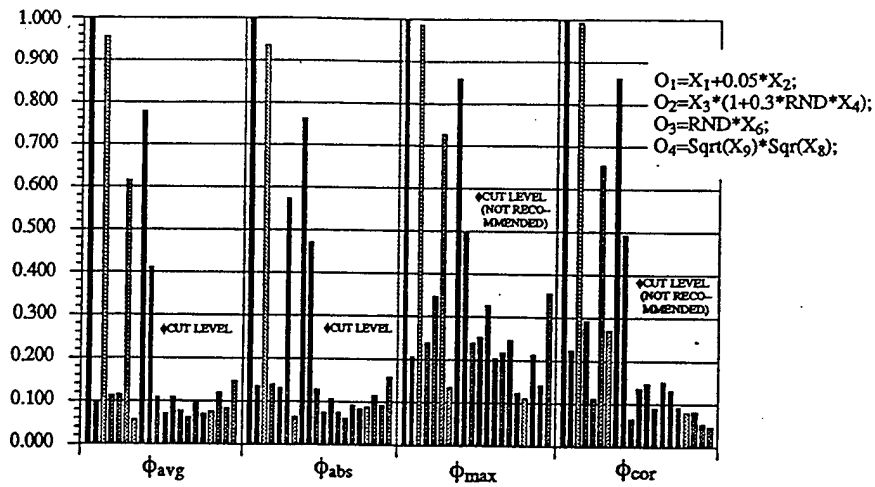


Fig. 9. Normalized input significance coefficients ϕ for different sensitivities (8)–(10) and pruning criterion (20) in comparison to significance coefficients evaluated using standard correlation method ϕ_{cor}. Significance coefficients are normalized. (RND is a random function with uniform distribution in $[-1;1]$.)

reduction of the MFNN size (input dimension reduced to 25%). The necessary number of training cycles has also decreased, but not so dramatically as in the first experiment.

An alternative approach to the presented perturbation based method is offered through correlation computation. Input significance coefficients $\Phi_{i,\mathrm{cor}}$, can be computed from the definition (24) based on sensitivity matrix entries given by Eq. (23).

$$S_{ki,\mathrm{cor}} \doteq \frac{\sum_{n=1}^{N}\left(o_k^{(n)}-\bar{o}_k\right)\left(x_i^{(n)}-\bar{x}_i\right)}{\sum_{n=1}^{N}\left(o_{k_{(n)}}-\bar{o}_k\right)\sum_{n=1}^{N}\left(x_i^{(n)}-\bar{x}_i\right)}, \tag{23}$$

$$\Phi_{i,\mathrm{cor}} \doteq \max_{k=1\ldots K}\{S_{ki,\mathrm{cor}}\}. \tag{24}$$

Note that this approach requires additional computational effort and it makes use of data only. This is in contrast to the proposed method of calculation of sensitivities and input significance coefficients which requires rather the use of the network model and partially data as well (input vectors only). Fig. 9 illustrates that both methods compare consistently with each other and yield comparable results.

Another experiment was performed using the IRIS data set. That set was first published by Fisher [13] and has been used widely as a testbed for statistical analysis techniques. The sepal length, sepal width, petal length, and petal width were measured on 50 iris specimens from each of 3 species, Iris setosa, Iris versicolor, and Iris virginica. The data set was divided randomly into a training data set containing 100 entries, and a testing data set containing the remaining 50 entries.

As mentioned, the proposed sensitivity method does not apply directly to neural network classifiers, but can still offer guidelines as to the ranked importance of inputs.

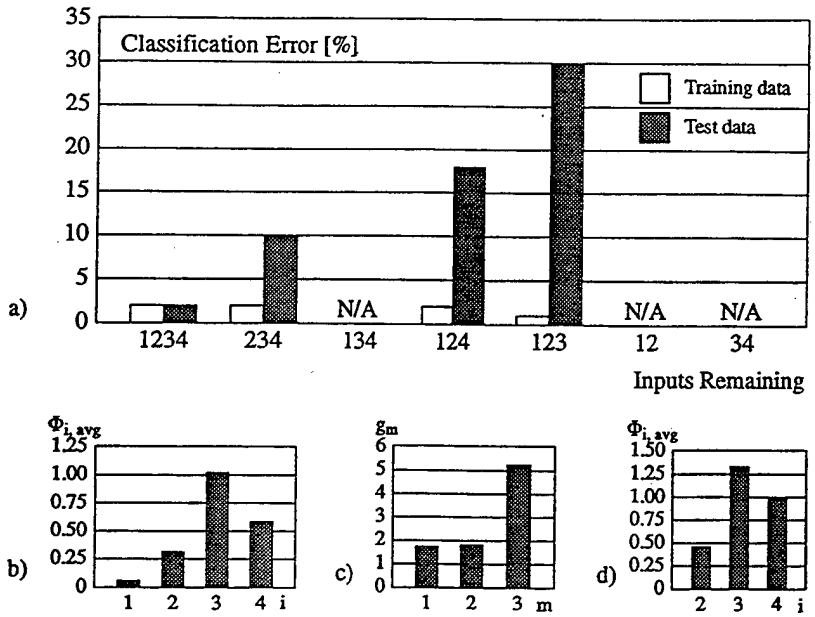


Fig. 10. Neural network training using IRIS data set and termination conditions set by MSE: (a) percentage of misclassification for training and testing data sets, MSE = 0.05; (b) coefficients Φ_i for the complete training data set, MSE = 0.01; (c) gap sizes for sorted input significance coefficients (b), MSE = 0.01; (d) coefficients Φi after removing input No. 1, MSE = 0.05.

Classifier in this example was trained for desired output values of -0.5 and 0.5. Although placing neuron outputs outside of the saturation region deteriorates the classifier's performance, it allows to use sensitivity method for input pruning.

Fig. 10 summarizes the result of computational experiment. As can be seen from Fig. 10(a), pruning a single input, No. 1, causes the increase of error to 10%, while removing inputs No. 3 and No. 4 leads to error of 18 and 30%, respectively. In addition, removing input No. 2 makes it impossible to train the classifier. The pruning algorithm results in significance coefficients as in Fig. 10(b), and the gap sizes as in Fig. 10(c). After sorting coefficients Φ_i into Φ_{im} and evaluating gap sizes $g_m m_{\text{cut}}$ was set to 3 according to the formula (19). Input number $i_4 = 1$ can be pruned because condition (20) is satisfied. Correlation analysis performed on IRIS data has not lead to a clear indication of which inputs can be pruned (see Fig. 11).

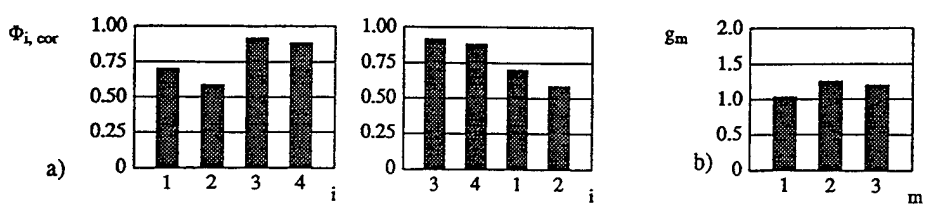


Fig. 11. Correlation between inputs and outputs for IRIS training data set: (a) input significance coefficients for the complete training data set; (b) gap values for sorted inputs (a).

7. Conclusions

Using the perturbation-based sensitivity approach for input layer pruning seems particularly useful when network training involves a large amount of redundant data. In the first phase, a network can be pre-trained until the training error has decreased satisfactorily. Then, sensitivity matrices can be evaluated and dimension of the input layer possibly reduced. Training can subsequently be resumed until the training error reduces to an acceptable value. This process can be repeated, however, usually only the first execution yields significant improvement. Numerical experiments indicate that an effort of further network retraining beyond the first pass can be too high in comparison to benefits of further minimization.

Should the redundancy in training data vectors exist, the proposed approach based on the average sensitivity matrices for input data pruning allows for building more efficient perceptron network models. This can be achieved at a relatively low computational cost and based on heuristic pruning criteria outlined in the paper. The approach proposed here is somewhat similar to the principal component analysis in the sense that it detects directions of basis vectors and their relative importance. In contrast to the eigenanalysis for the largest eigenvectors, basis vectors here are fixed. They correspond to the basis vectors in which the original training data are formulated. As such, the approach is aimed at identifying basis vectors yielding minimal projections in the fixed input space dimensions.

The applicability and significance of the presented method is mostly for continuous and differentiable mappings. The method would be even more useful if it allowed additionally merging inputs which are totally correlated on an input set while yielding same target responses. In addition, further extension of the proposed sensitivity-based input pruning approach for binary output networks such as classifiers and binary encoders would be desirable.

Acknowledgements

The authors express their thanks to the reviewers for their critical and constructive suggestions which have resulted in significant improvement of the paper.

References

[1] K.M. Hornik, M. Stinchcombea and H. White, Multilayer feedforward networks are universal approximators, *Neural Networks* 2 (1989) 359–366.

[2] K.I. Funahashi, On the approximate realization of continuous mappings by neural networks, *Neural Networks* 2 (1989) 183–192.

[3] J.M. Zurada, *Introduction to Artificial Neural Systems* (West Publishing Company, St. Paul, Minn., 1992).

[4] P.A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach* (Prentice-Hall, Englewood Cliffs, NJ, 1982).

[5] E.D. Karnin, A simple procedure for pruning back-propagation trained neural networks, *IEEE Trans. on Neural Networks* 1(2) (1990).

[6] M.C. Mozer and P. Smolensky, Skeletonization: A technique for trimming the fat from a network via relevance assessment, in: D.S. Touretzky (ed.), *Advances in Neural Information Processing I* (Morgan Kaufmann, 1989) 107–115.

[7] J.Y. Choi and C.H. Choi, Sensitivity analysis of multilayer perceptron with differentiable activation functions, *IEEE Transactions on Neural Networks* 3(1) (1992) 101–107.

[8] D.W. Ruck, S.K. Rogers and M. Kabrisky, Feature selection using a multilayer perceptron, *Neural Network Comp.* 20 (1990) 40–48.

[9] L.M. Belue and K.W. Bauer, Jr., Determining input features for multilayer perceptrons, *Neurocomputing* 7 (1995) 111–121.

[10] L. Fu and T. Chen, Sensitivity analysis for input vector in multilayer feedforward neural networks, *Proc. of IEEE International Conference on Neural Networks I* (San Francisco, CA, March 28–April 1, 1993) 215–218.

[11] J.M. Zurada, A. Malinowski and I. Cloete, Sensitivity analysis for pruning of training data in feedforward neural networks, *Proc. of First Australian and New Zealand Conference on Intelligent Information Systems* (Perth, Western Australia, December 1–3, 1993) 288–292.

[12] J.M. Zurada, A. Malinowski and I. Cloete, Sensitivity analysis for minimization of input data dimension for feedforward neural network, *Proc. of IEEE International Symposium on Circuits and Systems* (London, May 28–June 2, 1994) 447–450.

[13] R.A. Fisher, The use of multiple measurements in taxonomic, *Annals of Eugenics* 7 (1936) 179–188.

Jacek M. Zuradu is the S.T. Fife Alumni Professor of Electrical Engineering at the University of Louisville, Louisville, Kentucky. He is the author of the 1992 PWS text *Introduction to Artificial Neural Systems*, contributor to the 1993 and 1994 Ablex volumes *Progress in Neural Networks*, and co-author of the 1994 IEEE Press volume *Computational Intelligence: Imitating Life*. He is the author or co-author of more than 100 journal and conference papers in the area of neural networks, analog and digital VLSI circuits, and active filters. Dr. Zurada is now an Associate Editor of *IEEE Transactions on Neural Networks*, of *IEEE Transactions on Circuits and Systems, Part II*, *Neurocomputing*, and of the *Artificial Neural Networks Journal*. Dr. Zurada received a number of awards for distinction in research and teaching, including the 1993 Presidential Award for Research, Scholarship, and Creative Activity. He is a Fellow of IEEE.

Aleksander Malinowski was born in Gdansk, Poland on September 27, 1966. He received his M.Sc. degree in Electronics from the Technical University of Gdansk in 1990, and is currently working toward the Ph.D. degree in Computer Science and Engineering at the University of Louisville, Louisville, Kentucky. His research interests include artificial neural networks for control and modeling, neural network architecture optimization, and computer simulation and modeling. He has co-authored 23 journal and conference papers. He has been recipient of University of Louisville Fellowship in 1992-1996.

Shiro Usui was born in Qingdao, China, on October 22, 1943. He received his Ph.D. degree in Electrical Engineering and Computer Science from the University of California at Berkeley in 1974. From 1974 until 1979, Dr. Usui was a Lecturer in Electrical Engineering at Nagoya University. In 1979, he moved to the Department of Information and Computer Sciences at the Toyohashi University of Technology, Toyohashi-shi, Japan, where he is currently Professor, and Head of the Computer Center and Biological and Physiological Engineering Laboratory. His Laboratory pioneered a new interdisciplinary field, Physiological Engineering, to further understanding biological and physiological systems by combining electro-physiology with information sciences. He is presently serving on the Board of Directors for the JNNS and JSME & BE, and is President of the Biological and Physiological Engineering Group of SICE, Japan. He is a Fellow of IEEE and a member of IEICE of Japan, INNS and the Physiological Society of Japan.

Westin Galleria Hotel, Houston, Texas, USA
TUTORIALS: June 8, 1997 CONFERENCE: June 9-12, 1997

Inverse Mapping of Continuous Functions using Feedforward Neural Networks

Hatem M. Deif*, Jacek M. Zurada*, and Wojciech Kuczborski**
*Department of Electrical Engineering, University of Louisville
Louisville, Kentucky 40292, j.zurada@ieee.org
** Department of Computer and Communication Engineering,
Edith Cowan University, Joondalup, Australia.

1997 International Conference on Neural Networks (ICNN '97)
Westin Galleria Hotel, Houston, Texas, USA
TUTORIALS: June 8, 1997 CONFERENCE: June 9-12, 1997

1997 International Conference on Neural Networks (ICNN '97)
Westin Galleria Hotel, Houston, Texas, USA
TUTORIALS: June 8, 1997 CONFERENCE: June 9-12, 1997

Abstract

In this paper we present a methodology for solving inverse mapping of continuous functions modeled by multilayer feedforward neural networks. The methodology is based on an iterative update of the input vector towards a solution, which escapes local minima of the error function. The update rule is able to detect local minima through a phenomenon called "update explosion." The input vector is then relocated to a new position based on a probability density function (PDF) constructed over the input vector space. The PDF is built using local minima detected during the past search history. Simulation results demonstrate the effectiveness of the proposed method in solving the inverse mapping problem for a number of cases.

1. Introduction

It is known that multilayer feedforward neural networks can be trained to approximate continuous functions with a high degree of accuracy [1]. In many engineering applications, optimization problems need to be solved that require inverse solutions for nonlinear systems. Also, inverse mapping has important implications in cognitive and mental processes.

The inverse mapping considered here aims at relating an M-dimensional output space to an N-dimensional input space. The problem of inverse mapping can be stated as follows: Given the desired output vector $\mathbf{y}^d$, generate an input vector $\mathbf{x}$ that satisfies the forward mapping $\mathbf{y}(\mathbf{x}) = \mathbf{y}^d$. Note that, in general, more than one solution can exist.

For special cases of nonlinear functions where a convex error function can be defined over the input space, we can use iterative approaches such as the steepest descent method, Newton's method, the conjugate gradient method, etc. [3]. In cases when a convex error function can not be generated, stochastic optimization such as simulated annealing [4-6], or an efficient global search such as sub-energy tunneling and terminal repelling [7] can be used.

In this paper we propose a new approach for obtaining inverse mapping of a continuous function based on an iterative update of the input vector, while escaping from local minima. The update rule is determined by the pseudo-inverse of the gradient of the Lyapunov function. The update rule is able to detect local minima by generating an explosive amount of update at a local minimum, called "update explosion." The input vector is then relocated to a new position based on a probability density function (PDF) constructed over the input vector space. The PDF is built gradually using the local minima detected during the search process which helps in relocating the trajectory to a point that is close to the global minimum.

This work was supported in part by the ONR Grant N00014-93-1-0855

2. Input vector update

The input vector update rule can be formulated based on Lyapunov function of the considered system. The rule provides fast convergence to a global minimum while detecting local minima.

Assume $\mathbf{x}$ to be an N-dimensional input vector defined over the compact set I, where $I = [-a, a]^N \in R^N$, while $\mathbf{y}(\mathbf{x})$ represents the mapping learned by the neural network model, and $\mathbf{y}$ is the actual M-dimensional output vector corresponding to the input vector $\mathbf{x}$. Next, select the Lyapunov function V as follows:

$$V = \frac{1}{2}\|\tilde{\mathbf{y}}(\mathbf{x})\|^2, \tag{1}$$

where $\tilde{\mathbf{y}}(\mathbf{x})$ is an M-dimensional output error vector defined as $\tilde{\mathbf{y}}(\mathbf{x}) = \mathbf{y}^d - \mathbf{y}(\mathbf{x})$.

From (1), the time derivative of Lyapunov function can be expressed as follows:

$$\dot{V} = \left(\frac{\partial V}{\partial \mathbf{x}}\right)^T \dot{\mathbf{x}} = -\tilde{\mathbf{y}}^t \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \dot{\mathbf{x}} = -\tilde{\mathbf{y}}^t J \dot{\mathbf{x}} \tag{2}$$

where J represents the Jacobian matrix.

To form the input vector update rule, consider the following system [8]:

$$\dot{\mathbf{x}} = -\frac{V}{\|\frac{\partial V}{\partial \mathbf{x}}\|^2}\frac{\partial V}{\partial \mathbf{x}} = \frac{1}{2}\frac{\|\tilde{\mathbf{y}}\|^2}{\|J^t\tilde{\mathbf{y}}\|^2} J^t \tilde{\mathbf{y}}, \qquad \forall \tilde{\mathbf{y}} \neq 0. \tag{3}$$

The above system has such a property that $\dot{\mathbf{x}}$ in (3) keeps $\dot{V}$ in (2) negative throughout the convergence, as $\dot{V} = -V$. Furthermore, when $\mathbf{x}$ approaches a local minimum of $V(\mathbf{x})$, i.e., $\partial V/\partial \mathbf{x} = 0$ but $V(\mathbf{x}) \neq 0$, $\dot{\mathbf{x}}$ takes an excessive value. On the other hand, when $\mathbf{x}$ approaches a global minimum, i.e., $\partial V/\partial \mathbf{x} = 0$ and $V(\mathbf{x}) = 0$, $\dot{\mathbf{x}}$ converges to zero. Also note that, in (3), $(1/\|\partial V/\partial \mathbf{x}\|^2)(\partial V/\partial \mathbf{x})$ represents a pseudo-inverse [2] of $(\partial V/\partial \mathbf{x})^t$.

In the following, a novel update rule that assures convergence of the trajectory to local minima is proposed. The convergence to local minima achieved here is sufficient for detecting them. The update rule is expressed as follows:

$$\delta \mathbf{x}^i = \eta^i \dot{\mathbf{x}}^i \tag{4}$$

where subscript i represents the update step, η^i is the update coefficient initiated at $\eta^0 = 1$, and $\dot{\mathbf{x}}^i$ is given by equation (3). The convergence is accomplished by adaptively modifying the update coefficient η at each update step based on observing the trajectory. More precisely, η is decreased whenever the update step is too large to reach a local minimum. Update steps are considered to be too large when the trajectory changes direction. To measure direction changes, let P be a metric that represents the percentage of vector entries that change sign in the update step i.

$$P = \frac{N - sgn(\dot{\mathbf{x}}^i) \cdot sgn(\dot{\mathbf{x}}^{i-1})}{2N} \tag{5}$$

Note that P has a maximum of 1 when all of the elements in $\dot{\mathbf{x}}^i$ have an opposite sign to those of $\dot{\mathbf{x}}^{i-1}$, and a minimum of 0 when all corresponding elements of the two vectors agree in sign. Consider the following formula for η^i

$$\eta^i = \eta^{i-1}(1 - \alpha P) \tag{6}$$

where α is a fraction heuristically chosen to be 0.5 in order to guarantee that η will not be reduced by more than 50% at any update step. Besides providing a convergence to local minima, the update rule in (4) also provides accurate convergence to a global minimum once it is detected.

A local minimum is detected when the "update explosion" phenomenon occurs, i.e., when the magnitudes of all entries of the gradient $\partial V/\partial \mathbf{x}$ become less than a threshold S. For bipolar neurons, V has

a maximum of N, and S is heuristically recommended to be ($S < N/10a$). The update explosion can be expressed as follows:

$$||\dot{x}|| > \beta \tag{7}$$

where β is the explosion threshold heuristically estimated as $\beta \approx \frac{\sqrt{N}}{S}$ by substituting $\partial V/\partial x_i = S$ and $V = N$ in equation (3).

3. Escaping from local minima

In the previous section, local minima detection technique has been presented. In this section, we investigate a method for relocating the input vector whenever a local minimum is encountered along the input trajectory. The following new approach is based on [8] but differs in the form of the PDF used and in the manner in which it is utilized.

The idea is to search randomly for a relocation vector that will guide the trajectory to a global minimum (solution). The random search is based on a PDF constructed over the input vector space $\mathbf{x}$ based on the local minima detected during the past search history. The value of the PDF around a local minimum detected by the search process is reduced based on a function located at the local minimum, while it is increased over the rest of the input vector space through normalization. To provide a better chance of convergence to a solution, the input vector is relocated to the point with the highest value of PDF.

Formally, the PDF at the nth relocation (n^{th} local minimum), $p_n(\mathbf{x})$, can be expressed as follows:

$$p_n(\mathbf{x}) = \gamma(1 - \sum_{i=1}^{n} g_i(\mathbf{x})) \tag{8}$$

and

$$g_i(\mathbf{x}) = e^{-\frac{1}{2}\frac{||\mathbf{x}-\mathbf{m}_i||^2}{h_i^2}} \tag{9}$$

where $g_i(\mathbf{x})$ is a symmetric Gaussian function defined over the input space $\mathbf{x}$, vector $\mathbf{m}_i$ represents the ith local minimum, h_i is the standard deviation of $g_i(\mathbf{x})$, and γ is a normalization factor. In this sense, the PDF is modified whenever a local minimum is detected to avoid repeated convergence to the same local minimum.

To calculate h_i, assume the attraction domain Ω_i associated with the local minimum m_i to be an N-dimensional sphere of radius r_i. The attraction domains are assumed to be non-overlapping.

$$\Omega_i \cap \Omega_j = \phi \qquad \forall i \neq j. \tag{10}$$

where ϕ is the empty set. Since the value of g_i becomes negligible at $5h_i$ apart from the mean, an estimate of h_i that will satisfy the previous condition is chosen to be $h_i = r_i/5$. We will also assume Ω_1 to be the smallest expected attraction domain. Accordingly, a safe value for h_1 is heuristically assumed to be $h_1 \approx a/100$. As will be shown, the proposed algorithm will, however, increase the value of h_1 (if necessary) automatically to better represent the actual minimum m_1.

Next, h_i satisfying (10) is found:

$$h_i = \frac{||m_i - m_k||}{10} \tag{11}$$

where m_k is a local minimum closest to m_i, i.e., $||m_i - m_k|| < ||m_i - m_j|| \quad \forall j \neq i, k$. Also, to satisfy condition (10), if h_k is larger than h_i, it should be reset to h_i.

In the following, we present a practical method for generating samples from the PDF, $p_n(\mathbf{x})$, stated in equation (8). The method is inspired by the *acceptance-rejection method* due to Von Neumann [9]. To implement the method, let's define a univariate uniform distribution $u(0,1)$, and an N-dimensional

multivariate uniform distribution $v(\mathbf{x})$ over I, Also let $f_x(\mathbf{x})$ be the PDF scaled such that $0 < f_x(\mathbf{x}) \leq 1$. Then, we generate a random variate U and a random vector X from $u(0,1)$ and $v(\mathbf{x})$, respectively, and follow the following test to see whether the inequality $U \leq f_x(X)$ holds:

1. If the inequality holds, accept X as a vector generated from the PDF.
2. If the inequality is violated, reject the pair U, X and try again.

A sufficient number of samples to be generated was experimentally found to be $n_s \approx 10N^5$ where N is assumed to be less than 10. The relocation discussed above represents a beginning of a new step in the search process. Accordingly, before using formula (4), η should be reset to 1.

4. Modifications

So far, the relocation of the input vector trajectory was based only on the PDF, $p_n(\mathbf{x})$. In this section, the input vector trajectory is relocated to the sample vector X that has the highest value of a criterion function, $\psi(X)$, selected as follows:

$$\psi(X) = f_x(X) + \mu(X) + \nu(X), \tag{12}$$

$$\nu(X) = -\frac{\|X - m_i\|}{20a\sqrt{N}}, \tag{13}$$

$$\mu(X) = -\frac{\|\tilde{y}(X)\|}{20\sqrt{N}} \tag{14}$$

where $f_x(X)$ is the PDF scaled such that $0 < f_x(\mathbf{x}) \leq 1$, while the factors $\mu(X)$ and $\nu(X)$ represent the new modification. Consider a number of samples generated from the PDF having equally high PDF values. A relocation to the closest sample to the current local minimum will help in detecting the closest next local minimum and, consequently, provide a better estimation of local minima's attraction domains, Ω's. This is achieved by adding the new factor $\nu(X)$ to the criterion function, $\psi(X)$. On the other hand, a sample that has a small value of the Lyapunov function V is more likely to be close to a minimum. The trajectory from such a sample to this minimum is shorter than others. A relocation to this sample will save computation time and provide faster convergence. The factor $\mu(X)$ is added to the criterion function, $\psi(X)$, for this purpose. The constants in equations (13), (14) are chosen such that $\nu(X)$ and $\mu(X)$ have a maximum contribution of 10% to the criterion function $\psi(X)$.

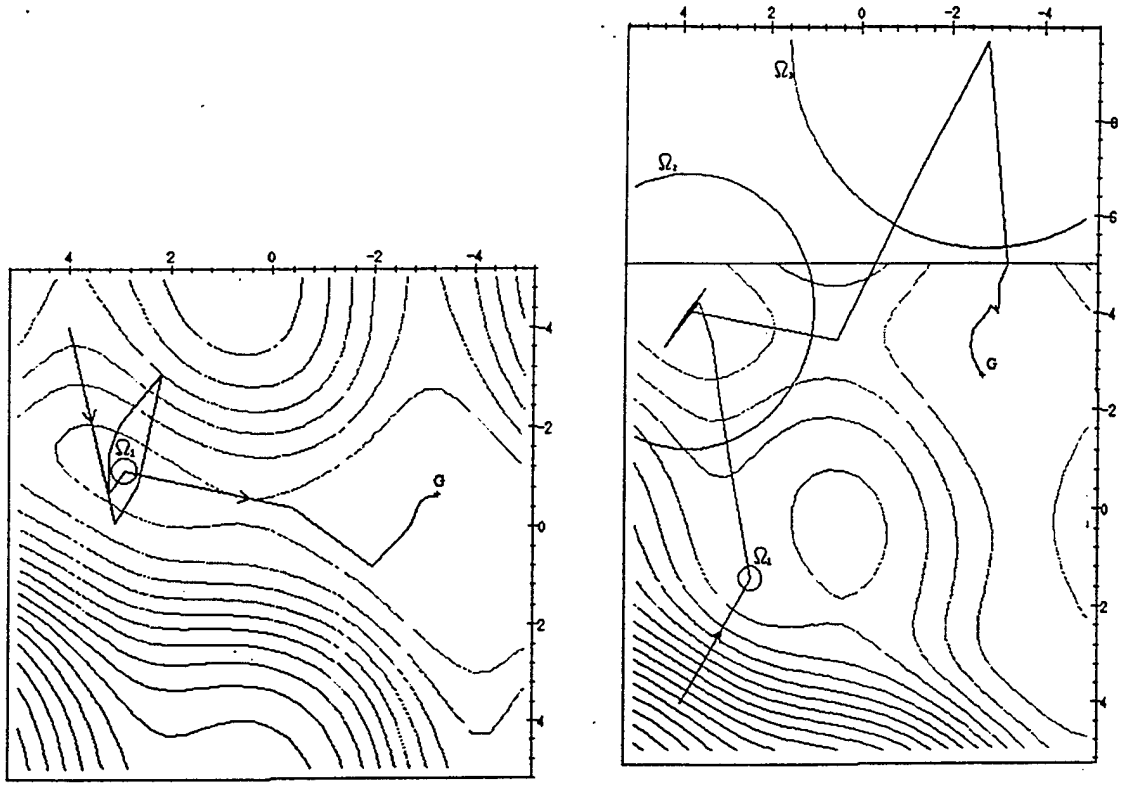
Secondly, if an update explosion occurs very close to a previously detected local minimum m_i, say within a distance of one tenth of its standard deviation h_i, the point is not considered a new local minimum. Instead, the situation is interpreted to be a repeated fall in the same local minimum. As a consequence, h_i is increased, say by 20%, to better represent the actual attraction domain of this local minimum.

Finally, we treat the point where the input vector trajectory goes out of input range, I, as a local minimum. This avoids repeated convergence to the same point.

5. Example simulations

A software program was developed to implement the new algorithm. Many cases of continuous functions have successfully been tested. In this section, we present two experiments based on a two-layer feedforward neural network, which maps a two-dimensional input space to a two-dimensional output space, i.e., $N = M = 2$. The network used in the first experiment has six neurons in its hidden layer, while the second one has eight neurons. Bipolar sigmoidal neurons were used in both experiments. The augmented weight matrices of the first neural network are:

$$\mathbf{W}_1^T = \begin{bmatrix} 1 & 1 & 1 & 1 & .1 & .1 \\ .1 & .1 & .1 & .1 & 1 & 1 \\ -4 & -3 & 3 & 4 & -1 & 1 \end{bmatrix}, \quad \mathbf{W}_2 = \begin{bmatrix} 3.2 & -3.2 & 2 & -2 & 2 & -2 & 1.5 \\ .5 & .6 & .5 & .2 & .8 & .7 & 2.1 \end{bmatrix} \tag{15}$$



(a) (b)

Fig. 1: Lyapunov functions and the input trajectories. (A) Experiment 1 with a 2-6-2 model. (b) Experiment 2 with a 2-8-2 model.

while those of the second network are:

$$\mathbf{W}_1^T = \begin{bmatrix} 1 & 1 & 1 & 1 & .1 & .1 & .1 & .1 \\ .1 & .1 & .1 & .1 & 1 & 1 & 1 & 1 \\ -4 & -3 & 3 & 4 & -4 & -3 & 3 & 4 \end{bmatrix}, \quad \mathbf{W}_2 = \begin{bmatrix} 3.2 & -3.2 & 2 & -2 & 2 & -2 & 2 & -2 & 2 \\ .5 & .6 & .5 & .2 & .8 & .7 & .8 & .7 & 2.1 \end{bmatrix} \tag{16}$$

where $\mathbf{W}_1$ represents hidden layer weights, and $\mathbf{W}_2$ represents output layer weights.

The input space in both cases is defined by the compact set $I = [-5, 5]^2$. The Lyapunov functions and input trajectories for the first and second experiments are illustrated in Figures 1a and 1b, respectively. In both cases, the starting point was selected arbitrarily and a solution was found very accurately at points having an rms error of 0.0001. In the two experiments, β in equation (7) was chosen to be 10 and h_1 was assumed to be 0.05. The attraction domains corresponding to local minima are shown by circles in the two figures.

In the first experiment, Figure 1a shows the successful convergence of the trajectory to the local minimum, m_1, based on the new proposed update rule. Once the local minimum is detected, the input vector is relocated to a point that guides the trajectory to the solution, G. In Figure 1b, the trajectory detects a saddle point, m_1, upon the first update. The saddle point is treated the same as a local minimum. Next, the trajectory detected a local minimum at m_2 then gets out of range at m_3, which is also treated the same as a local minimum. Finally, the trajectory detects the global minimum, G.

6. Conclusion

In this paper, a novel algorithm for obtaining inverse mapping of continuous functions learned by multilayer feedforward neural networks is presented. In numerous numerical experiments it has been found that the introduced input vector update with variable update coefficient assures accurate detection of local minima as well as accurate convergence to a global minimum (solution). Furthermore, we presented a fast method of escaping from local minima and reaching a solution, based on a PDF constructed over the input vector space and a proposed criterion function. Simulation results demonstrate the effectiveness of the proposed method in providing correct and efficient inverse mapping for various continuous functions. The method is applicable to algorithms of design centering and yield optimization as referenced in [10-11].

References

[1] J. M. Zurada, *Introduction to Artificial Neural Systems.* Boston: PWS, 1992.

[2] A. Albert, *Regression and the Moore-Penrose Pseudoinverse.* New York: Academic, 1972.

[3] D. G. Luenberger, *Introduction to Linear and Nonlinear Programming.* Reading, MA: Addison-Wesley, 1973.

[4] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies," *J. Stat. Phys.*, vol. 34, pp. 975-986, 1984.

[5] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, " A learning algorithm for Boltzman machines," *Cognitive Science*, vol. 9, pp. 147–169, 1985.

[6] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications.* Reidel, 1987.

[7] J. W. Burdick, B. C. Cetin, and J. Bahren, "Efficient global redundant configuration resolution via sub-energy tunneling and terminal repelling," *Proc. IEEE Int. Conf. Robotics and Automation*, Sacramento, California - April 1991, pp. 939–944.

[8] S. Lee and R. M. Kil, "Inverse mapping of continuous functions using local and global information," *IEEE Trans. Neural Networks*, vol. 5, no. 3, pp. 409–423, May 1994.

[9] J. von Neumann, "Various techniques used in connection with random digits," *U.S. Nat. Bur. Stand Appl. Math. Ser.*, No. 12, pp. 36-38, 1951

[10] J. M. Zurada, A. Lozowski, and A. Malinowski, "Yield improvement in GaAs IC manufacturing using neural network inverse modeling," Submitted to *Proc. IEEE Int. Conf. Neural Networks*, Houston, Texas - February 1-8, 1997.

[11] J. M. Zurada, A. Lozowski, and A. Malinowski, "Design centering in GaAs IC manufacturing", Accepted to *Proc. IEEE Aerospace. Conf.*, Snowmass, Colorado - June 1997.

1997 International Conference on Neural Networks (ICNN '97)
Westin Galleria Hotel, Houston, Texas, USA
TUTORIALS: June 8, 1997 CONFERENCE: June 9-12, 1997

Yield Improvement in GaAs IC Manufacturing Using Neural Network Inverse Modeling

Jacek M. Zurada, Andrzej Lozowski, Aleksander Malinowski
Department of Electrical Engineering, University of Louisville
Louisville, KY 40292, ph: 502-852-6314
jmzura02@starbase.spd.louisville.edu

Abstract—This paper describes a neural network based method of design centering for microelectronic circuits fabrication process. Process data are first evaluated for principal components and subsequently modeled using multilayer perceptron networks in a reduced and transformed input space. Perceptron network models are then inverted, and center settings of input variables are computed by using the inverse PCA transformation. The approach allows for maximizing the fabrication yield of GaAs circuits used in aviation electronics systems. Example of yield maximization for MMIC fabrication data is provided to illustrate the proposed technique.

1. Introduction

The design of the microelectronic integrated circuits involves consideration of the fabrication cost and leads to the tradeoff between system specifications, such us complexity and frequency requirements, and acceptable fabrication yield [1]. The yield maximization of GaAs Microwave/Millimiter Wave Monolithic Integrated Circuits (MMIC) with respect to the material, process, and device parameters is the objective of this paper.

The fabrication process model identification is an important step in the proposed design centering approach [2]. Stages of the microelectronic circuit fabrication process can be efficiently modeled with multilayer perceptron neural networks (NN) and the Principal Component Analysis (PCA) of underlying data. These methods are found to be useful for capturing the relationships between various stages in the manufacturing process as well as between the process parameters and the resulting device parameters. Once the model is identified, a practical degree of design centering can be achieved by inverse modeling. In practice, the design centering problem requires the solution of the desired values of early manufacturing parameters (or process attributes) given the target performance and tolerance of the final product.

The design centering approach introduced here is employed to improve the gate-final stage yield of GaAs 0.5mm$\times$200μm MESFET fabrication process. The modeled parameters are extracted empirically. Ten post-gate characteristics are used as the model input: drain–source saturation current, drain–gate, gate–source, and drain–gate resistances, source resistance, drain resistance, pinch-off voltage, transconductance, gate-metal sheet resistance, and gate layer width. The output are eight final DC device characteristics: drain–source saturation current, drain–gate, gate–source, and drain–gate resistances, source resistance, drain resistance, pinch-off voltage, and transconductance.

The modeling of each stage first requires PCA preprocessing and then building an NN, as shown in Fig. 1. The PCA extracts orthogonal principal directions in multidimensional input space in descending order as characterized by corresponding variances. This allows for reduction of the original input data dimension crucial for inverse modeling later on. The PCA provides linear operator matrices for the forward and inverse data transformation. An NN is used in the reduced space to approximate the relationship between input and output characteristics of a modeled stage. After training, the NN approximates a nonlinear vector function, which represents the stage-to-stage process model identification.

A model acquired in this manner can be used for the design centering task. Assuming target values and tolerances for final semiconductor device characteristics at the final stage of the fabrication process, the desired values of the earlier stage parameters can be composed in two steps: first, the value of the intermediate variables at the network input satisfying the output target can be found by inverting the function performed by the trained network. Afterwards, the optimum values of these variables ensuring maximum yield probability

This work has been supported by the ONR Grant N00014-93-1-0855.

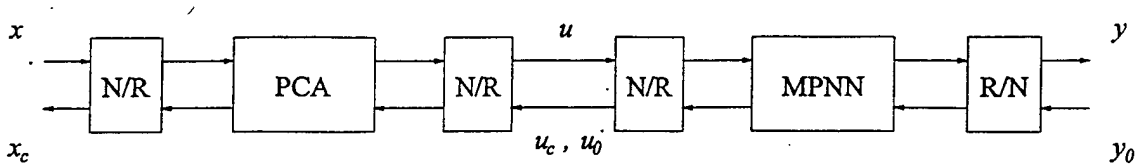


Fig. 1: Microfabrication stage model block diagram. N/R's represent normalization/denormalization steps.

under noncorrelated normal distribution of process variations in the principal directions are estimated. Finally, the center settings for the original input variables are evaluated based on using the inverse PCA operator.

2. Formalization of the design centering problem

The VLSI microfabrication process is described by its input and output characteristics that are captured in available measurement data. Each data point reveals the input/output relationship resulting from the material and the technology. From the viewpoint of analysis, the data points taken at various locations of a wafer are regarded probabilistically as random events and are characterized by the input and output distributions. Thus let x and y be the input and output random variables in the form of an n-vector and m-vector, respectively, with the assumption that there are n input characteristics and m output characteristics for the given stage. The relationship between x and y can be formally expressed as a function μ that maps input characteristics data into the output characteristics data:

$$y = \mu(x) \tag{1}$$

A center value x_c needs to be set at the input in order to achieve a specific, required output (target value) y_0 during the stage fabrication process. However, due to the randomness of the fabrication factors, equipment imperfection and fluctuation of the settings, the actual x is randomly distributed around x_c. When many factors are involved the random spread can be approximated by a Gaussian distribution with mean x_c: $p(x, x_c) = N(x_c, \sigma_x)$. Entries of x are correlated with each other due to the probabilistic dependencies between distributions of individual components in x which is manifested by non-zero off-diagonal entries in the covariance matrix. Moreover, the technology-related spread of characteristics is assumed to be beyond the user control. Only the center input value x_c can be set when targeting the desired output y_0.

The goal of design centering in the fabrication process is to maximize the final product yield by choosing proper settings of the input parameters. The output characteristics are then expected to produce a given target value and fit into the tolerance limits with the highest probability. Assume that the product is acceptable if the target output value y_0 is obtained with tolerance δ_y. Define the target set $\omega_y = \{y : y_{i_{\min}} \leq y_i \leq y_{i_{\max}}\}$ Thus, each entry y_i must belong to the region bounded by $y_{i_{\min}} = (1 - \delta_{y_i})y_0$ and $y_{i_{\max}} = (1 + \delta_{y_i})y_0$. Formally, the process yield can be characterized by probability $\Pr(y \in \omega_y)$. Since this probability is to be maximized and the only input parameter that can be controlled is the center input value x_c, the definition of the design centering task now takes the following form:

$$\max_{x_c} \Pr(y \in \omega_y) \tag{2}$$

Expression (2) provides a functional for optimization. Note that input and output distributions are related through equation (1) and generally $\mu(x_c) \neq y_0$ due to nonlinearity of function μ.

Typically, when creating a model of a stage, many measurements are taken of all relevant process factors or characteristics, but they are related to each other due to their mutual correlations. Since optimization in a multidimensional space is both difficult and time consuming, especially when nonlinear process models are involved, the space size is first reduced. By reducing the input space dimensionality, more efficient algorithms can be used while computational complexity can be reduced to a reasonable level.

The entire fabrication process model consists of two components: PCA and modeling through NN. Relationships can be calculated in both directions, i.e., from input to output and from output to input. The intermediate variable u, referred to as "abstract variable," represents the normalized and compressed space in which the design centering will be performed. In the forward direction the output sample u can be obtained from input data sample x by using the PCA operator that projects x into u followed by the NN mapping $u \rightarrow y$. Given the desired output value y_0, the corresponding variable u_0 (if in existence) can be computed by

an iterative search for solution using the inverse of the NN mapping. Subsequently, the corresponding input x can be found by using the inverse PCA operator.

2.1. Principal Component Analysis

As indicated in Fig. 1, the original input data is transformed into u by the PCA and two normalization stages. The input normalization of x is necessary to unbias the input data and balance their scaling. The PCA changes the input variables representation and reduces dimension from n to m, where $m < n$. Also, the data becomes uncorrelated after the PCA. Afterwards, another normalization equalizes each variable entry variance. The resulting data representation u is a random variable composed of m entries u_i of zero correlation between each other. The input data is characterized by means $\langle x_i \rangle$ and standard deviations σ_{x_i}. Prior to the PCA, normalized input $\hat{x}$ is calculated using the following equation:

$$\hat{x}_i = \frac{x_i - \langle x_i \rangle}{\sigma_{x_i}} \tag{3}$$

The resulting variable $\hat{x}$ has zero mean and unit variance at each entry. Successively, eigenvectors of autocorrelation matrix $R = \langle \hat{x}\hat{x}^T \rangle$ have to be found to obtain the PCA operators. Let v_k be an eigenvector of matrix R and λ_k the corresponding eigenvalue such that they yield equation $Rv_k = \lambda_k v_k$. Additionally, let eigenvectors v_k be orthonormal so the norm $v_k^T v_k = 1$ for each of the eigenvectors. It is also beneficial to use a descending order of eigenvalues, such that $\lambda_k \geq \lambda_{k+1}$. Eigenvectors v_k span a new basis for the input data representation. A PCA operator matrix M can be created to transform input $\hat{x}$ into its projection $\hat{u}$ in the new basis. Grouping first m eigenvectors of the largest eigenvalues in matrix $M = [v_k]^T$ with $k = 1, \ldots, m$ creates a rectangular $m \times n$ matrix with property $MM^T = I$. This matrix serves as the PCA operator which transforms input $\hat{x}$ into vector $\hat{u}$:

$$\hat{u} = M\hat{x} \tag{4}$$

The new data points $\hat{u}$ belong to an m-dimensional space which is reduced as compared to the original input space dimension. But in addition, data points $\hat{u}$ are now uncorrelated, which can be expressed by $\langle \hat{u}\hat{u}^T \rangle = \Lambda$, where Λ is a diagonal matrix with entries λ_k, $k = 1, \ldots, m$ on the diagonal. In other words $\langle \hat{u}_k \hat{u}_l^T \rangle = \lambda_k$ if $k = l$ and $\langle \hat{u}_k \hat{u}_l^T \rangle = 0$ if $k \neq l$. This means that $\hat{u}$ belongs to the m-dimensional distribution and λ_k is a variance of the k-th entry in this distribution. Note that λ_k is also a variance of the data points $\hat{x}$ projected onto the direction of eigenvector v_k which represents the k-th principal direction of the input data distribution. Since entries $\hat{u}_k$ have different variances another normalization step can simplify the data analysis. Denote the normalized data points by u. The normalization in this step is simple and reads:

$$u_k = \frac{1}{\sqrt{\lambda_k}} \hat{u}_k \tag{5}$$

The new data representation has the property $\langle uu^T \rangle = I$ suitable for design centering algorithms. Each point u can be inversely transformed to the original input space with the inverse PCA transformation operator

$$B = M^T \tag{6}$$

Due to the dimension reduction performed by operator M, point $\hat{x}$ and point obtained through inversion $BM\hat{x}$ are not identical if $m < n$. The data representation error is $e = \hat{x} - BM\hat{x}$. It may be shown [3] that the average squared error $||e||^2$ equals the sum of all eigenvalues associated with the eigenvectors not included in the PCA operator matrix M. Error norm $||e||^2$ can be used in choosing the appropriate dimension m of the data points u space.

2.2. Inverse projection through neural model

Mapping $x \rightarrow y$ representing the process is generally a continuous nonlinear function. The PCA part of the entire model is a linear transformation. Hence a function approximator has to be used to complete the task of modeling the process. An NN [4] is proposed for this purpose. Additional normalization and denormalization

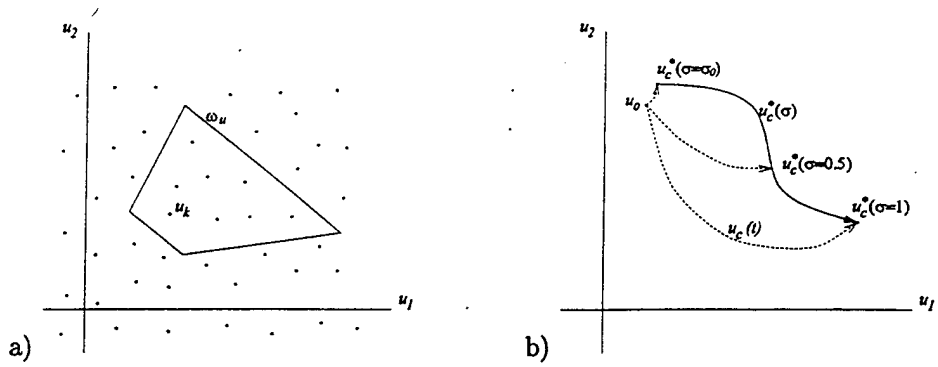


Fig. 2: (a) Approximating the yield probability, (b) Movement of the solution with respect to σ.

steps need to be done at the network input and output to enable the network to learn the stage characteristics. Classic error backpropagation training has been found sufficient to train the neural network. For the sake of finding an input u_0 yielding target output y_0 through the neural model, the algorithm introduced in [5] will be used. Define the solution error as a norm $E = ||y - y_0||^2$. The error gradient dE/du will allow for iterative search in the u space for solution to the desired output y_0. The gradient entries read:

$$\frac{\partial E}{\partial u_k} = \sum_i \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial u_k} \tag{7}$$

Then u can be evaluated iteratively according to the steepest descent method: $u_i' = u_i - \kappa\frac{\partial E}{\partial u_i}$ where κ controls the algorithm convergence rate [6].

2.3. Optimization algorithm

The solution to expression (2) should be searched in u-coordinates since they represent orthonormalized space for the input data distribution with reduced dimension. Define region ω_u such that implication $(u \in \omega_u) \Rightarrow (y \in \omega_y)$ is valid. In other words all the points u which belong to region ω_u will result in acceptable output values $y = f(u)$. Note that the output space dimension is greater than m, therefore the inverse implication does not necessarily hold true, but still, $\Pr(y \in \omega_y) = \Pr(u \in \omega_u)$ Since variable u space is orthonormalized, the data points distribution can now be represented by a symmetric m-dimensional Gaussian $p(u, u_c)$ centered at some u_c that will be moved while optimization is performed:

$$p(u, u_c) = N(u_c, \sigma) = \frac{1}{(\sqrt{2\pi}\sigma)^m} e^{-\frac{\|u-u_c\|^2}{2\sigma^2}} \tag{8}$$

Here $\sigma = 1$, and $\|u - u_c\|$ is a norm of a distance between u and the center point u_c. The PCA obviously yields $u_c = 0$, however, the design centering will provide some non-zero value of u_c that will be considered as a solution x_c when transformed back into the input space. Denote the yield probability as p_c which in the u-space can be described by the integral:

$$p_c = \Pr(u \in \omega_u) = \int_{\omega_u} p(u, u_c) du \tag{9}$$

Now assume that the space is uniformly covered by random points u_k as shown in Fig. 2. The points neighborhoods s_k composed together fill the entire space. If the number of points is sufficiently large, probability p_c can be approximated by the sum $p_c = \sum_{k\in\sigma} s_k p(u, u_c)$. The goal of the design centering is equivalent to maximizing probability p_c by moving the center point: $\max_{u_c} p_c$. The solution u_c^* should be found as a result of an optimization algorithm with functional p_c. Define gradient of p_c that will be useful for this algorithm:

$$\frac{dp_c}{du_c} = \sum_{k\in\delta} s_k p(u_k, u_c)(-\frac{1}{2\sigma^2})\frac{d}{du_c}\|u_k - u_c\|^2 \tag{10}$$

Gradient (10) indicates the direction toward which the center point should be moved in order to increase the yield probability p_c. Assume that the optimization algorithm is used in the neighborhood of the global solution at this stage. The following simple gradient-based optimization algorithm is proposed:

$$\frac{du_c}{dt} = \frac{dp_c}{du_c}, \qquad u_c(0) = u_0 \tag{11}$$

Regarding now u_c as time variable $u_c = u_c(t)$ with initial condition u_0, the differential equation has a fixed point at u_c^*. As long as the initial condition is in the neighborhood of the global solution, the proposed algorithm will generate trajectory $u_c(t)$ that leads from u_0 to u_c^*. Intuitively, choosing u_0 such that $f(u_0) = y_0$ brings u_c close to u_c^*. This holds when f is linear, and is sufficient for nonlinear f with properties of smoothness and monotonicity resulting from technology and chemical processes. The need to evaluate terms at every point k with the algorithm described by (11) is a distinct disadvantage. Although dimensionality of the u-space is reduced due to PCA, the algorithm can still be computationally inefficient. The efficiency can be improved by the following redefinition:

$$\frac{du_c}{dt} = \sum_{k \notin \delta} s_k p(u_k, u_c) \frac{1}{2\sigma^2} \frac{d}{du_c} \|u_k - u_c\|^2 \tag{12}$$

By now σ was treated as a unity constant. However, during the optimization σ can be slowly varied, the result u_c^* will be the same provided that the final value of σ is 1. Let σ be a parameter which will be slowly changed from some small initial value σ_0, up to 1 at the end of optimization. Perturbing σ will affect the solution u_c^* which now becomes a function $u_c^* = u_c^*(\sigma)$. Parameter σ can be used to control the number of points affecting the location of the final solution.

3. Numerical results of the algorithm

The data come from measurements taken on a 4×4.5mm high density structure reticle repeated some 200 times per wafer. Process and device characteristics were measured at a sufficient density to fully characterize variations across the wafer [7]. A horizontal slice of 14 reticles across the middle of the wafer was chosen for modeling purposes. This provided 69 data sets that allowed for examination of the most crucial variations and the effect they have on MMIC performance. Prior to the design centering the fabrication stage model has to be approximated based on the input–output characteristics. The general model shown in Fig. 1 requires the PCA of the input characteristics. Using the collected measurement data related to the input, the normalized vector $\hat{x}$ is first obtained from equation (3). Successively, eigenvalues λ_k of the normalized data autocorrelation matrix R are calculated. Choosing abstract space dimension $m = 2$, the PCA operator matrix M is then built of two eigenvectors associated with the largest two eigenvalues. Afterwards, new data points u are calculated following equations (4) and (5).

To complete the model from Fig. 1 two layer feedforward NN with 2 inputs, 6 hidden units, and 8 outputs is trained using points u as the input training set and corresponding final characteristics measurements y as the output training set. Out of the 69 data sets the first 50 are used for training and the remaining 19 serve as the testing set. The NN model will be later used to inversely calculate point u_0 corresponding to target output y_0. Therefore, it is important to inspect the abstract space region in which the data points u are distributed. A solution u_0 not belonging to the distribution of u should be considered as unacceptable and the corresponding target y_0 treated as unavailable within the constructed neural model. At this stage the entire fabrication process model is developed. Using the model output characteristics y can be calculated based on any input x that belongs to the identified input characteristics' distribution. Conversely, by using the neural mapping inversion and inverse PCA operator, an input x for any output sample y can be found when y belongs to the output characteristics' distribution. The internal abstract representation u of both inputs and outputs is available for design centering tasks.

The goal of the following numerical illustration is to inspect and then maximize a simulated fabrication yield when the MESFET target characteristics y_0 are required with tolerance δ. Three tolerance δ values: 5%, 10%, and 15% are considered. By using the simple inversion–based approach and equations (5), (6), and (3), values of u_0 and x_0 can be found. Fabrication yield is then estimated assuming that input characteristics are attempted around the inverse solution x_0. Due to variations resulting from technology some of the fabricated

Table 1: Fabrication yield for inverse solution and centered solution with various allowed tolerances.

tolerance δ	inverse x_0	centered x_c
5%	7.2%	7.5%
10%	27.1%	27.6%
15%	52.1%	55.8%

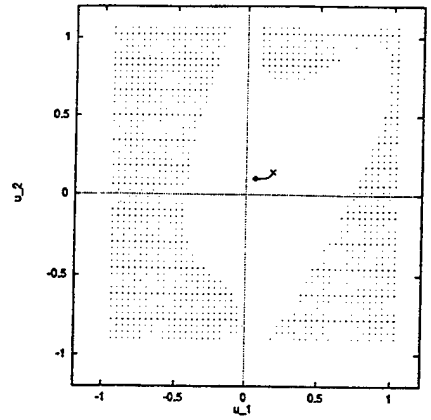


Fig. 3: Design centering in MESFET gate-final fabrication stage. Diamond represents an inverse u_0 to target y_0 in the abstract coordinates u_1–u_2. Centered value u_c, indicated by the cross, allows for the yield maximization, within tolerance δ equal 15%.

devices have final characteristics y off the required tolerance δ which lowers the yield. To estimate the yield 1000 random input points distributed around x_0 with the original input data distribution variances are tested for tolerance requirements. The middle column in Table 1 contains the yield as a percentage number of points meeting the criteria.

The approach introduced in the previous chapter can be employed to improve the yield. The inverse solution u_0 is regarded as an initial condition to the optimization algorithm described by equation (12). Points neighboring u_0 are inspected for tolerance criterion after mapping to the output. Neighborhood of u_0 is shown in Fig. 3 for 15% tolerance δ. Points u which fail to fall into the tolerance region after mapping to the output are marked with dots in these figures. Thus the blank area surrounded by dotted boundaries is the projection of the output tolerance region into the abstract space. Starting with the initial u_0, the optimization algorithm drags the center point to another location u_c, which is considered the centered solution to the yield maximization task. Afterwards, the centered point u_c is transformed into the original input space and results with desired centered input x_c. The yield for these new centered solutions is then estimated in the same manner as for x_0. The improved yield is shown in the right column in Table 1. As indicated in the table the design centering improves the yield, especially when large tolerance for the target y_0 is required.

4. Conclusions

The presented design centering approach allows for yield maximization in fabrication processes without major changes to technology and available means. The yield can be significantly improved when nonlinear relationships are involved in the process characterization. This is the case in GaAs microelectronic devices manufacturing. The design centering algorithm can efficiently work even with a large amount of measurement data since Principal Component Analysis of the data reduces the problem size and the "curse of dimensionality" is avoided.

References

[1] G. L. Creech, J. M. Zurada, and P. B. Aronhime, "Feedforward neural networks for estimating ic parametric yield and device characterization," in *Proc. IEEE Int. Symposium on Circuit and Systems*, vol. 2, (Seattle, WA), pp. 1520–1523, Apr. 30–May 3, 1995.

[2] J. C. Zhang and M. A. Styblinski, *Yield and Variability Optimization of Integrated Circuits.* Kluwer Academic Publishers, 1995.

[3] A. Cichocki and R. Unbehauen, *Neural Network for Optimization and Signal Processing.* John Wiley & Sons, 1993.

[4] J. M. Zurada, *Introduction to Artificial Neural Systems.* Boston: PWS, 1992.

[5] D. A. Hoskins, J. N. Hwang, and J. Vagners, "Iterative inversion of neural networks and its application to adaptive control," *IEEE Transactions on Neural Networks*, vol. 3, pp. 292–301, March 1992.

[6] A. Malinowski, *Reduced Perceptron Networks and Inverse Mapping Control.* PhD thesis, University of Louisville, Louisville, KY, June 1996.

[7] G. L. Creech and J. M. Zurada, "GaAs MESFET DC characteristics and process modeling using neural networks," in *Proc. of the Artificial Neural Networks in Engineering*, (St. Louis, MI), Nov. 13–16, 1994.

1997 IEEE
AEROSPACE CONFERENCE
February 1-8, 1997 Snowmass, Colorado

Design Centering in GaAs IC Manufacturing

Jacek M. Zurada, Andrzej Lozowski, Aleksander Malinowski
Department of Electrical Engineering
University of Louisville
Louisville, KY 40292
502-852-6314
jmzura02@starbase.spd.louisville.edu

Abstract—This paper describes a practical method of design centering for microelectronic circuits fabrication process. Process data are first evaluated for principal components and subsequently modeled using multilayer perceptron networks in a reduced and transformed input space. Perceptron network models are then inverted, and center settings of input variables are computed by using the inverse PCA transformation. The approach allows for maximizing the yield of fabricated GaAs circuits used in aviation electronics systems. Example of yield maximization for MMIC fabrication data is provided to illustrate the proposed technique.

TABLE OF CONTENTS

1. INTRODUCTION.
2. FORMALIZATION OF THE DESIGN CENTERING PROBLEM.
3. THE APPROACH.
4. APPLICATION TO YIELD OPTIMIZATION OF GATE-FINAL FABRICATION STAGE.
5. CONCLUSIONS.

1. INTRODUCTION

The majority of the development cost for many military systems lies in the design and fabrication of the microelectronic integrated circuits (IC). In order to achieve acceptable fabrication yield, the integrated circuits need to meet certain difficult system specifications involving complexity and frequency requirements [1].

The goal of this work is to maximize the fabrication yield of Gallium Arsenide (GaAs) Microwave/Millimeter Wave Monolithic Integrated Circuits (MMIC) with respect to the material, process, and device parameters, while achieving the best possible circuit performance. The techniques developed in the present research are applicable to GaAs IC technology and are also valid for other fabrication technologies, such as CMOS or BiCMOS technology.

Stages of the microelectronic circuit fabrication process can be efficiently modeled with multilayer perceptron neural networks (NN) supported by Principal Component Analysis (PCA) of the underlying data. These specific tools are found to be useful for capturing the relationships between various stages in the manufacturing process as well as between the process parameters and the resulting device parameters. Once the model is identified, a practical degree of design centering can be achieved by inverse modeling. In practice, the design centering problem requires the solution of the desired values of early manufacturing parameters (or process attributes) given the target performance of the final product.

The first step in the design centering is the

This work has been supported by the ONR Grant N00014-93-1-0855

fabrication process model identification [2]. The following critical stages of the GaAs IC fabrication process were selected for modeling [3]: substrate/active layer (S), post-contact/recess (CR), post-gate-metal (G), and final (F). The measurement data distribution for the S process stage consists of ten substrate characteristics: two optical scatterings, Neut deep donor density, substrate resistivity, Hall mobility and carrier concentration, doping concentration, implant activation, drift mobility I, and drift mobility II.

Measurements for stage CR include: drain-source saturation currents and resistances (both contact and recess), contact resistance, contact and ohmic metal sheet resistance and ohmic metal layer width. G and F stage characteristics are the MESFET DC parameters: drain, gate, source, drain-source, drain-gate and gate-source resistances, drain-source saturation current, pinch-off voltage and device transconductance. Also, gate metal sheet resistance and gate metal width are included in the G stage measurements.

The modeling of each stage requires first PCA preprocessing and then building a neural network, as shown in Fig. 1. The PCA extracts orthogonal principal directions in multidimensional input space in descending order as characterized by corresponding eigenvalues (variances). This allows for reduction of the original input data dimension crucial for inverse modeling later on. The PCA provides matrices M and B which are the forward (compressing) and inverse (expanding) linear operators, respectively. Preliminary calculations indicate that the characteristics describing the consecutive fabrication stages are mutually correlated between each other. The data distribution for stages S (10 variables), CR (8 variables) and G (10 variables) can be reduced to 6, 5 and 7 abstract variables, respectively, with normalized estimation error better than 1% (after compression and expansion).

A multilayer perceptron neural network is used following the dimension reduction to approximate the relationship between input and output characteristics of a modeled stage. After training the NN performs nonlinear vector function f which represents the stage to stage process model identification.

The model acquired in this manner can be used for the design centering task. Assuming target values and tolerances for final semiconductor device characteristics at stage F of the fabrication process, the desired values of earlier stages S, CR or G parameters can be obtained in two steps: first the value of the abstract variables at the network input satisfying the output target can be found by inverting the function performed by the trained network. Afterwards, the optimum values of these variables ensuring maximum yield probability under noncorrelated normal distribution of process variations in the principal directions are estimated. Finally, the center settings for the original input variables are evaluated based on using inverse PCA operator B.

2. Formalization of the Design Centering Problem

The VLSI microfabrication process is described by its input and output characteristics that are captured in available measurement data. This allows for fabrication process identification. Each data point reveals the input/output relationship resulting from the material and the technology. From the viewpoint of analysis, the data points taken at various locations of a wafer are regarded probabilistically as random events and are characterized by the input and output distributions. Thus let $\boldsymbol{x}$ and $\boldsymbol{y}$ be the input and output random variables in the form of an n-vector and m-vector, respectively, with

the assumption that there are n input characteristics and m output characteristics for the given stage.

The relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$ can be formally expressed as a function μ that maps input characteristics data into the output characteristics data:

$$\boldsymbol{y} = \mu(\boldsymbol{x}) \tag{1}$$

A center value $\boldsymbol{x}_c$ is to be maintained at the input in order to achieve a specific, required output (target value) $\boldsymbol{y}_0$ during the stage fabrication process. However, due to the randomness of the fabrication factors, equipment imperfection and fluctuation of the settings, the actual $\boldsymbol{x}$ is typically randomly distributed around $\boldsymbol{x}_c$. When many factors are involved and many fabrication cases considered, the random spread can be approximated by a Gaussian distribution with mean $\boldsymbol{x}_c$ and covariance matrix σ_x. The input distribution thus reads

$$p(\boldsymbol{x}, \boldsymbol{x}_c) = N(\boldsymbol{x}_c, \sigma_x) \tag{2}$$

Here, $p(\boldsymbol{x}, \boldsymbol{x}_c)$ represents the actual distribution of input values $\boldsymbol{x}$ when maintaining the center value $\boldsymbol{x}_c$ is attempted. Entries in vector $\boldsymbol{x}$ are expected to correlate with each other due to probabilistic dependencies between distributions of individual components in $\boldsymbol{x}$ is manifested by non-zero off-diagonal entries in covariance matrix σ_x. Moreover, the technology related spread of characteristics is assumed to be beyond control. Only the center input value $\boldsymbol{x}_c$ can be set when targeting at desired output $\boldsymbol{y}_0$.

The goal of design centering in the fabrication process is to maximize the final product yield by choosing proper settings of the input parameters. The output characteristics are then expected to produce a given target value and fit into the tolerance limits. Assume that the product is acceptable if the target output value $\boldsymbol{y}_0$ is obtained with tolerance δ_y. Define the target set ω_y as follows:

$$\omega_y = \{\boldsymbol{y} : y_{i_{\min}} \leq y_i \leq y_{i_{\max}}\} \tag{3}$$

Thus, each entry y_i must belong to the region bounded by $y_{i_{\min}} = (1 - \delta_{y_i})y_0$ and $y_{i_{\max}} = (1 + \delta_{y_i})y_0$. Formally, the process yield can be characterized by probability $\Pr(\boldsymbol{y} \in \omega_y)$. Since this probability is to be maximized and the only input parameter that can be controlled is the center input value $\boldsymbol{x}_c$, the definition of the design centering task now takes the following form:

$$\max_{\boldsymbol{x}_c} \Pr(\boldsymbol{y} \in \omega_y) \tag{4}$$

Equation (4) provides a functional for optimization. Maximizing this functional is equivalent to maximizing the process yield. Note that input and output distributions are related through equation (1) and generally $\mu(\boldsymbol{x}_c) \neq \boldsymbol{y}_0$ due to nonlinearity of function μ.

3. The Approach

Typically, when creating a model of a stage many measurements are taken of all relevant process factors or characteristics. Most of the inspected characteristics are related to each other due to their mutual correlations. The design centering is an optimization process and will involve all of these input factors. Since optimization in a multidimensional space is both difficult and time consuming, especially when nonlinear process models are involved, the space size is first reduced. By reducing the input space dimensionality, more efficient algorithms can be used while computational complexity can be reduced to a reasonable level.

The entire fabrication process model consists of two components: PCA and the neural network modeling through MPNN. Relationships can be calculated in both directions, i.e., from the input to the output and from the output to the input. Thus four operations are required to handle the data. The

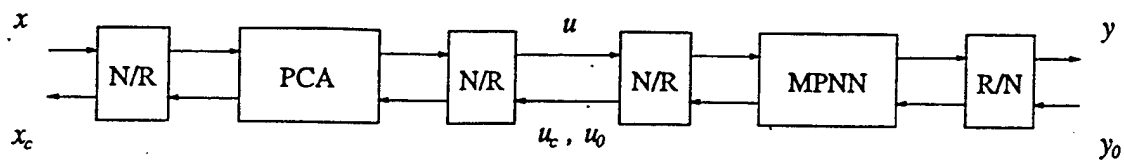


Fig. 1. Microfabrication stage model block diagram.

intermediate variable $\boldsymbol{u}$, refered to as "abstract variable" represents normalized and compressed space, in which the design centering will be performed.

In the forward direction the output sample $\boldsymbol{u}$ can be obtained from input data sample $\boldsymbol{x}$ by using the PCA operator that projects $\boldsymbol{x}$ into $\boldsymbol{u}$ and then the neural network mapping $\boldsymbol{u} \rightarrow \boldsymbol{y}$. Given the desired output value $\boldsymbol{y}_0$, the corresponding variable $\boldsymbol{u}_0$ (if in existence) can be computed by an iterative search for solution using the inverse of the neural network mapping. Subsequently, the corresponding input $\boldsymbol{x}$ can be found by using inverse PCA operator.

Principal Component Analysis

The Principal Component Analysis (PCA) of the measurements of input characteristics is used in order to reduce the model input dimensionality. As indicated in Fig. 1, the original input data is transformed into $\boldsymbol{u}$ by the PCA and two normalization stages. The input normalization of $\boldsymbol{x}$ is necessary to unbias the input data and balance their scaling. The PCA changes the input variables representation and reduces dimension from n to m, where $m < n$. Also, the data becomes uncorrelated after the PCA. Afterwards, another normalization equalizes each variable entry variance. The resulting data representation, referred to as $\boldsymbol{u}$, is a random variable composed of m entries u_i of zero correlation between each other. This property will substantially simplify the design centering described later on. The following is the description of variable transformation $\boldsymbol{x} \rightarrow \boldsymbol{u}$.

The input data is characterized by means $\langle x_i \rangle$ and standard deviations σ_{x_i}. Prior to the PCA, normalized input $\hat{\boldsymbol{x}}$ is calculated using the following equation:

$$\hat{x}_i = \frac{x_i - \langle x_i \rangle}{\sigma_{x_i}} \tag{5}$$

The resulting variable $\hat{\boldsymbol{x}}$ has zero mean and unit variance at each entry Successively, autocorrelation matrix R is calculated as follows:

$$R = \langle \hat{\boldsymbol{x}} \hat{\boldsymbol{x}}^T \rangle \tag{6}$$

In order to obtain a PCA operator, eigenvectors of matrix R have to first be found. Let $\boldsymbol{v}_k$ be an eigenvector of matrix R and λ_k the corresponding eigenvalue, such that they yield the equation:

$$R\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k \tag{7}$$

Additionally, let eigenvectors $\boldsymbol{v}_k$ be orthonormal so the norm $\boldsymbol{v}_k^T \boldsymbol{v}_k = 1$ for each of the eigenvectors. It is also beneficial to use a descending order of eigenvalues, such that $\lambda_k \geq \lambda_{k+1}$.

Eigenvectors $\boldsymbol{v}_k$ span a new basis for the input data representation. A PCA operator matrix M can be created to transform input $\hat{\boldsymbol{x}}$ into its projection $\hat{\boldsymbol{u}}$ in the new basis. Grouping first m eigenvectors of the largest eigenvalues in matrix M:

$$M = [\boldsymbol{v}_k]^T \qquad k = 1, \ldots, m \tag{8}$$

creates a rectangular $m \times n$ matrix having property $MM^T = I$. This matrix serves as the PCA operator which transforms input $\hat{\boldsymbol{x}}$ into vector $\hat{\boldsymbol{u}}$:

$$\hat{\boldsymbol{u}} = M\hat{\boldsymbol{x}} \tag{9}$$

The new datapoints $\hat{u}$ belong to an m-dimensional space which is reduced as compared to the original input space dimension. But in addition, data points $\hat{u}$ are now uncorrelated, which can be expressed by $\langle \hat{u}\hat{u}^T \rangle = \Lambda$, where Λ is a diagonal matrix with entries λ_k, $k = 1, \ldots, m$ on the diagonal. In other words $\langle \hat{u}_k \hat{u}_l^T \rangle = \lambda_k$ if $k = l$ and $\langle \hat{u}_k \hat{u}_l^T \rangle = 0$ if $k \neq l$. This means that $\hat{u}$ belongs to the m-dimensional distribution and λ_k is a variance of the k-th entry in this distribution. Note that λ_k is also a variance of the datapoints $\hat{x}$ projected onto the direction of eigenvector v_k which represents the k-th principal direction of the input data distribution.

Since entries $\hat{u}_k$ have different variances another normalization step can simplify the data analysis. Denote the normalized datapoints by u. The normalization in this step is simple and reads:

$$u_k = \frac{1}{\sqrt{\lambda_k}} \hat{u}_k \tag{10}$$

Finally, by following steps expressed by equations (5), (9), and (10) each of input data point x can be transformed into point u in the new, reduced space. The new data representation has the property $\langle uu^T \rangle = I$ making it suitable for design centering algorithms. Each point u can be inversely transformed to the original input space with a controlled degree of accuracy depending on dimension m. Let B be the inverse PCA transformation operator

$$B = M^T \tag{11}$$

Due to the dimension reduction performed by operator M, point $\hat{x}$ and inversely obtained point $BM\hat{x}$ are not identical if $m < n$.

Define an error of data representation in the reduced space related to the model input as a difference between the original point $\hat{x}$ and its representation

$$e = \hat{x} - BM\hat{x} \tag{12}$$

It may be shown [4] that the average squared error $||e||^2$ equals the sum of all eigenvalues associated with the eigenvectors not included in the PCA operator matrix M:

$$||e||^2 = \langle e^T e \rangle = \sum_{k=m+1}^{n} \lambda_k \tag{13}$$

Error norm $||e||^2$ can be used in computing the dimension m of the data points u space. Note that this error is related to the PCA only and is a part of the error of the entire fabrication process model.

Inverse projection through neural model

Mapping $x \rightarrow y$ representing the process is generally a continuous nonlinear function. The PCA part of the entire model is a linear transformation. Hence a function approximator has to be used to complete the task of modeling the fabrication process. An MPNN [5] is proposed for this purpose. Additional normalization and renormalization steps need to be done at the network input and output to enable the network learn the stage characteristics. Classic error backpropagation training is sufficient to train the neural network.

For the sake of finding an input u_0 given target output y_0 through the neural model the algorithm, introduced in [6] will be used. Define solution error E as a norm:

$$E = ||y - y_0||^2 \tag{14}$$

The error gradient dE/du will allow for iterative search in the u space for solution to the desired output y_0. The gradient entries read:

$$\frac{\partial E}{\partial u_k} = \sum_i \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial u_k} \tag{15}$$

Then u can be evaluated iteratively according to the steepest descent method:

$$u_i' = u_i - \kappa \frac{\partial E}{\partial u_i} \tag{16}$$

where κ controls the algorithm convergence rate [7].

Optimization algorithm

The solution to expression (4) should be searched in u-coordinates since they represent orthonormalized space for the input data distribution with reduced dimension. Region ω_y represents all acceptable output variable values resulting from the tolerance and target point requirements. Define region ω_u such that implication $(u \in \omega_u) \Rightarrow (y \in \omega_y)$ is valid. In other words all the points u which belong to region ω_u will result in acceptable output values $y = f(u)$. Note that the output space dimension is greater than m, therefore the inverse implication does not neccessarily hold true. Nevertheless the following probabilities are equal:

$$\Pr(y \in \omega_y) = \Pr(u \in \omega_u) \tag{17}$$

Since variable u space is orthonormalized, the data points distribution can now be represented by a symmetric m-dimensional Gaussian $p(u, u_c)$ centered at some u_c that will be moved while optimization:

$$p(u, u_c) = N(u_c, \sigma) = \frac{1}{(\sqrt{2\pi}\sigma)^m} e^{-\frac{\|u-u_c\|^2}{2\sigma^2}} \tag{18}$$

Here σ equals 1 and is used for further purposes, and $\|u - u_c\|$ is a norm of a distance between variable u value and the center point u_c. The PCA obviously yields $u_c = 0$, however, the design centering will provide some non-zero value of u_c that will be considered as a solution x_c when transformed back into the input space.

Denote the yield probability as p_c. In the

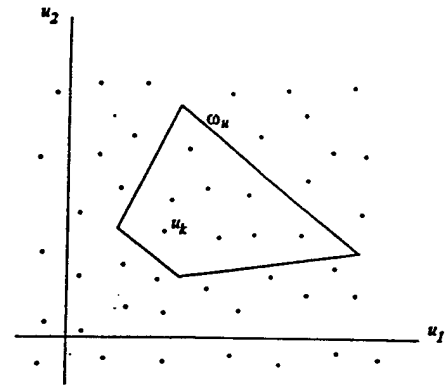


Fig. 2. Approximating the yield probability.

u-space it can be described by the integral:

$$p_c = \Pr(u \in \omega_u) = \int_{\omega_u} p(u, u_c) du \tag{19}$$

Now assume that the space is uniformly covered by random points u_k, as shown in Fig. 2. The points neighborhoods s_k composed together fill the entire space. The points belonging to region ω_u create set δ such that the volume of ω_u equals $V_{\omega_u} = \sum_{k\in\delta} s_k$. If the number of points is sufficiently large, probability p_c can be approximated by the following sum:

$$p_c = \sum_{k\in\sigma} s_k p(u, u_c) \tag{20}$$

The goal of the design centering is equivalent to maximizing probability p_c by moving the center point u_c:

$$\max_{u_c} p_c \tag{21}$$

The solution to (21) is a point u_c^* that should be found as a result of an optimization algorithm with functional p_c. Define gradient of p_c that will be useful for this algorithm:

$$\frac{dp_c}{du_c} = \sum_{k\in\delta} s_k p(u_k, u_c)(-\frac{1}{2\sigma^2})\frac{d}{du_c}\|u_k - u_c\|^2 \tag{22}$$

Gradient (22) indicates the direction toward which the center point should be moved in order to increase the yield probability p_c. At

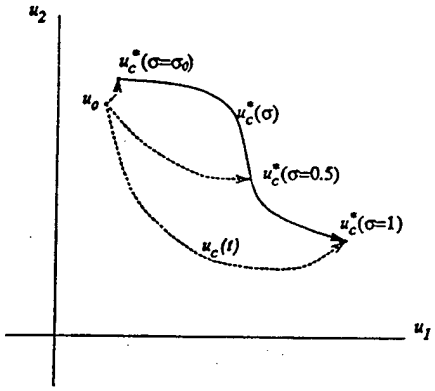


Fig. 3. Movement of the solution with respect to σ.

the solution $\boldsymbol{u}_c^*$ the gradient is zero:

$$\left.\frac{dp_c}{d\boldsymbol{u}_c}\right|_{\boldsymbol{u}_c=\boldsymbol{u}_c^*} = \vec{0} \tag{23}$$

Generally, this gradient can be zero at more than one point, however, not each of these points represents the global solution of maximum probability p_c. Assume that the optimization algorithm is used in the neighborhood of the global solution at this stage. The following simple gradient-based optimization algorithm is proposed:

$$\frac{d\boldsymbol{u}_c}{dt} = \frac{dp_c}{d\boldsymbol{u}_c}, \qquad \boldsymbol{u}_c(0) = \boldsymbol{u}_0 \tag{24}$$

Regarding now $\boldsymbol{u}_c$ as time variable $\boldsymbol{u}_c = \boldsymbol{u}_c(t)$ with initial condition $\boldsymbol{u}_0$, the differential equation has a fixed point at $\boldsymbol{u}_c^*$ satisfying equation (23). As long as the initial condition is in the neighborhood of the global solution, the proposed algorithm will generate trajectory $\boldsymbol{u}_c(t)$ that leads from $\boldsymbol{u}_0$ to $\boldsymbol{u}_c^*$. Refer to Fig. 3 for explanation of the fixed point concept. Intuitively, choosing $\boldsymbol{u}_0$ such that $f(\boldsymbol{u}_0) = \boldsymbol{y}_0$ brings $\boldsymbol{u}_c$ close to $\boldsymbol{u}_c^*$. This would work perfectly if f was linear, but is sufficient for nonlinear f with properties of smoothness and monotonicity resulting from technology and chemical processes.

The need to evaluate terms at every point k in the algorithm described by (24) is a distinct disadvantage. Although dimensionality of the u-space is reduced due to PCA, the algorithm can still be computationally inefficient. The efficiency can be improved by the following redefinition. Note that $\frac{dp_c}{dt} = -\frac{d}{dt}(1-p_c)$. Probability $(1-p_c)$ represents points that miss the target region and can be used for the algorithm as well:

$$\frac{d\boldsymbol{u}_c}{dt} = \sum_{k\notin\delta} s_k p(\boldsymbol{u}_k, \boldsymbol{u}_c)\frac{1}{2\sigma^2}\frac{d}{d\boldsymbol{u}_c}\|\boldsymbol{u}_k - \boldsymbol{u}_c\|^2 \tag{25}$$

By now σ was treated as a unity constant. However, while the optimization σ can be slowly varied, the result $\boldsymbol{u}_c^*$ will be the same provided that the final value of σ is 1. Let σ be a parameter which will be slowly changed from some small initial value σ_0, up to 1 at the end of optimization. Perturbing σ will affect the solution $\boldsymbol{u}_c^*$ location which now becomes a function of σ:

$$\boldsymbol{u}_c^* = \boldsymbol{u}_c^*(\sigma) \tag{26}$$

Parameter σ can be used to control the number of points affecting the solution location.

4. Application to Yield Optimization of Gate-Final Fabrication Stage

The design centering approach introduced in this work is employed to improve the gate-final stage yield of GaAs 0.5mm×200μm MESFET fabrication process. The modeled parameters are extracted empirically. Ten post-gate characteristics are used as the model input $\boldsymbol{x}$:

- G-Idss, drain–source sat. current (mA/mm)
- G-Rds, drain–gate resistance (Ω·mm)
- G-Rgs, gate–source resistance (Ω·mm)
- G-Rs, source resistance (Ω·mm)
- G-Rdg, drain–gate resistance (Ω·mm)
- G-Rd, drain resistance (Ω·mm)
- G-Vpo, pinch-off voltage (V)
- G-Gm, transconductance (mS/mm)

TABLE I

EIGENVALUES OF NORMALIZED INPUT MEASUREMENT POINTS $\hat{x}$ AUTOCORRELATION MATRIX.

k	λ_k
1	5.22504
2	1.8639
3	1.28745
4	0.820715
5	0.54935
6	0.103034
7	0.0740204
8	0.059971
9	0.00934686
10	0.00717611

- G-Rsh, gate-metal sheet resistance (Ω·mm)
- G-Wg, gate layer width (μm)

The output y consists of eight final DC device characteristics:

- F-Idss, drain–source sat. current (mA/mm)
- F-Rds, drain–gate resistance (Ω·mm)
- F-Rgs, gate–source resistance (Ω·mm)
- F-Rs, source resistance (Ω·mm)
- F-Rdg, drain–gate resistance (Ω·mm)
- F-Rd, drain resistance (Ω·mm)
- F-Vpo, pinch-off voltage (V)
- F-Gm, transconductance (mS/mm)

The data come from measurements taken on a 4×4.5mm high density structure reticle repeated some 200 times per wafer. Process and device characteristics were measured at a sufficient density to fully characterize variations across the wafer [3]. A horizontal slice of 14 reticles across the middle of the wafer was chosen for modeling purposes. These reticles were chosen since they contained the only available properly formatted substrate and active layer characteristics. This provided 69 data sets that allowed for examination of the most crucial variations and the effect they have on MMIC performance.

Prior to the design centering the fabrication stage model has to be obtained based on the input–output characteristics. The general model shown in Fig. 1 requires the PCA of the input characteristics. Using the collected measurement data related to the input, the normalized vector $\hat{x}$ is first obtained from equation (5). Successively, eigenvalues λ_k of the normalized data autocorrelation matrix R are calculated using equation (7). The eigenvalues are shown in Table I.

Choosing abstract space dimension $m = 2$, 3×10 PCA operator matrix M is then built of two eigenvectors associated with the largest two eigenvalues. Afterwards, new data points $\boldsymbol{u}$ are calculated following equations (9) and (10).

To complete the model from Fig. 1 two layer feedforward neural network with 2 inputs, 22 hidden units, and 8 outputs is trained using points $\boldsymbol{u}$ as the input training set and corresponding final characteristics measurements $\boldsymbol{y}$ as the output training set. Out of the 69 data sets the first 50 are used for training and the remaining 19 serve as the testing set. The neural network model will be later used to inversely calculate point $\boldsymbol{u}_0$ corresponding to target output $\boldsymbol{y}_0$. Therefore, inspecting the abstract space region in which the data points $\boldsymbol{u}$ are distributed is of importance. A solution $\boldsymbol{u}_0$ not belonging to the distribution of $\boldsymbol{u}$ should be considered as unacceptable and the corresponding target $\boldsymbol{y}_0$ treated as unavailable within the constructed neural model.

The distribution of $\boldsymbol{u}$ and the distribution of the output points $\boldsymbol{y}$ projected into the abstract space in the same manner as the input are shown in Fig. 4a and 4b separately for the training and testing pairs $\boldsymbol{u} \rightarrow \boldsymbol{y}$. Sharp ends of the arrows represent points $\boldsymbol{u}$ whereas the heads of the arrows indicate projected outputs. Arrows themselves represent mapping that is to be performed by the net-

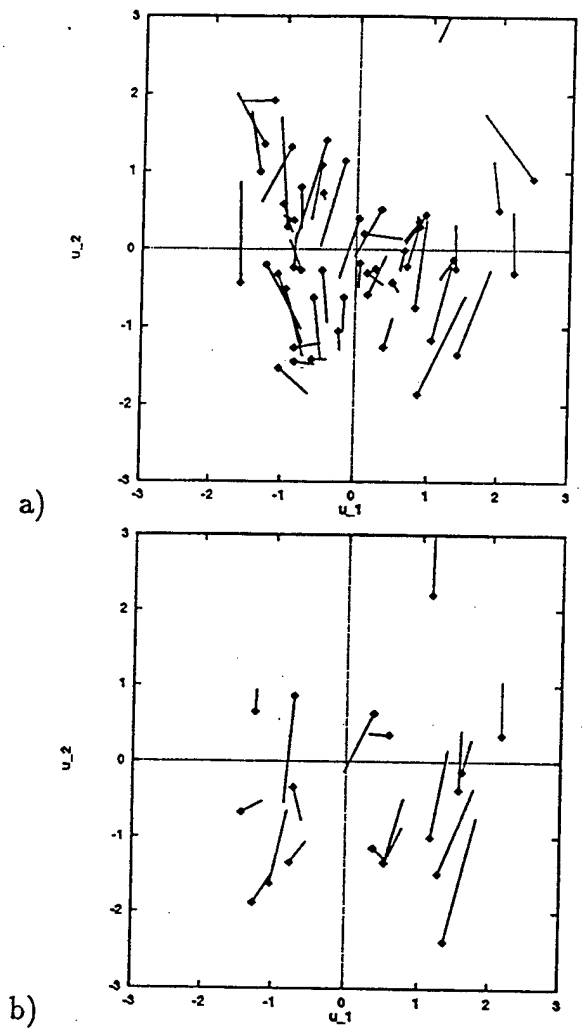


Fig. 4. Representation of the neural required mapping in the abstract space. Arrows start at data points $\boldsymbol{u}$ and end at projections of the output data points $\boldsymbol{y}$. (a) Training pairs (b) testing pairs.

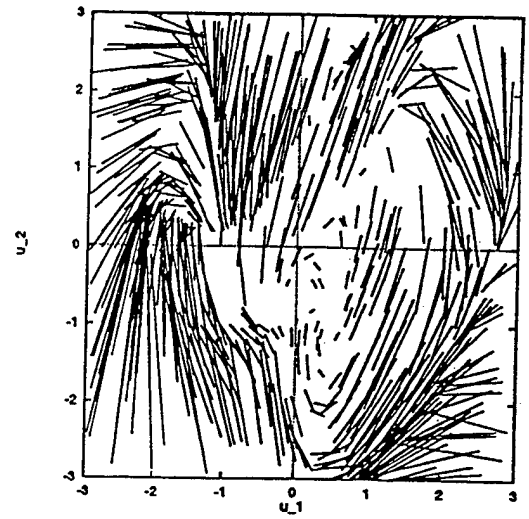


Fig. 5. Actual mapping performed by the trained neural network sampled at random points of the abstract space.

TABLE II
TARGET MESFET CHARACTERISTICS

Characteristics	Target value
F-Idss	224.0
F-Rds	2.674
F-Rgs	3.514
F-Rs	0.8926
F-Rdg	3.678
F-Rd	1.053
F-Vpo	-1.495
F-Gm	201.1

work. The trained network is able to interpolate only within the region where points $\boldsymbol{u}$ are distributed. The actual mapping learned by the network is visualized in Fig. 5 for a large number of random samples distributed uniformly in the $\boldsymbol{u}$ space.

At this stage the entire fabrication process model is developed. Using the model output characteristics $\boldsymbol{y}$ can be calculated based on any input $\boldsymbol{x}$ that belongs to the identified input characteristics' distribution. Conversely, by using the neural mapping inversion and inverse PCA operator, an input $\boldsymbol{x}$ for any output sample $\boldsymbol{y}$ can be found when $\boldsymbol{y}$ belongs to the output characteristics' distribution. The internal abstract representation $\boldsymbol{u}$ of both inputs and outputs is available for design centering tasks.

The goal of the following numerical illustration is to inspect and then maximize a MESFET simulated fabrication yield when the device target characteristics $\boldsymbol{y}_0$ are required with tolerance δ. The target $\boldsymbol{y}_0$ is shown in Table II. Three tolerance δ values: 5%, 10%, and 15% are considered.

TABLE III

FABRICATION YIELD FOR INVERSE SOLUTION AND CENTERED SOLUTION WITH VARIOUS ALLOWED TOLERANCES.

δ	inverse x_0	centered x_c
5%	7.2%	7.5%
10%	27.1%	27.6%
15%	52.1%	55.8%

By using a simple inversion–based approach, values of u_0 and x_0 can be found from equations (16) and then (10), (11), and (5). The inverse solution for input characteristics x_0 is included in the left column of Table IV.

Fabrication yield is then estimated assuming that input characteristics are attempted to be centered around the inverse solution x_0. Due to variations resulting from technology some of the fabricated devices have final characteristics y off the required tolerance δ which lowers the yield. To estimate the yield 1000 random input points distributed around x_0 with the original input data distribution variances are tested for tolerance requirements. The middle column in Table III contains the yield as a percentage number of points meeting the criteria.

The approach introduced in the previous chapter can be employed to improve the yield. The inverse solution u_0 is regarded as an initial condition to the optimization algorithm described by equation (25). Points neighboring u_0 are inspected for tolerance criterium after mapping to the output. Neighborhood of u_0 is shown in Fig. 6 for various tolerances δ. Points u which fail to fall into the tolerance region after mapping to the output as in equation (3), are marked with dots in these figures. Thus the blank area surrounded by dotted boundaries is the projection of the output tolerance region into the abstract space. Starting with the initial u_0, the optimization algorithm drags the center point to another location

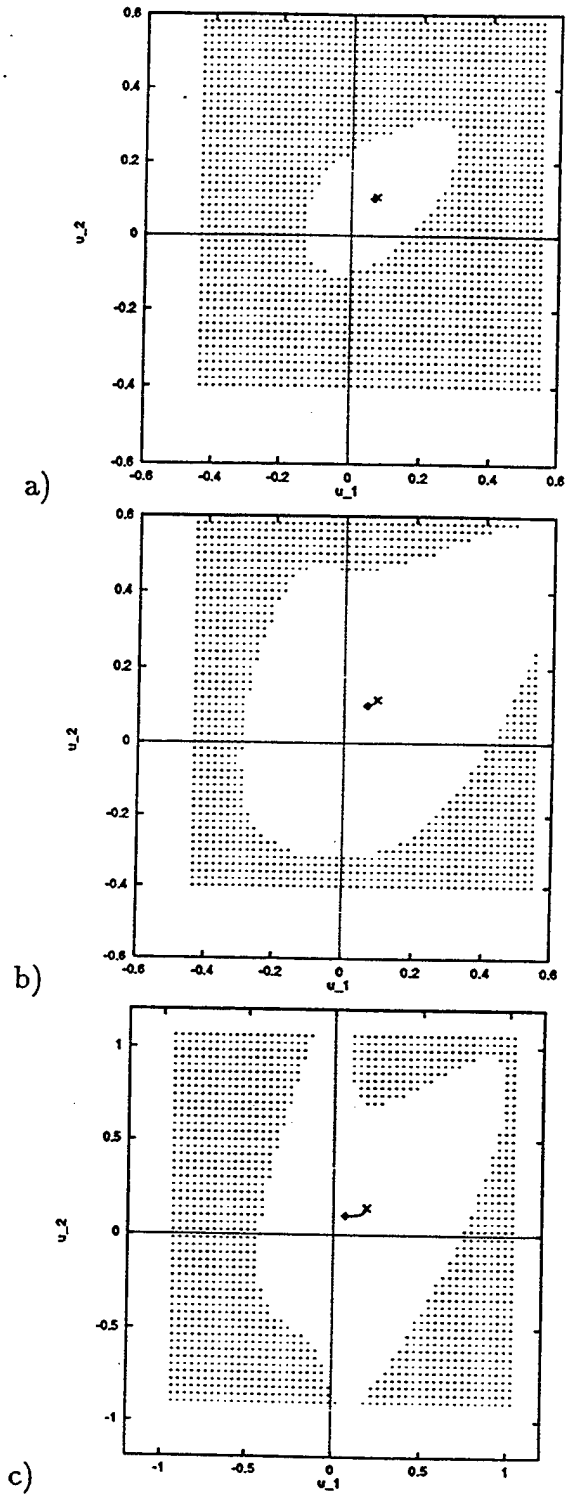


Fig. 6. Design centering in MESFET gate-final fabrication stage. Diamond represents an inverse u_0 to target y_0 in the abstract coordinates u_1–u_2. Centered value u_c, indicated by the cross, allows for the yield maximization, within tolerances δ equal (a) 5%, (b) 10%, and (c) 15%.

TABLE IV

INVERSE AND CENTERED SOLUTIONS FOR GIVEN TARGET AND TOLERANCES.

Characteristics	x_0	$x_c(5\%)$	$x_c(10\%)$	$x_c(15\%)$
G-Idss	218.661	218.45	217.853	214.729
G-Rds	2.85771	2.85954	2.8644	2.88793
G-Rgs	3.76301	3.76394	3.76624	3.77608
G-Rs	1.09795	1.09804	1.09836	1.10043
G-Rdg	3.44558	3.44691	3.45003	3.46232
G-Rd	0.797384	0.797955	0.799305	0.804678
G-Vpo	-1.22247	-1.22169	-1.22055	-1.22155
G-Gm	203.292	203.254	203.15	202.614
G-Rsh	0.0584099	0.058427	0.0584712	0.058677
G-Wg	10.0784	10.0784	10.0784	10.0794

u_c, which is considered the centered solution to the yield maximization task. Afterwards, the centered point u_c is transformed into the original input space and results with desired centered input x_c. Solutions for the three values of tolerance are shown in the right three columns of Table IV. The yield for these new centered solutions is then estimated in the same manner as for x_0. The improved yield is shown in the right column in Table III. As indicated in the table the design centering improves the yield especially when large tolerance for the target y_0 is required.

5. CONCLUSIONS

The presented design centering approach allows for yield maximization in fabrication processes without major changes to technology and available means. The yield can be significantly improved when nonlinear relationships are involved in the process characterization. This is the case in GaAs microelectronic devices manufacturing. The design centering algorithm can efficiently work even with large amount of measurement data since Principal Component Analysis of the data reduces the problem size and the "curse of dimensionality" is avoided.

REFERENCES

[1] G. L. Creech, J. M. Zurada, and P. B. Aronhime, "Feedforward neural networks for estimating ic parametric yield and device characterization," in *Proc. IEEE Int. Symposium on Circuit and Systems*, vol. 2, (Seattle, WA), pp. 1520–1523, Apr. 30–May 3, 1995.

[2] J. C. Zhang and M. A. Styblinski, *Yield and Variability Optimization of Integrated Circuits.* Kluwer Academic Publishers, 1995.

[3] G. L. Creech and J. M. Zurada, "Gaas mesfet dc characteristics and process modeling using neural networks," in *Proc. of the Artificial Neural Networks in Engineering*, (St. Louis, MI), Nov. 13–16, 1994.

[4] A. Cichocki and R. Unbehauen, *Neural Network for Optimization and Signal Processing.* John Wiley & Sons, 1993.

[5] J. M. Zurada, *Introduction to Artificial Neural Systems.* Boston: PWS, 1992.

[6] D. A. Hoskins, J. N. Hwang, and J. Vagners, "Iterative inversion of neural networks and its application to adaptive control," *IEEE Transactions on Neural Networks*, vol. 3, pp. 292–301, March 1992.

[7] A. Malinowski, *Reduced Perceptron Networks and Inverse Mapping Control.* PhD thesis, Univerity of Louisville, Louisville, KY, June 1996.

Jacek M. Zurada earned his M.S. and Ph.D. degrees from the Technical University of Gdansk, Poland. He now holds the S.T. Fife Alumni Professorship of Electrical Engineering at the University of Louisville, Louisville, Kentucky. He is the author of the 1992 West text "Introduction to Artificial Neural Systems," contributor to the 1994 and 1995 Ablex volumes "Progress in Neural Networks," and co-editor of the 1994 IEEE

Press volume "Computational Intelligence: Imitating Life." Dr. Zurada is author or co-author of more than 120 journal and conference papers in neural networks, analog and digital VLSI circuits, and active filters. Dr. Zurada is now an Associate Editor of "IEEE Transactions on Neural Networks," of "IEEE Transactions on Circuits and Systems, Part II," "Neurocomputing," and of the "Artificial Neural Networks Journal." He is an IEEE Fellow and Distinguished Speaker of the Neural Networks Council.

Aleksander Malinowski was born in Gdansk, Poland on September 27, 1966. He was a student at the Technical University of Gdansk in 1985–90. He received his Master of Science degree in May 1990 with specialization in the area of computer simulation and modeling of electronic circuits and VLSI. He was a faculty member of the Technical University of Gdansk in 1990–92. In August, 1992, he entered the University of Louisville, Louisville, KY. He has been recipient of University of Louisville Fellowship in 1992–96. He received his Doctor of Philosophy degree in August 1996 with specialization in the area of artificial neural systems and control. Currently he is a Postdoctorate Fellow at the University of Louisville and is doing research in the area of inverse modeling and design centering for VLSI fabrication. His current research interests include computational intelligence and artificial neural networks for modeling, neural network architecture optimization, and computer simulation and modeling. He has co-authored 4 journal and 22 conference papers.

Creech, G. L., Zurada, J. M.Neural Network Modeling of GaAs IC Material and MESFET Device Characteristic, Proc. of the OAI Neural Networks Symposium and Workshop, Athens, Ohio, August 21-22, 1995, pp. 257-266.

Neural Network Modeling of GaAs IC Material and MESFET Device Characteristics

GREGORY L. CREECH† and JACEK M. ZURADA‡

†*Solid State Electronics Directorate, Wright Laboratory*
Wright-Patterson AFB, OH 45433-7319
(513) 255-7663

‡*Department of Computer Science and Engineering*
University of Louisville
Louisville, KY 40292
(502) 852-6314

Abstract

This paper provides an overview of research focused on the utilization of neuro-computing technology to model critical in-process GaAs IC material and device characteristics. Artificial neural networks are employed to develop neural network models of complex relationships between material and device characteristics at critical stages of the semiconductor fabrication process. Measurements taken and subsequently used in modeling include doping concentrations, layer thicknesses, planar geometries, resistivities, and device voltages, and currents. The neural network architecture utilized in this research is the multilayer perceptron neural network (MLPNN). The MLPNN is trained in the supervised mode using the generalized delta learning rule. The MLPNN has demonstrated with good results the ability to model these characteristics and provide an effective tool for parametric yield prediction and whole wafer characterization in semiconductor manufacturing.

I. INTRODUCTION

Integrated-circuit (IC) technologies are expected to produce uniform device properties over a large wafer area. This uniformity is difficult to achieve for GaAs IC technology because of material and processing deviations. There are large variations, within a wafer, of important material properties which strongly influence yield-limiting factors in final device performance. In part, these material problems arise because of strong radial and axial variations in thermal gradients during bulk crystal growth, which affect local stoictriometry [1]. Other yield-limiting non-uniformities occur during the wafer fabrication process [2,3]. It is essential that these variations and the effects they have on device/circuit performance are understood and properly modeled.

Traditional IC process/device modeling approaches, whether analytical or empirical, do not utilize the parametric values specific to a certain device's location on a wafer. Variations of parametric values are typically represented statistically. Actually the values are random variables described by joint probability density functions [4,5]. Once the statistical distribution is determined, the effects of these variations on the device/circuits performance is analyzed by performing simulations by means of, among others, Monte Carlo techniques [6,7].

As shown in [1,8,9], many of these parametric variations do not occur in a random manner across a wafer but in a radial and/or axial pattern. Also, due to the physical correlations existing between FET characteristics these parameters should not be treated as uncorrelated,

---

This paper was partially supported by ONR Grant-N00014-93-1-0855

mutually independent random variables [10,11]. The modeling approach described in this paper presents a methodology in which a specific device's characteristics can be modeled based on its physical location within a wafer. Correlated variations are represented in the characteristic values of each individual device.

This research has focused on many different aspects of neural network modeling of semiconductor characteristics, two of which are presented in this paper. First, the development of neural network models for the estimation of IC parametric yield is demonstrated. Measurements of material and/or device characteristics taken at earlier fabrication stages are used to develop models of the final DC parameters. Yield-limiting characteristics are modeled and the resulting value compared to acceptance windows to estimate the parametric yield. Secondly, neural network models are developed in the inverse direction. Characteristics measured at Final are used as the input to model critical in-process characteristics. The modeled characteristics are used for whole wafer mapping and statistical characterization. This characterization can be accomplished with minimal in-process testing.

The concepts and methodologies used in the development of the neuro-models are presented. The modeling results are provided and compared to the actual measured values of each characteristic. A discussion of these results and the direction that any further research should take is provided.

II. PROCESS/DEVICE CHARACTERIZATION

IC manufacturing consists of many distinguishable fabrications stages. A large and representative number of measurements of process attributes, key device parameters, and layout geometries taken during the fabrication process is needed to provide a statistical database for neural network modeling of the process and/or IC devices. The classical method for obtaining the characteristics of semiconductor materials, processes, and devices is to collect data from microelectronic test structures [12] [13].

The test data used in this work was taken across an entire wafer at a sufficient density to fully characterize the fabrication process and device variations across the wafer. The measurement data used for characterization originated from a 4x4.5 mm high-density test structure reticle repeated some 200 times on a 3" Gallium Arsenide (GaAs) wafer. Each reticle contains an array of microelectronic test structures developed to analyze the uniformity of the fabrication process and the resulting device/circuit performance characteristics. The majority of the characteristics were measured on the Metal Semiconductor Field Effect Transistor (MESFET) device. This test structure/device (referred to as device from this point on) is at the center of this modeling effort.

Whole wafer testing was conducted on the starting substrate material (S) and during wafer processing at four critical steps: Ohmic or Post-contact (C), Post-recess (R), Post-gate (G), and at the completion of fabrication (Final or F). A discussion of the physical significance of each measured characteristic used in this modeling effort is beyond the scope of this paper. However, Table 1, lists each characteristic by the fabrication stage which they characterize, name, and symbol. Parameters which characterize the same fabrication stage are grouped together and serve as input-output for each neuro-model developed.

The characteristics were measured across the entire wafer in one test sweep. The parameter values are stored such that the reticle is identified by XXYY, and the structure within a reticle is identified with xxyy. Substrate characteristics, which are taken prior to the process step which defines the XXYY reticle location, are reported in millimeters. Computer routines have been written and verified that reference the millimeter data to the reticle locations. The measured characteristic is then assigned the respective reticle XXYYxxyy. This method of test structure identification allows for the tracking of parameter values for a specific device from one process stage to the next. This is imperative to MLPNN model development. The characteristics for a specific device must be tracked from one stage to the next to maintain the input-output relationships

Fab Stage	Characteristic Name	Symbol	Fab Stage	Characteristic Name	Symbol
S	2-Optical Scattering	(OBSA/B)	C	Drain-source sat. current	(C-Idss)
S	Neutral deep donor dens.	(EL2)	C	Drain-source resistance	C-Rds)
S	Substrate resistivity	(Rho)	C	Contact resistance	(Rc)
S.	Substrate Hall mobility	(MuH)	C	Contact metal sheet res.	(C-Rsh)
S	Substrate Carrier Conc.	(Ns)	C	Ohmic metal layer width	(O-W)
S	Doping Concentration	(Nd)	C	Ohmic metal sheet res.	(O-Rsh)
S	Implant Activation	(ETA)	R	Drain-source sat. current	(R-Idss)
S	Drift Mobility (Vg=0)	(Mu0)	R	Drain-source resistance	(R-Rds)
S	Drift Mobility (Vg=-1.5)	(Mu1)	F	Drain-source sat. current	(F-Idss)
G	Drain-source sat. current	(G-Idss)	F	Drain-source resistance	(F-Rds)
G	Drain-source resistance	(G-Rds)	F	Gate-source resistance	(F-Rgs)
G	Gate-source resistance	(G-Rgs)	F	Source resistance	(F-Rs)
G	Source resistance	(G-Rs)	F	Drain-gate resistance	(F-Rdg)
G	Drain-gate resistance	(G-Rdg)	F	Drain resistance	(F-Rd)
G	Drain resistance	(G-Rd)	F	Pinch-off voltage	(F-Vpo)
G	Pinch-off voltage	(G-Vpo)	F	Transconductance	(F-Gm)
G	Transconductance	(G-Gm)	G	Gate metal width	(G-W)
G	Gate metal sheet res.	(G-Rsh)			

S - Substrate/Active Layer C - Ohmic/Post-Contact R - Post-Recess
G - Post-Gate F - Final DC

TABLE 1. Material and MESFET device Characteristics modeled using neural networks. Characteristics for each respective fabrication stage serve as input-output pairs for model development and verification.

necessary for creating, training and modeling data sets. Also, the measured parameters location within a wafer are maintained for the purpose of wafer mapping.

III. DATA SELECTION and NETWORK ARCHITECHTURE

One of the principle objectives of this work is to model the effect that material and process variations have on the performance characteristics of the active devices used in integrated circuits. The active device is typically where the effects of these variations become most evident and is a major contributor to yield loss. Therefore, as mentioned earlier, the MESFET is at the center of this modeling effort. For network training purposes, a density of six data vectors per reticle was chosen. All measured parameters within a reticle are referenced to six specific XXYYxxyy MESFET locations. Training vectors are formed by assigning each of the non-MESFET characteristics to the nearest-neighbor MESFET. Training files were created for each of the fabrication process stages identified previously.

A training file consisting of data from each reticle would contain over 1200 training vectors. It is desirable to develop training files of a manageable size to train the neural network models in an efficient manner. Yet, one desires to have training files which statistically represent the variations across a wafer. Through the examination of the nature in which device variations occur [1,2], it was determined that a horizontal slice of reticles across the wafer would provide enough data to statistically characterize the wafer variations, yet provide a manageable data set.

Hence, a horizontal slice of 14 reticles across the middle of the wafer was chosen for training purposes. This provided 84 training data vectors. The data was analyzed and 15 data vectors, whose measurements indicated non-functional MESFETs (i.e. Idss=0, etc.), were discarded. Of the remaining 69 data sets, 50 were used to train the neural networks and 19 were reserved to test the neural networks. For testing the inverse models, data vectors of the wafer's

entire population of functional MESFETs, a total of 678, were used to perform whole wafer characterization of certain critical parameters. Each of the neural network models developed in this work are evaluated by comparing the actual values of these parameters to the modeled values.

The neural network architecture used in this modeling effort is the multilayer perceptron neural network. The MLPNN learns the similarities or patterns among sets of input-output data. The network is trained in the supervised mode using the generalized delta learning rule. It has one hidden layer, and uses continuous perceptrons. The algorithm used to implement the MLPNN was written in-house and is given in [14]. The size of the hidden layer in each MLPNN was determined experimentally by varying the number of hidden neurons and selecting the number which resulted in the lowest training error over a number of training sessions while maintaining adequate generalization. Each model took 20-40 minutes to train on a 100 Mhz computer. Once trained, the recall of the modeled parameters from the network is almost instantaneous.

IV. YIELD ESTIMATION

Accurate and computationally efficient methods for estimating integrated circuit (IC) parametric yield have been under development for years. In general, parametric yield is formulated by determining if the measured values of certain critical performance parameters fall within a predetermined tolerance range about the target value for that parameter. During IC fabrication, parametric test are performed to determine discrepancies between the actual performance and the desired performance. This can involve the screening of final, or F-stage, DC device parameters such as: saturated drain current, F-Idss; transconductance, F-Gm; and pinch-off voltage, F-Vpo. Accurate estimation of parametric yield during the manufacturing process relies on the ability to predict the effect of material and process variations on device parameters. The MLPNN models accomplish this task.

A. MLPNN Models

Three models of the F-stage DC characteristics were developed, refer to Figure 1, each model having input which represents a different stage of the fabrication process. Specifically the three models are denoted as: 1) **S->F**, which has 10 measurements used to characterize the substrate and active layer materials as input; 2) **CR->F**, which uses 8 measurements taken at the post-contact and post-recess stage; and 3) **G->F**, which uses 8 measurements made at post-gate as input. The characteristics for each respective stage are given in Table 1. The number of hidden layer perceptrons for each model was determined experimentally as 22.

The three MLPNN models are used to predict the values of F-Idss, F-Gm, and F-Vpo, as well as the other F-stage characteristics, for each of the 19 MESFETs in the test set. The yield is estimated by comparing these modeled values to the tolerance ranges for the respective characteristic. If the value falls within the range then it is considered to have passed, if not it fails. The estimated percent yield is then calculated and compared to the actual yield.

B. Results

Upon completion of training, the developed models were tested. Each test vector was used as input to the respective MLPNN model. The resulting outputs represent the modeled device characteristics at the final fabrication stage. For each MLPNN: 1) the modeled values have been compared to the actual measurement and the relative error calculated, and 2) the parametric yield has been estimated using the modeled values and have been compared to the actual parametric yield.

Figure 2 shows the average relative error between the MLPNN modeled values and the actual measurements for all the final DC parameters for each MLPNN. Each model perform a rather accurate computation of the device characteristics. As discovered in [15], the best model is

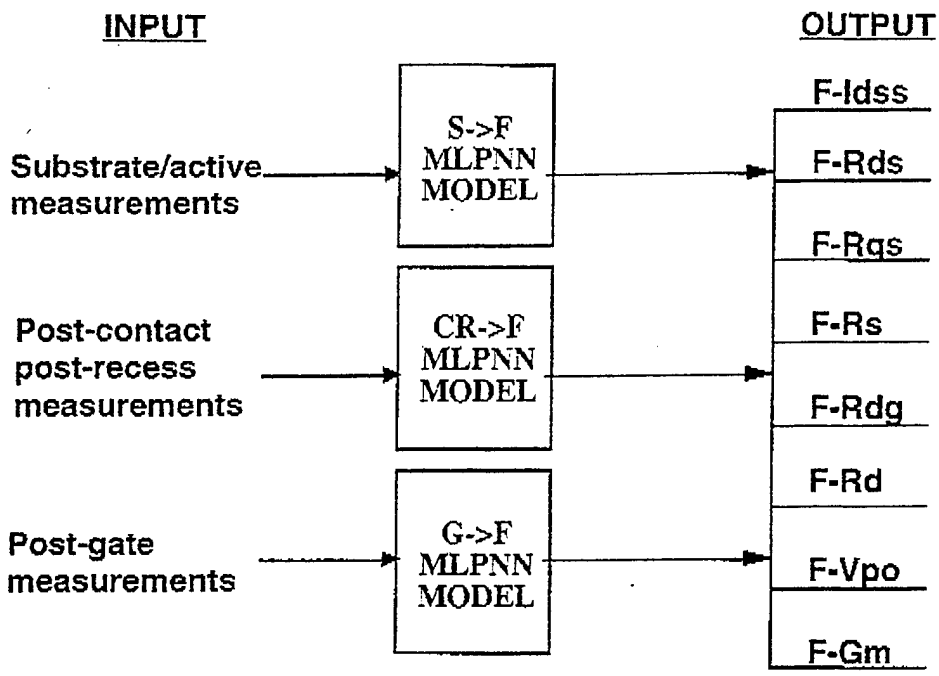


Figure 1. The three different MLPNN models developed. Each model independently predicts the output parameters.

the one which has the post-gate (G) data as input (i.e. G->F). The results obtained here, using the G-stage data exclusively as input, are better than those reported in [15], with errors at or less than 3% for all device characteristics.

Figure 3a-c, are bar charts of the actual yield and estimated yield using each MLPNN model's predicted values of F-Idss, F-Gm, and F-Vpo. The yield is calculated for three tolerance ranges; +/- 5% (Fig. 3a), +/- 10% (Fig. 3b), and +/- 20% (Fig. 3c). The tolerance ranges are computed as +/- 5%, 10%, and 20% of the parameters target values. The target value for each parameter is: Idss= 227 mA, Gm= 208 mS, and Vpo= -1.54 V.

As can be seen from Figure 3, the MLPNN computed values resulted in yield estimates which are very accurate. As suggested by the relative errors, the yield estimates were better for the MLPNN models developed using characteristics measured at the later stages of the fabrication process. The accuracy went from very good for the S->F MLPNN to excellent for the CR->F and G->F MLPNNs. Even for the tight tolerance range of 5%, the yield estimates are very credible.

IV. Inverse MLPNN Models

Developing methods to provide affordable and reproducible high frequency products is a major objective of the GaAs IC industry. Fundamental to meeting this objective is to increase circuit yields by developing uniform fabrication technologies. This requires the analysis and statistical characterization of critical process and device characteristics across many wafers. Ideally, this analysis would utilize whole wafer high density material, process, and device characteristics measured at critical stages of the fabrication process. A large number of measured characteristics, taken across many wafers, is needed to provide a statistical database for process and device characterization. The amount of testing required to obtain the data to implement the ideal approach is prohibitive. A dominant factor in the high cost associated with IC product development is testing requirements. Typically, whole wafer testing is only performed after fabrication is complete.

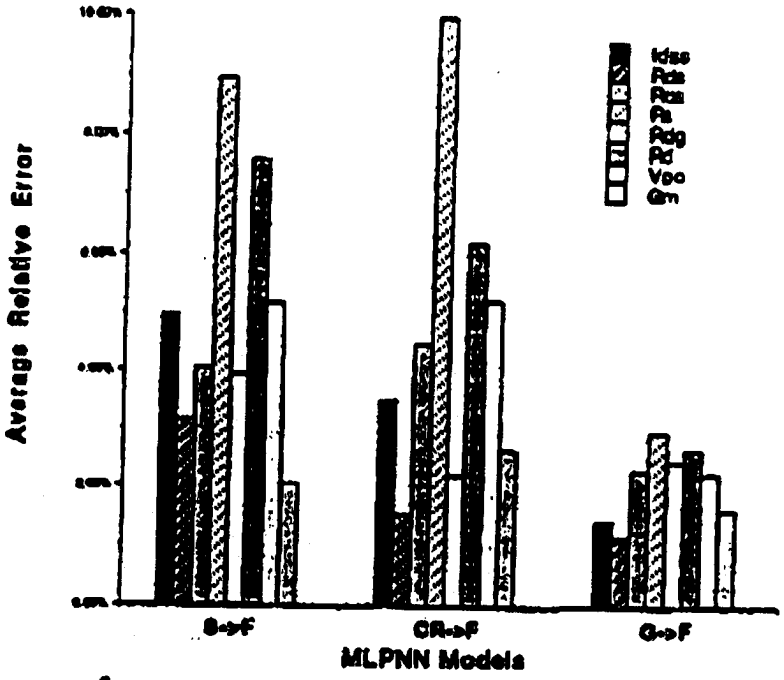


Figure 2. Average Relative Error Between MLPNN Modeled Values and Actual Measurements for Selected Fabrication Stages.

FROM : DR.JACEK ZURADA ELENG U LOUISV PHONE NO. : 502 852 6314 Mar. 26 1997 04:42PM P2

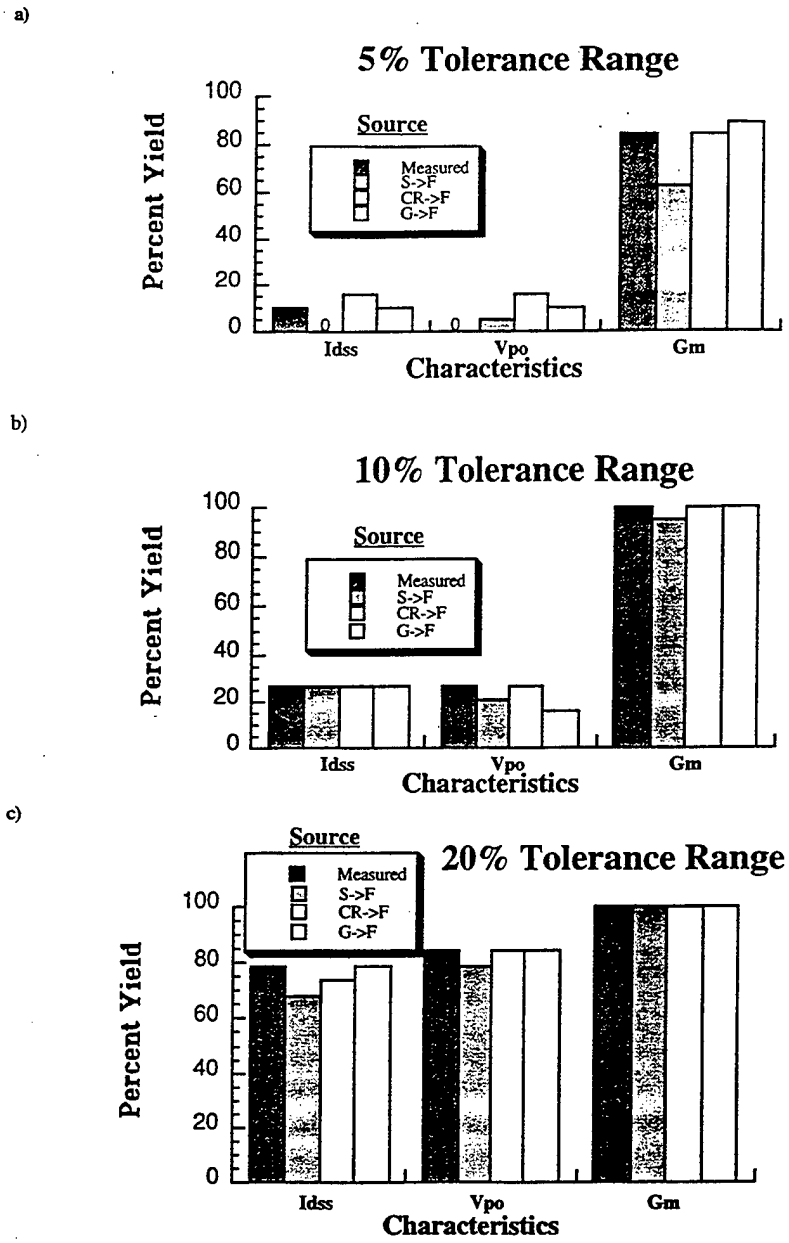


Figure 3. Comparision of actual and MLPNN computed yield a) 5% b) 10% c) 20% Tolerance.

The inverse modeling approach described here presents a methodology in which whole wafer in-process characterization is possible with minimal in-process testing. This reduced testing makes it affordable to analyze the process and device variations over many wafers. Thus, allowing one to examine the most crucial variations and the effect they have on IC performance.

The feed-forward neural network has been previously applied in such areas as microwave circuit analysis and optimization [16], microstrip circuit design [17], and device characterization for VLSI simulation [18]. More recently, the MLPNN has demonstrated, with good accuracy, the ability to model GaAs MESFET process and device characteristics in the forward direction [19].

A. MLPNN Models

All four critical stages of the GaAs MESFET IC fabrication process were selected for inverse modeling. That is, substrate/active layer (S), post-contact (C), post-recess (R), and post-gate (G). Figure 4 illustrates the four different process stage models developed. Each model has as input the same 8 F-stage characteristics listed in Table 1 and independently predicts the output characteristics of each respective fabrication stage. Due to the absence of whole wafer test data for some substrate/active layer characteristics and to improve model efficiency, the number of output characteristics for some of the inverse MLPNN models were slightly reduced from those listed in Table 1. Therefore, the symbol of the specific characteristics modeled for each stage are provided below. Refer to Table 1, for the names of the characteristics the symbols represent.

The four process stage models are denoted as: 1) **F->S**, which consists of 7 outputs. Best results during training were obtained using a hidden layer consisting of 17 perceptrons. The outputs of the **F->S** stage model are the characteristics of the bare substrate and the ion-implanted active layer: Nd, Ns, EL2, ETA, Rho, Mu0, MuH; 2) **F->C**, consists of 4 outputs. Best results during training were obtained using a hidden layer consisting of 12 perceptrons. The outputs of the **F->C** stage model are the post-contact characteristics: C-Idss, C-Rds, C-Rsh, O-Rsh; 3) **F->R**, consists of 2 outputs. Best results during training were obtained using a hidden layer consisting of 8 perceptrons. The outputs of the **F->R** stage model are the post-recess characteristics: R-Idss, R-Rds; 4) **F->G**, consists of 9 outputs. Best results during training were obtained using a hidden layer of 19 perceptrons. The outputs of the **F->G** stage model are the post-gate characteristic: G-Idss, G-Rds, G-Vpo, G-Rd, G-Rgs, G-Gm, G-Rs, G-Rdg, G-Rsh.

B. Results

Table 2, lists the statistical mean and standard deviation for the modeled values and the actual measurements for each characteristic modeled using the F->C, F->R, and F->G MLPNNs.

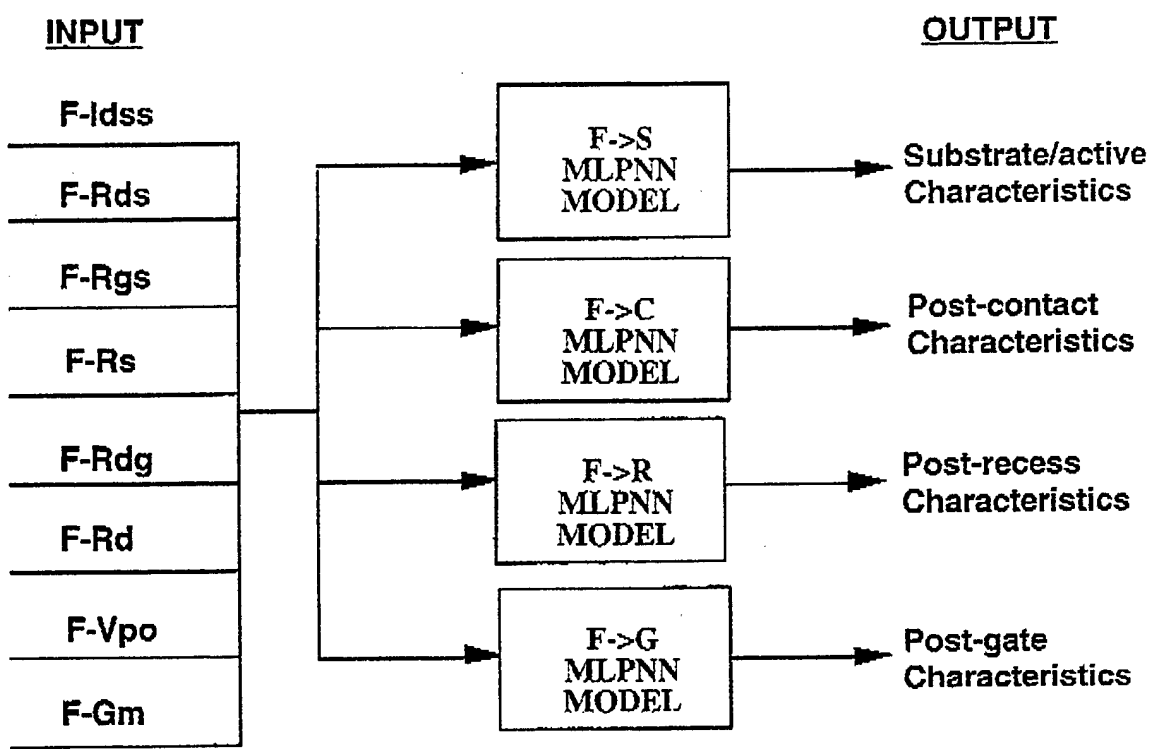


Figure 4. The four inverse MLPNN models developed.

The statistics were taken over the 678 test sets. The statistics of the MLPNN predicted values are in excellent agreement with the actual statistics. Figure 5a is the wafer map of the measured G-Idss values, while Figure 5b is the wafer map of the MLPNN modeled G-Idss. As can be seen the general spatial relationship of the characteristic across the wafer are recreated. Figure 6a-b, shows the same relationship for the R-Idss characteristic.

Table 3, lists the average relative error between the F->S MLPNN modeled values and the actual measurements. The error is slightly higher for these characteristics than those of the other stage models. That is to be expected considering the amount of processing the wafer is subjected to after these measurements are made. The final DC measurements seem to correlate best with the mobility. Whole wafer comparisons are not available for the S-stage characteristics because of the destructive nature of the test required for characterization. The results, while still acceptable, could possibly be better if the sites used for training were selected in a more optimal fashion.

Characteristic	Mean		STD. DEV.	
	Actual	Modeled	Actual	Modeled
G_Idss	221	228	36.2	36.7
G-Rds	2.83	2.78	0.267	0.247
G-Rgs	3.79	3.74	0.152	0.120
G-Rs	1.10	1.08	0.058	0.048
G-Rdg	3.47	3.43	0.191	0.130
G-Rd	0.798	0.752	0.079	0.056
G-Vpo	-1.43	-1.49	0.217	0.224
G-Gm	204	208	7.37	7.29
G-Rsh	0.059	0.057	0.0024	0.0023
R-Idss	638	643	34.3	37.8
R-Rds	2.35	2.43	0.129	0.113
C-Idss	925	918	28.2	36.5
C-Rds	1.63E4	1.63E4	751	780
C-Rsh	0.349	0.336	0.019	0.015

TABLE 2. Statistics for the actual and modeled values.

EL2	Rho	Ns	Muh	Nd	ETA	Mu0
5.9%	4.6%	4.2%	1.4%	5.9%	5.8%	1.3%

TABLE 3. Average relative error for F->S characteristics

V. CONCLUSIONS

This paper presents a new methodology for modeling of semiconductor process/device characteristics, in both the forward and inverse direction. The modeling technique discribed utilizes artificial neuro-computing technology. Specifically, the multilayer perceptron neural network (MLPNN) is employed for model development.

In the forward direction, measurements of characteristics taken at previous fabrication processing stages are used as input to a MLPNN and the next stage output values are modeled. For inverse modeling, whole wafer measurements of final DC device characteristics are used as input to a MLPNN and in-process characteristic values are modeled. This approach eliminates the

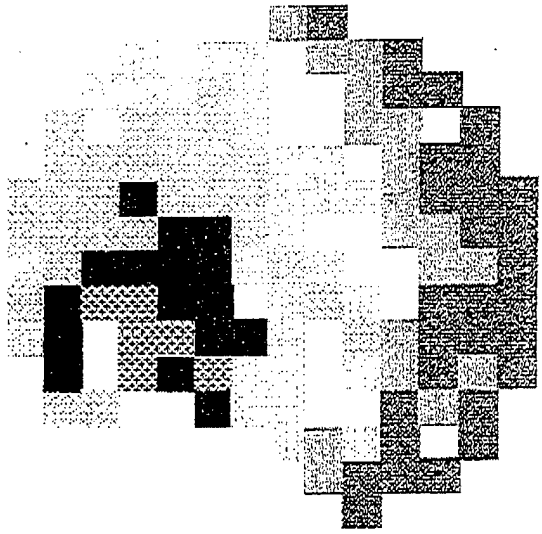


Figure 5a -Wafer Map of measured post-gate Idss.

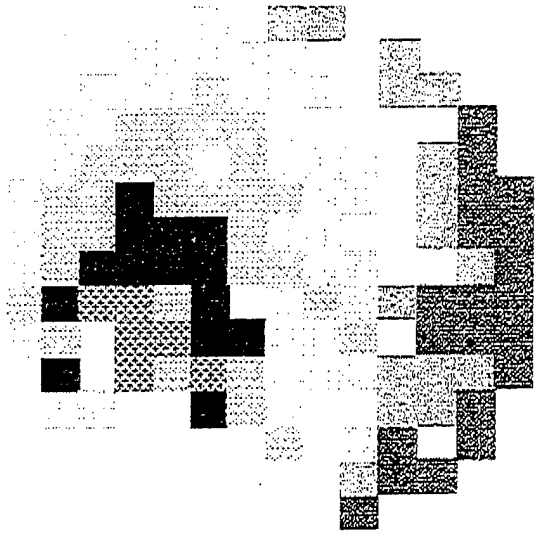


Figure 5b - Wafer Map of modeled post-gate Idss.

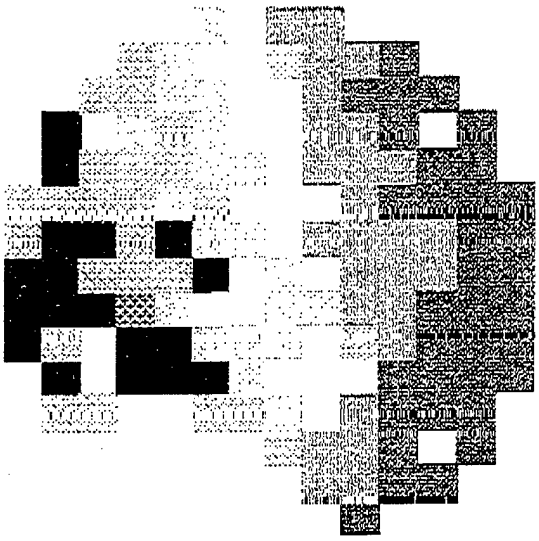


Fig. 6a -Wafer Map of measured post-recess Idss.

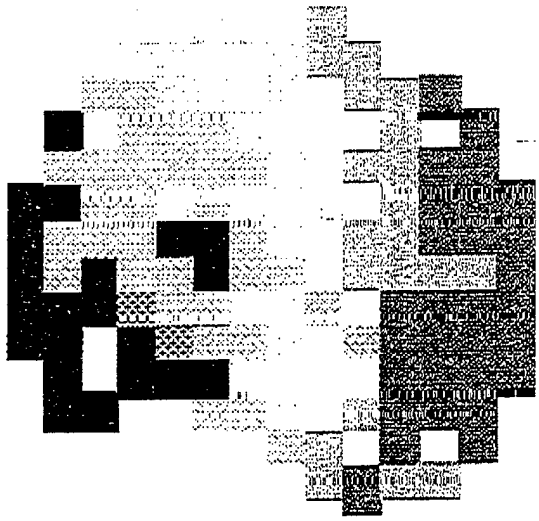


Fig. 6b - Wafer Map of modeled post-recess Idss.

need to statistically describe parametric variations across a wafer. Training is accomplished by using the actual measurements as input and output pairs. The MLPNN inherently encodes the statistics of these variations. The data presented show the approach can provide accurate results.

It is shown that the MLPNN model is a useful tool for estimating the parametric yield during the manufacturing process. There is excellent agreement between the actual yield and the estimated yield using the MLPNN modeled values. Also, we have demonstrated the MLPNNs ability to provide whole wafer statistics and wafer maps of important characteristics at critical stages of the fabrication process. This is accomplished by utilizing just a small amount of in-process testing.

The approach presented is technology independent and could be extended to other fabrication or production processes.

REFERENCES

1. D.C. Look, D.C. Walters, R.T. Kemerley, J.M. King, M.G. Mier, J.S. Sewell and J.S. Sizelove, "Wafer-Level Correlations of EL2, Dislocation Density, and FET Saturation Current at Various Processing Stages," *J. Electronic Material,* Vol. 18, No. 4, 1989.
2. J. King, et.al, "Overview of MIMIC Phase I Materials/Device Correlation Study," GOMAC Degist, 1991.
3. R.Anholt, J.King, J.Gillespie, R.Worley, "Relationship between process and materials variations and variations in S-parameters," *Int. J.Microwave, mm-Wave CAE* , Vol.1,
4. W. Maly, A. J. Strojwas, S. W. Director, "VLSI Yield Prediction and Estimation: A Unified Framework," IEEE *Trans. on Computer-Aided Design* , Vol. CAD-5, No. 1, Jan. 86,pp.114-130. No.3, 1991, pp.339-343.
5. C. Spanos, S. W. Director, "Parameter Extraction for Statistical Process Characterization," IEEE *Trans. on Computer-Aided Design of Integrated Circuits and Systems,* Vol. CAD-1, No. 5, Jan. 1986, pp. 66-78.
6. M. Styblinski, L. J. Opalski, "Algorithms and Software Tools for IC Yield Optimization based on Fundamental Fabrication Parameters," IEEE *Trans. on Computer-Aided Design of Integrated Circuits and Systems,* Vol. CAD-1, No. 5, Jan. 1986, pp. 66-78.
7. J.P. Spoto, W.T. Coston, and C.P. Hernandez, "Statistical Integrated Circuit Design and Characterization," IEEE *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-5, No. 1, Jan. 1986, pp. 90-103.
8. D.H. Rosenblatt, W.R. Hitchens, R.E. Anholt, and T.W. Sigmon, "GaAs MESFET Device Dependences on Ion-Implant Tilt and Rotation Angles," IEEE *Electron Device Letters*, Vol. 9, No. 3, March 1988.
9. P. Dobrilla, J.S. Blakemore, A.J. McCamant, K.R. Gleason, and R.Y. Koyama, "GaAs field-effect-transistor properties, as influenced by the local concentrations of midgap native donors and dislocations," *Applied Physics Letters,* Vol. 47, No. 6, 1985.
10. J. Purviance, M.D. Meehan, D.M. Collins, "Properties of FET Statistical Data Bases," *IEEE MTT Symp.*, 1990, 567-570.
11. R. Anholt, R. Worley, R. Neidhard, "Statistical Analysis of GaAs MESFET S-parameter Equivalent-Circuit Models," *Int. Journal. Microwave, mm-Wave CAE* , Vol.1, No.3, 1991, pp.263-270.
12. Khera, D., Zaghloul, M. E., Linholm, L. W., Wilson, C. L., "A Neural Network Approach for Classifying Test Structure Results," Proc. of the 1989 *Intern. Confer. on Microelectronic Test Structures,* Vol. 2, No. 1, March 1989, pp. 200-204.
13. Zaghloul, M. E., Khera, D., Linholm, L. W., Reeve, C. P., "A Machine-Learning Classification Approach for IC Manufacturing Control Based on Test Structure Measurements," IEEE *Trans. on Semic. Manuf.*, Vol. 2, No. 2, May 1989, pp. 47-53.
14. J.M. Zurada, "Introduction to Artificial Neural Systems," *West Publishing Co.*, St. Paul, Minn., 1992, pp.186-190.
15. G. L. Creech and J. M. Zurada, "GaAs MESFET DC Characteristics and Process Modeling Using Neural Networks," Intelligent Eng. Systems Through Art. Neural Networks, Vol. 4, ASME Press, 1994, pp. 1005-1113.
16. A.Zaabab,Q.Zhang, M.Nakhla, "Analysis and Optimization of Microwave Circuits and Devices Using Neural Network Models," *IEEE MTT-S Digest,* 1994, pp.393-396.
17. T.Horng,C.Wang,N.Alexopoulos, "Microstrip circuit design using Neural Networks," *IEEE MTT-S Digest,* 1993, pp.413-415.
18. P.Ojala,J. Saarinen, and K. Kaski, "Modified Back-propagation Neural Network Device Characterization for VLSI Circuit Simulation," Proceddings of WCNN, 1994.
19. G. L. Creech, J. M. Zurada, P. A. Aronhime, "Feedforward Neural Networks for Estimating IC Parametric Yield and Device Characterization," Proceedings of the IEEE International Symposium on Circuits and Systems, 1995.

GaAs Manf. Technology Conf., New Orleans, LA May 7-11, 1995 (accepted, oral pres.)

INVERSE MODELING OF MMIC FABRICATION MATERIAL AND DEVICE CHARACTERISTICS USING FEED-FORWARD NEURAL NETWORKS

GREGORY L. CREECH† and JACEK M. ZURADA‡

†*Solid State Electronics Directorate, Wright Laboratory*
Wright-Patterson AFB, OH 45433-7319
(513) 255-7663

‡*Department of Computer Science and Engineering*
University of Louisville
Louisville, KY 40292
(502) 852-6314

ABSTRACT

A novel, low cost, approach for modeling in-process material and device characteristics is described. Multilayer perceptron neural networks (MLPNN) are trained using error back propagation to model these characteristics at critical stages of the fabrication process. The modeled characteristics are used for whole wafer mapping and statistical characterization. We demonstrate, with good results, that the MLPNN models facilitate whole wafer analysis of in-process material and device variations with minimal in-process testing.

INTRODUCTION

Integrated-circuit (IC) technologies are expected to produce uniform device properties over a large wafer area. This uniformity is difficult to achieve for GaAs IC technology because of material and processing deviations. From wafer to wafer, as well as within a wafer, there are large variations of material and process properties which strongly influence important factors in MMIC performance [1,2]. It is essential that these variations are understood and properly modeled.

Developing methods to provide affordable and reproducible MMIC products is a major objective of the GaAs IC industry. Fundamental to meeting this objective is to increase circuit yields by developing uniform fabrication technologies. This requires the analysis and statistical characterization of critical process and device characteristics across many wafers.

Ideally, the process analysis would utilize whole wafer high density material, process, and device characteristics measured at critical stages of the fabrication process. A large number of measured characteristics, taken across many wafers, is needed to provide a statistical database for process and device characterization [2,3]. The amount of testing required to obtain the data to implement the ideal approach is prohibitive. A dominant factor in the high cost associated with MMIC product development is testing requirements. Typically, whole wafer testing is only performed after fabrication is complete.

The inverse modeling approach described in this paper presents a methodology in which whole wafer in-process characterization is possible with minimal in-process testing. This reduced testing makes it affordable to analyze the process and device variations over many wafers. Thus, allowing one to examine the most crucial variations and the effect they have on MMIC performance.

The feed-forward neural network has been previously applied in such areas as microwave circuit analysis and optimization [4], microstrip circuit design [5], and device characterization for VLSI simulation [6]. More recently, the MLPNN has demonstrated, with good accuracy, to ability to model GaAs MESFET process and device characteristics in the forward direction [7].

MLPNN MODELS

The neural network architecture used in this modeling effort is the multilayer perceptron neural network. The MLPNN learns the similarities or patterns among sets of input-output data. The modeled parameters are extracted empirically. In theory, neural networks have been shown to model any degree of non linearity [4]. The cost associated with implementing a neural network is low. Developing a neural network model is unlike software development, the network is trained, not programmed.

The MLPNN in this work is trained in the supervised mode using the generalized delta learning rule. It has one hidden layer, and uses continuous perceptrons. The algorithm used to

This paper was partially supported by ONR Grant-N00014-93-1-0855

implement the MLPNN was written in-house and is given in [8]. The size of the hidden layer in each MLPNN was determined experimentally by varying the number of hidden neurons and selecting the number which resulted in the lowest training error over a number of training sessions. Each model took 20-40 minutes to train on a 100 Mhz computer. Once trained , the recall of the modeled parameters from the network is almost instantaneous.

Four critical stages of the GaAs IC fabrication process were selected for modeling: Substrate/active layer (S), post-contact (C), post-gate-recess (R), post-gate metal (G). The letter preceding the characteristics listed below applies to the fabrication stages as denoted above. Each model uses as input 8 final DC device characteristics. They are:

•F-Idss, drain-source sat. current, (mA/mm)
•F-Rdg, drain-gate resistance, (ohm-mm)
•F-Rds, drain-source resistance, (ohm-mm)
•F-Rd, drain resistance, (ohm-mm)
•F-Rgs, gate-source resistance, (ohm-mm)
•F-Vpo, pinch-off voltage, (V)
•F-Rs, source resistance, (ohm-mm)
•F-Gm, transconductance, (mS/mm)

Fig.1 illustrates the four different process stage models developed. The network model for the **S** process stage, denoted as **F->S**, consists of 7 outputs. Best results during training were obtained using a hidden layer consisting of 17 neurons. The outputs of the **F->S** stage model are the characteristics of the bare substrate and the ion-implanted active layer:

•S-Nd, doping concentration, (cm^3)
•S-Ns, substrate carrier concentration, (cm^3)
•S-EL2, neut deep donor density, (cm^2)
•S-ETA, implant activation, (%)
•S-Rho, substrate resistivity, (ohm-mm)
•S-Mu, drift mobility (Vg=0), (cm^2/V-sec)
•S-MuH, substrate Hall mobility, (cm^2/V-sec)

The network model for the **C** process stage, denoted as **F->C**, consists of 4 outputs. Best results during training were obtained using a hidden layer consisting of 12 neurons. The outputs of the **F->C** stage model are the post-contact characteristics:

•C-Idss •C-Rds
•C-C_Rsh, Contact metal sheet resistance
•C-O_Rsh, Ohmic metal sheet resistance

The network model for the **R** process stage, denoted as **F->R**, consists of 2 outputs. Best results during training were obtained using a hidden layer consisting of 8 hidden neurons. The outputs of the **F->C** stage model are the post-recess characteristics; •R-Idss •R-Rds

The network model for the **G** process stage, denoted as **F->G**, consists of 9 outputs. Best results during training were obtained using a hidden layer consisting of 19 hidden neurons. The outputs of the **F->G** stage model are the post-gate characteristic:

•G-Idss •G-Rds •G-Vpo •G-Rd
•G-Rgs •G-Gm •G-Rs •G-Rdg
•G-Rsh, Gate metal sheet resistance

DATA SELECTION

The data used in this work originated from measurements taken on a 4x4.5 mm high-density test structure reticle repeated some 200 times per wafer. The fabrication process used an ion-implanted active layer and a recess-etched gate with a nominal length of 0.5 µm. Process and device characteristics were measured at a sufficient density to fully characterize variations across the wafer. The reticles contain an array of 0.5x200 µm MESFET, Van der Pauw patterns, transmission line models, and standard process control monitor structures. Whole wafer testing was conducted on the substrates and during wafer processing at four critical steps: Ohmic or Post-contact, Post-recess, Post-gate, and Final. The majority of the characteristics have been measured on the 0.5x200 µm MESFET. This test structure device is at the center of this modeling effort. The parameter values are stored such that the reticle is identified by XXYY, and the structure within a reticle is identified with xxyy. This method of test structure identification allows for the tracking of parametric values for a specific device from one process stage to the next. This is imperative to MLPNN model development. The characteristics for a specific device must be tracked from one stage to the next to maintain the input-output relationships necessary for creating training and modeling data sets. Also, measured parameters location within a wafer are maintained for wafer mapping.

A horizontal slice of 14 reticles across the middle of the wafer was chosen for training purposes. These reticles where chosen for two reasons; 1) due to the nature in which device variations occur [1], 2) they contained the only available properly formatted substrate and active layer characteristics. This provided 84 data sets. The data was screened for non-functional MESFETs (i.e. Idss=0, etc.), which were excluded from training. Of the remaining 69 data sets, 50 were used to train the MLPNNs and 19 were reserved to test the MLPNN performance at modeling the substrate and active layer characteristics. After training the final DC characteristics of the wafer's functional MESFETs, total of 678, were used as input to the MLPNNs and whole wafer characterization of certain critical parameters was performed. To

demonstrate the MLPNN performance the actual values of these parameters are compared to the MLPNN modeled values.

RESULTS

Upon completion of training, each MLPNN model was tested. First, each of the 19 test sets (the ones **not** used in training) were input to the F->S MLPNN model. The resulting outputs represent the modeled substrate and active layer characteristics. The average relative error between the MLPNN modeled values and the actual measurement are computed. Secondly, the 678 test sets where input to the three remaining MLPNN models F->C, F->R, and F->G. The mean and standard deviation is computed for the MLPNN modeled values and the actual measurements. Whole wafer mapping of certain critical characteristics is provided for comparison purposes.

F->C, F->R, and F->G Stage MLPNNs

Table 1, list the statistical mean and standard deviation for the MLPNN modeled values and the actual measurement for each characteristic. The statistics were taken over the 678 test sets. Each model performs a rather accurate computation of each characteristic. The MLPNN modeled values provide the process engineer with a very good indication of each characteristics actual statistics. Fig. 2a is the wafer map of the actual G-stage Idss and Fig. 2b, is the wafer map of the MLPNN modeled G-Idss. As can be seen the general spatial relationship of the characteristic across the wafer are recreated. Fig. 3a-b, shows the same relationship for the R-Idss. Whole wafer mapping, with similar results, is available for each of the characteristics listed in Table 1.

F->S Stage MLPNN

Table 2, list the substrate/active layer characteristics and the associated average relative error taken over the 19 test sets.. The error is slightly higher for these characteristics than those of the other stage models. That is to be expected considering the amount of processing the wafer is subjected to after these measurements are made.

The final DC measurements seem to correlate best with the mobility. Again, whole wafer comparisons are not available for the S stage characteristics. The results, while still acceptable, could possibly be better if the sites used for training were selected in a more optimal fashion.

CONCLUSIONS

This paper presents a new low cost method for modeling of semiconductor material and device characteristics using multilayer perceptron neural networks. Whole wafer measurements of final DC device characteristics are used as input to a MLPNN and in-process characteristic values are modeled. In-process measurements representing only 5% of whole wafer testing are used to develop the MLPNN model. We have demonstrated the MLPNNs ability to provide whole wafer statistics and wafer maps of important characteristics at critical stages of the fabrication process. Further more this is accomplished by utilizing just a small amount of in-process testing.

TABLE 1. Stats. for actual and modeled values.

CHARS	MEAN		STD. DEV.	
	Actual	Modeled	Actual	Modeled
G_Idss	221	228	36.2	36.7
G-Rds	2.83	2.78	0.267	0.247
G-Rgs	3.79	3.74	0.152	0.120
G-Rs	1.10	1.08	0.058	0.048
G-Rdg	3.47	3.43	0.191	0.130
G-Rd	0.798	0.752	0.079	0.056
G-Vpo	-1.43	-1.49	0.217	0.224
G-Gm	204	208	7.37	7.29
Rsh-G	0.059	0.057	0.0024	0.0023
R-Idss	638	643	34.3	37.8
R-Rds	2.35	2.43	0.129	0.113
C-Idss	925	918	28.2	36.5
C-Rds	1.79	1.77	0.074	0.068
O_Rsh	1.63E4	1.63E4	751	780
C_Rsh	0.349	0.336	0.019	0.015

TABLE 2. Avg. rel. error for F->S characteristics

EL2	Rho	ns	Muh	Ndp	ACT%	Mu
5.9%	4.6%	4.2%	1.4%	5.9%	5.8%	1.3%

REFERENCES

1. R.Anholt, J.King, J.Gillespie, R.Worley, "Relationship between process and materials variations and variations in S-parameters," *Int. J.Microwave, mm-Wave CAE* , Vol.1, No.3, 1991, pp.339-343.
2. J. King, et.al, "Overview of MIMIC Phase I Materials/Device Correlation Study," GOMAC Degist, 1991.
3. C.E. Schuster, et.al, "High Density Test Structures For Assessing MIMIC performance," GOMAC Digest, 1991.
4. A.Zaabab,Q.Zhang, M.Nakhla, "Analysis and Optimization of Microwave Circuits and Devices Using Neural Network Models," *IEEE MTT-S Digest*, 1994, pp.393-396.

5 T.Horng,C.Wang,N.Alexopoulos, "Microstrip circuit design using Neural Networks," *IEEE MTT-S Digest,* 1993, pp.413-415.
6. P.Ojala,J. Saarinen, and K. Kaski, "Modified Back-propagation Neural Network Device Characterization for VLSI Circuit Simulation,"
7. G. L. Creech and J. M. Zurada "GaAs MESFET DC Characteristics and Process Modeling Using Neural Networks," Intelligent Eng. Systems Through Art. Neural Networks, Vol. 4, ASME Press, 1994 (Proc. of ANNIE '94 conf. St. Louis, Mo., Nov.13-16, 1994).
8. J.M. Zurada, "Introduction to Artificial Neural Systems," *West Publishing Co.*, St. Paul, Minn., 1992, pp.186-190.

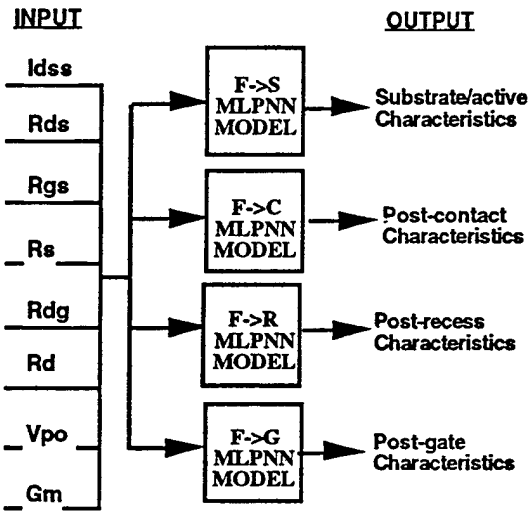


Figure 1. - The four different MLPNN models. Each model independently predicts its outputs.

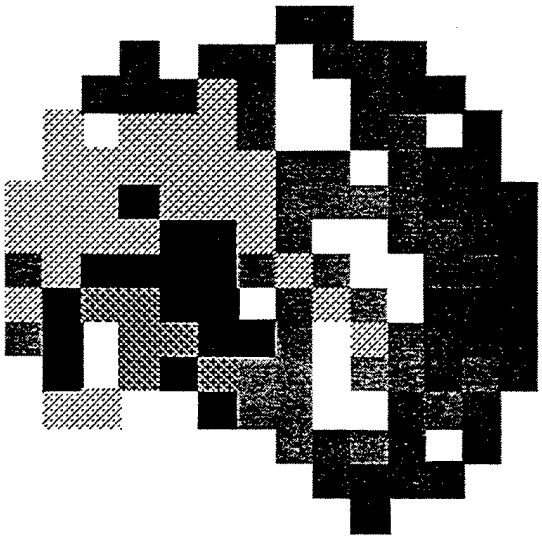


Figure 2a -Wafer Map of measured post-gate Idss.

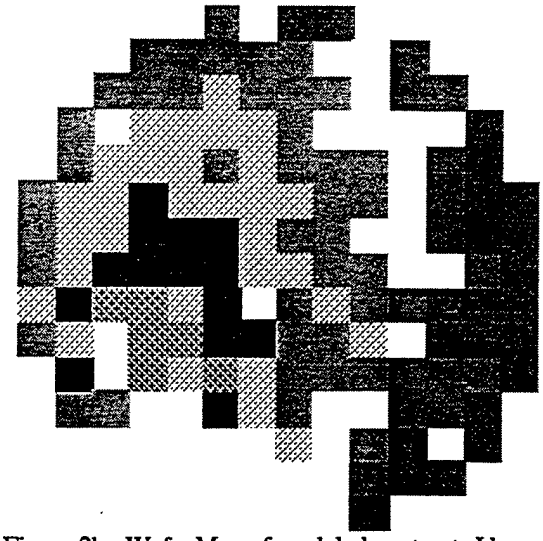


Figure 2b - Wafer Map of modeled post-gate Idss.

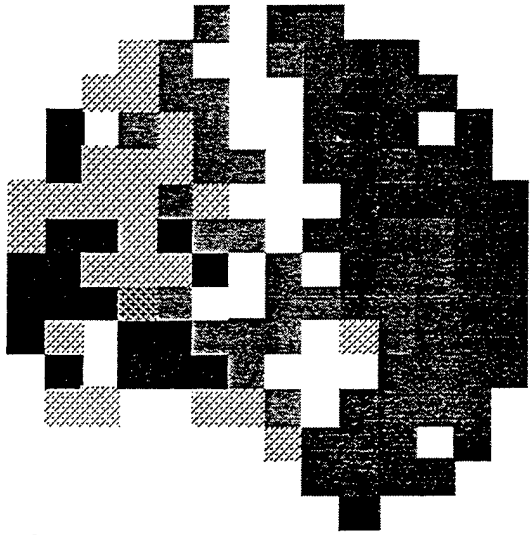


Fig. 3a -Wafer Map of measured post-recess Idss.

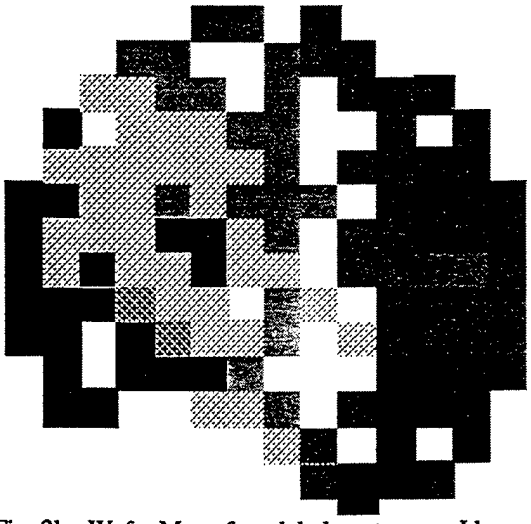


Fig. 3b - Wafer Map of modeled post-recess Idss.

1995 IEEE International Symposium on
Circuits and Systems (ISCAS'95)
Seattle, Washington
Apr. 29 - May 3, 1995

FEEDFORWARD NEURAL NETWORKS FOR ESTIMATING IC PARAMETRIC YIELD AND DEVICE CHARACTERIZATION

Gregory L. Creech*, Jacek M. Zurada, and Peter B. Aronhime

Computer Science and Engineering Program
University of Louisville
Louisville, KY 40292
ph. (502) 852-6289, fax (502) 852-6807
jmzura02@ulkyvx.louisville.edu

Abstract

A unique and accurate approach for modeling semiconductor device characteristics and estimating IC parametric yield is described. Multilayer perceptron neural networks (MLPNN) are trained using error back propagation to model DC device characteristics measured at the final fabrication stage. Measurements of material and/or device characteristics taken at earlier fabrication stages are used to develop neural network models of the final DC parameters. A very good agreement has been found between the actual measurements and the MLPNN modeled parameters, and the resulting yield estimations are in excellent agreement with the actual yield.

Introduction

Accurate and computationally efficient methods for performing semiconductor device characterization and for estimating integrated circuit parametric yield have been under development for years [1]. In general, parametric yield is computed by determining if a key device parameter's measured value falls within a certain tolerance range. IC technologies are expected to produce uniform device properties over a large wafer area. This uniformity is especially difficult to achieve for GaAs IC technology because of material and processing deviations. From wafer to wafer, as well as within a wafer, there are large variations of material and process properties which strongly influence important factors in final device/circuit performance [2].

Traditional IC process/device modeling approaches, whether analytical or empirical, do not utilize the parametric values specific to a certain device's location on a wafer. Variations of parametric values are typically represented statistically. The parametric values are actually treated as random variables described by joint probability density functions [1,3]. Once the statistical distribution is determined, the effect of the material/process variation on the device/circuit's performance is analyzed by performing simulations using Monte Carlo techniques [1,4,5].

As shown in [2,6,7], many of these parametric variations do not occur in a random manner across a wafer but in a radial and axial pattern. The modeling approach described in this paper presents a methodology in which a specific device's characteristics can be modeled based on its physical location within a wafer. At the start of and at intermediate stages of the fabrication process material and device measurements are taken on or at the location of a specific MESFET. These measurements are used to model those specific MESFET characteristics measured at the final fabrication stage. Variations are represented in the characteristics values of each individual device.

Each model is developed through the supervised learning of an MLPNN. The approximating neural network progressively combines the device variation and its statistics into the fitting relationship [8]. The MLPNNs have demonstrated the ability to model complex processes and device parameters with good accuracy [9], but using these models for yield estimation is still a rather unexplored subject.

Device Characterization/Data Selection

IC manufacturing consists of many distinguishable fabrication stages. A large and representative number of measurements of material and device characteristics is needed to provide a

This work was partially supported by ONR Grant -N00014-93-1-0855, *Permanent Address: Wright Laboratory, WL/ELMT, WPAFB, Ohio 45433-7327

statistical database for neural network modeling of IC devices.

The data used in this work was taken across an entire wafer at a sufficient density to fully characterize the GaAs device variations across the wafer. The measurement data used for characterization originated from a 4x4.5 mm high density test structure reticle repeated some 200 times per wafer. Whole wafer testing was conducted on the substrates (S) and during wafer processing at four critical steps: Post-contact and Post-recess (CR), Post-gate (G), and Final. The majority of the characteristics have been measured on the 0.5x200 micron MESFET. This test structure device is at the center of this modeling effort. The data was prepared as described in detail in [9].

Due to the nature of device variations [2], it was determined that a horizontal slice of reticles across the wafer would provide enough data to characterize the wafer variations, yet provide a manageable data set. A horizontal slice of 14 reticles across the middle of the wafer was chosen. This provided 84 training vectors. The data was then analyzed and 15 training vectors, whose measurements indicated non-functional MESFETs, i.e. Idss=0, were discarded. Of the remaining 69 vectors, 50 were used to train the network models, and 19 were used to test the models. The training vectors were shuffled in a random manner before training to create test files.

Network Models

The MLPNN network, with continuous perceptrons and one hidden layer, has been trained in the supervised mode using the generalized delta learning rule. The algorithm used to implement and train the MLPNN is given in [10]. The number of hidden layer perceptrons for each model was determined experimentally as 22. This was accomplished by varying the number of hidden neurons and selecting the number which resulted in the lowest training error over a number of training sessions.

Three models of the final DC characteristics were developed, each model having input which represents a different stage of the fabrication process. The final DC device characteristics modeled are:

- •F drain-source saturation current (Idss)
- •F drain-source resistance (Rds)
- •F gate-source resistance (Rgs)
- •Source resistance (Rs)
- •Drain-gate resistance (Rdg)
- •Drain resistance (Rd)
- •Pinch-off voltage (Vpo)
- •Transconductance (Gm)

Fig. 1 illustrates the three different stage models developed. The S->F stage model uses 10 measurements to characterize the substrate and active layer materials as input. These are:

- •Two Optical Scattering (OBS)
- •Neut deep donor density (EL2)
- •Substrate resistivity (Rho)
- •Substrate Hall mobility (MuH)
- •Substrate Carrier Conc. (ns)
- •Doping Concentration (Nd)
- •Implant Activation (ETA)
- •Drift Mobility (Vg=0) (Mu0)
- •Drift Mobility (Vg-1.5) (Mu1)

The CR->F stage model uses 8 measurements taken at the post-contact and post-recess stage as input. These are:

- •P-C drain-source saturation current (C-Idss)
- •P-C drain-source resistance (C-Rds)
- •P-R drain-source saturation current (R-Ids)
- •P-R drain-source resistance (R-Rds)
- •Contact resistance (Rc)
- •Contact metal sheet resistance (C-Rsh)
- •Ohmic metal layer width (O-W)
- •Ohmic metal sheet resistance (O-Rsh)

The G->F stage model uses 8 measurements made at post-gate as input. These are:

- •P-G drain-source saturation current (G-Idss)
- •P-G drain-source resistance (G-Rds)
- •Gate-source resistance (G-Rgs)
- •Source resistance (G-Rs)
- •Drain resistance (G-Rd)
- •Pinch-off voltage (G-Vpo)
- •Transconductance (G-Gm)
- •Drain-gate resistance (G-Rdg)

The development of the three separate models allow for device characterization and parametric yield estimation at these three distinct stages of the IC manufacturing process.

Yield Estimation

Parametric tests are performed during IC fabrication to determine discrepancies between the actual performance and the desired performance. This can involve screening of key DC device parameters such as: saturated drain current, Idss; transconductance, Gm; and pinch-off voltage, Vpo [1]. Accurate estimation of parametric yield during the manufacturing process relies on the ability to predict the effect of process variations on device parameters. The MLPNN models accomplish this

task.

The three MLPNN models extrapolate the values of Idss, Gm, and Vpo, as well as the other characteristics. The yield is estimated by comparing the modeled values to the tolerance ranges for the respective characteristic. If the value falls within the range, then the device is considered to have passed. The estimated percent yield is then calculated and compared to the actual yield.

Results

Upon completion of training, the developed stage models were tested. Each test vector is used as input to the respective MLPNN model. The resulting output represents the modeled device characteristics at the final fabrication stage. For each MLPNN: 1) the modeled values have been compared to the actual measurement and the relative error calculated, and 2) the parametric yield has been estimated using the modeled values and have been compared to the actual parametric yield.

Fig. 2 shows the average relative error between the MLPNN modeled values and the actual measurements of all the final DC parameters for each MLPNN. Each model performs a rather accurate computation of the device characteristics. As discovered in [9], the best model is the one which has the post-gate (G) data as input (i.e. G->F). The results obtained here using the G-stage data exclusively as input are better than those reported in [9] with errors at or less than 3% for all device characteristics.

Fig. 3a-c are bar charts of the actual yield and estimated yield calculated using each of the MLPNN model's predicted values of Idss, Gm, and Vpo. The pattern in which the bar is filled represents the source of the values used for the yield calculation. When a zero (0) appears in the chart, this is to indicate zero yield for that parameter. The yield is calculated for three tolerance ranges; +/-5% (Fig. 3a), +/-10% (Fig. 3b), and +/-20% (Fig. 3c). The tolerance ranges are computed as 5%, 10%, and 20% of the parameter's target value. The target value for each parameter is: Idss=227 mA, Gm=208 mS, and Vpo=-1.54 V.

As can be seen from Fig. 3, the MLPNN computed values resulted in yield estimates which are very accurate. As suggested by the relative errors, the yield estimates were better for the MLPNN models developed using characteristics measured at later stages of the fabrication process. The accuracy went from very good for the S->F MLPNN to excellent for the CR->F and G->F MLPNNs. Even for the tight tolerance range of 5%, the yield estimates are very credible.

Conclusions

This paper presents both a new methodology and new results for modeling of semiconductor device characteristics and performing parametric yield estimation using multilayer perceptron networks. Measurements of material and device characteristics taken at early fabrication stages are used as input to a MLPNN and final DC device characteristics are modeled. This approach eliminates the need to statistically describe parametric variations across a wafer. Training is accomplished using the actual measurements as input and output pairs. The trained MLPNN inherently encodes the statistics of these variations. The data presented show that the approach can provide accurate results. The authors acknowledge that the number of MLPNN test cases may not be fully sufficient and are currently extending this work to full wafer evaluation.

It is also shown that the MLPNN model is a useful tool for estimating the parametric yield during the manufacturing process. There is an excellent agreement between the actual yield and the estimated yield using the MLPNN computed values. Moreover, the approach presented is technology-independent and could be extended to other fabrication or production processes.

References

[1] W. Maly, A.J. Strojwas, S.W. Director, "VLSI Yield Prediction and Estimation: A Unified Framework," *IEEE Trans. on Computer-Aided Design*, Vol. CAD-5, No. 1, Jan. 86, pp. 114-130.

[2] D.C. Look, D.C. Walters, R.T. Kemerley, J.M. King, M.G. Mier, J.S. Sewell and J.S. Sizelove, "Wafer-Level Correlations of EL2, Dislocation Density, and FET Saturation Current at Various Processing Stages," *J. Electronic Material*, Vol. 18, No. 4, 1989.

[3] C. Spanos, S.W. Director, "Parameter Extractions for Statistical Process Characterization," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-1, No. 5, Jan. 1986, pp. 66-78.

[4] M. Styblinski, L.J. Opalski, "Algorithms and Software Tools for IC Yield Optimization Based on Fundamental Fabrication Parameters," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-1, No. 5, Jan. 1986, pp. 66-78.

[5] J.P. Spotto, W.T. Coston, and C.P. Hernandez,

"Statistical Integrated Circuit Design and Characterization," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-5, No. 1, Jan. 1986, pp. 90-103.

[6] D.H. Rosenblatt, W.R. Hitchens, R.E. Anholt, and T.W. Sigmon, "GaAs MESFET Device Dependences on Ion-Implant Tilt and Rotation Angles," *IEEE Electron Device Letters*, Vol. 9, No. 3, March 1988.

[7] P. Dobrilla, J.S. Blakemore, A.J. McCamant, K.R. Gleason, and R.Y. Koyama, "GaAs Field-Effect Transistor Properties, as Influenced by the Local Concentrations of Midgap Native Donors and Dislocations," *Applied Physics Letters*, Vol. 47, No. 6, 1985.

[8] W. Davis "Process Variation Analysis Employing Artificial Neural Networks," *Proc. of the Intern. Joint Confer. on Neural Networks*, Baltimore, Maryland, June 7-11, 1992, vol. II, pp. 260-265.

[9] G.L. Creech, J.M. Zurada, "GaAs MESFET DC Characteristics and Process Modeling Using Neural Networks," *Intelligent Eng. Systems Through Art. Neural Networks*, vol. 4, ASME Press, 1994 (Proc. of the ANNIE '94 conf., St. Louis, Mo., Nov. 13-16, 1994.

[10] J.M. Zurada, "Introduction to Artificial Neural Systems," West Publishing Company, St. Paul, Minn., 1992, pp. 186-190.

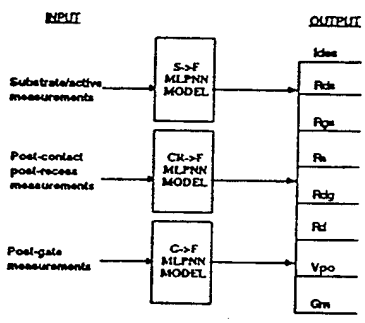


Fig. 1 The Three Different MLPNN Models Developed. Each Model Independently Predicts the Output Parameters.

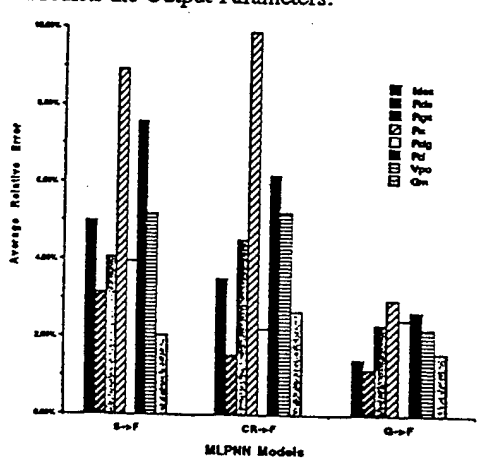
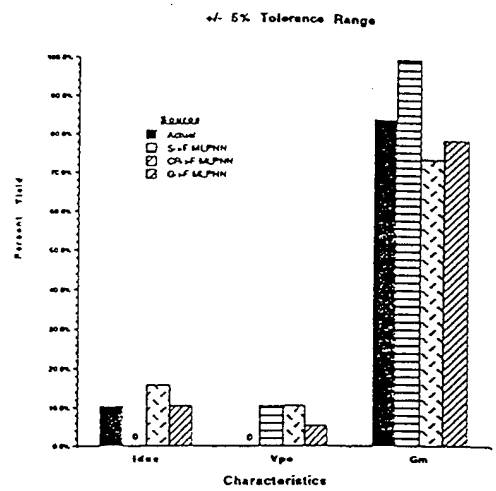
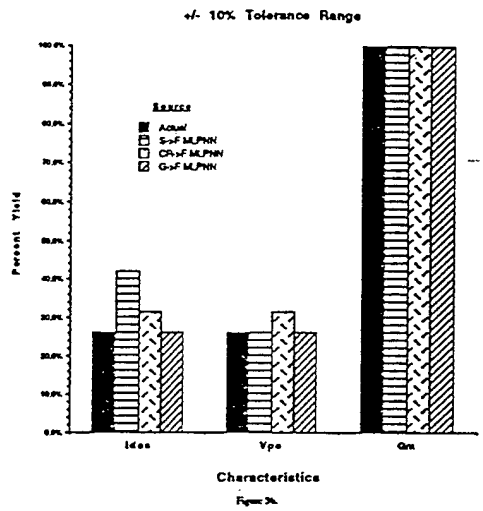


Fig. 2 Average Relative Error Between MLPNN Modeled Values and Actual Measurements for Selected Fabrication Stages.





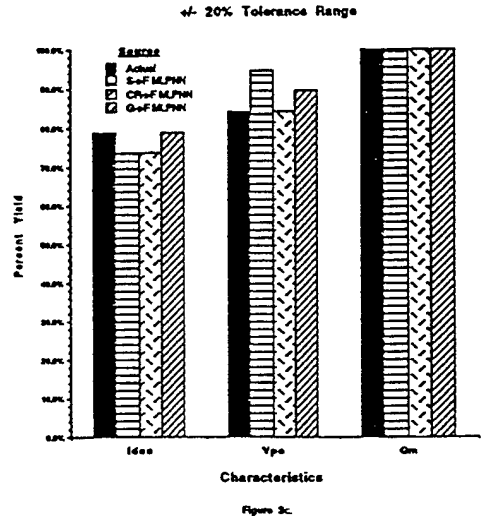


Fig. 3 Comparison of Actual and NN Computed Yield
a) 5% b) 10% c) 20% Tolerance

ASME PRESS SERIES ON
INTERNATIONAL ADVANCES IN DESIGN PRODUCTIVITY

INTELLIGENT ENGINEERING SYSTEMS THROUGH ARTIFICIAL NEURAL NETWORKS

VOLUME 4

Proceedings of the Artificial Neural Networks in Engineering (ANNIE '94) conference, held November 13-16, 1994, in St. Louis, Missouri, U.S.A.

EDITED BY

Cihan H. Dagli
Dept. of Engineering Management
University of Missouri-Rolla
Rolla, Missouri, U.S.A.

Benito R. Fernández
Dept. of Mechanical Engineering
The University of Texas at Austin
Austin, Texas, U.S.A.

Joydeep Ghosh
Dept. of Electrical and
Computer Engineering
The University of Texas at Austin
Austin, Texas, U.S.A.

R. T. Soundar Kumara
Dept. of Industrial and
Management Systems Engineering
Penn State University
University Park, Pennsylvania, U.S.A.

EW YORK

GaAs MESFET DC CHARACTERISTICS AND PROCESS MODELING USING NEURAL NETWORKS

GREGORY L. CREECH* and JACEK M. ZURADA
Department of Computer Science and Engineering
University of Louisville
Louisville, KY 40292

ABSTRACT

A unique approach for modeling semiconductor device characteristics using multilayer perceptron neural networks (MLPNN) trained using the error back propagation is described. Measurements of material, process, and device characteristics at various fabrication stages are used to develop neural network models of these characteristics and of the final DC parameters at various processing stages. There is very good agreement between the actual measurements and the MLPNN modeled parameters.

INTRODUCTION

Integrated-circuit (IC) technologies are expected to produce uniform device properties over a large wafer area. This uniformity is difficult to achieve for GaAs IC technology because of material and processing deviations. From wafer to wafer, as well as within a wafer, there are large variations of material and process properties which strongly influence important factors in final device/circuit performance [1].

Traditional IC process/device modeling approaches, whether analytical or empirical, do not utilize the parametric values specific to a certain device's location on a wafer. Variations of parametric values are typically represented statistically. Actually the parametric values are treated as random variables described by joint probability density functions [2]. Once the statistical distribution is determined, the effect of the material/process variation on the device/circuits performance is analyzed by performing simulations by means of, among others, Monte Carlo techniques [3, 4].

As shown in [1, 5, 6], many of these parametric variations do not occur in a random manner across a wafer but in a radial and axial pattern. The modeling approach described in this paper presents a methodology in which a specific device's characteristics can be modeled based on its physical location within a wafer. Material and process stage measurements are taken on a specific MESFET and/or nearby test structures. These measurements are used to model that specific MESFET at the next fabrication stage. The variations are represented in the characteristic values of each individual test structure and device. Since this method utilizes the actual measured values it eliminates the need to develop statistical representations of parameter variations.

Each model is developed through the supervised learning of a MPLNN. The approximating neural network progressively combines the process stage variation and its statistics into the output parameters fitting relationship [7].

---

*Permanent Address: Wright Laboratory, WL/ELMT, WPAFB, Ohio 45433-7327
This paper was supported by ONR Grant-N00014-93-1-0855

DEVICE/PROCESS CHARACTERIZATION

IC manufacturing consists of many distinguishable fabrications stages. A large and representative number of measurements of process attributes, key device parameters, and layout geometry's taken during the fabrication process is needed to provide a statistical database for neural network modeling of the process and/or IC devices. The classic method for obtaining the characteristics of semiconductor materials, processes, and devices is to collect data from microelectronic test structures [8, 9].

The data used in this work was taken across the entire wafer at a sufficient density to fully characterize the wafer. The measurement data used for characterization originated from a 4x4.5 mm high-density test structure reticle repeated some 200 times per wafer. The reticles contain an array of the 0.5x200 micron MESFET, Van der Pauw patterns, transmission line models, and standard process control monitor structures. Whole wafer testing was conducted on the substrates and during wafer processing at four critical steps: Ohmic or Post-contact, Post-recess, Post-gate, and Final. The majority of the characteristics have been measured on the 0.5x200 micron MESFET. This test structure/device (referred to as device from this point on) is at the center of this modeling effort.

NETWORK MODELS

The neural network architecture used in this modeling effort is the MLPNN. The network is trained in the supervised mode using the generalized delta learning rule. It has one hidden layer, and uses continuous perceptrons. The size of the hidden layer was determined experimentally by varying the number of hidden neurons and selecting the number which resulted in the lowest training error over a number of training sessions.

The following stages of the GaAs IC fabrication process were selected for device modeling: Contact and Gate-Recess (CR), Gate metal (G), and Final (F). Fig.1 illustrates the hierarchical manner in which the process stages are modeled. The inputs for the CR stage model are the measurements which characterize the bare substrate and the ion-implanted active layer (S). These include:

•Two Optical Scattering	(OBS)	•Doping Concentration	(Nd)
•Neut deep donor density	(EL2)	•Implant Activation	(ETA)
•Substrate resistivity	(Rho)	•Drift Mobility (Vg=0)	(Mu0)
•Substrate Hall mobility	(MuH)	•Drift Mobility (Vg=1.5)	(Mu1)
•Substrate Carrier Conc.	(ns)		

The CR stage outputs are the two MESFET device parameters measured after the Post-contact (P-C) stage and again at the Post-recess (P-R) stage and the layer width and sheet resistance of the ohmic and contact metal:

•P-C drain-source sat. cur.	(C-Idss)	•Contact resistance	(Rc)
•P-C drain-source res.	(C-Rds)	•Contact metal sheet res.	(C-Rsh)
•P-R drain-source sat. cur.	(R-Ids)	•Ohmic metal layer width	(O-W)
•P-R drain-source res.	(R-Rds)	•Ohmic metal sheet res.	(O-Rsh)

The network model for the CR process stage, denoted as S=>CR, consists of 10 inputs and 8 outputs. Best results during training were obtained using a hidden layer consisting of 20 hidden neurons and the bias input.

The next model of a process stage, Gate metal (G), has 18 inputs. The inputs to this process stage model are all the characteristics measured prior to this process

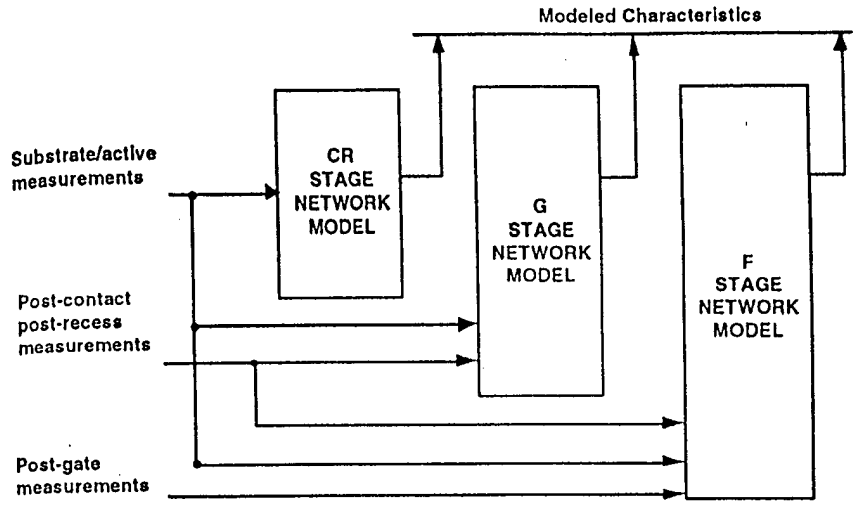


Figure 1 - Illustration of the hierarchical manner the process stages are modeled.

taking place. These are the ones listed above for the inputs and outputs of the S=>CR model. The outputs of G stage network model are the characteristics which are measured at post-gate (P-G). These include:

•P-G drain-source sat. cur.	(G-Idss)	•Gate-source brkdown volt.	(G-Vbgs)
•P-G drain-source res.	(G-Rds)	•Pinch-off voltage	(G-Vpo)
•Gate-source resistance	(G-Rgs)	•Transconductance	(G-Gm)
•Source resistance	(G-Rs)	•Peak - Ids	(G-Ids-pk)
•Drain-gate resistance	(G-Rdg)	•Electrical alignment	(G-AL)
•Drain resistance	(G-Rd)	•Gate metal sheet res.	(G-Rsh)
•Drain-gate brkdown volt.	(G-Vbdg)	•Gate metal width	(G-W)

The network model for the G process stage, denoted as SCR=>G, consists of 18 inputs and 14 outputs. Best results during training were obtained using a hidden layer consisting of 16 hidden neurons and the bias input.

The next characterization occurs when the front-side fabrication is complete. Many of the same characteristics which are measured at post-gate are measured again at final (F). However, these characteristics change in value due to further processes the wafer goes through after the gate metal is deposited. The final stage network model has 32 inputs. These inputs are all the characteristics measured prior to final test and are described above. There are 19 outputs for the F stage model which represent the measurements taken at the final stage. These are:

•F drain-source sat. cur.	(F-Idss)	•Peak - Ids	(F-Ids-pk)
•F drain-source res.	(F-Rds)	•Gate length	(Lg)
•F gate-source res.	(F-Rgs)	•Mimic-Cap	(C)

•Source res.	(F-Rs)	•Interconnect sheet res.	(i-Rsh)
•Drain-gate res.	(F-Rdg)	•Interconnect width	(i-W)
•Drain res.	(F-Rd)	•Resistor film sheet res.	(p-Rsh)
•Drain-gate brkdown volt.	(F-Vbdg)	•Resistor film width	(p-W)
•Gate-srce brkdown volt.	(F-Vbgs)	•Barrier height	(BH)
•Pinch-off voltage	(F-Vpo)	•Back-gating	(BG)
•Transconductance	(F-Gm)		

The F stage network model, denoted as SCRG=>F, consists of 32 inputs and 19 outputs. The best results during training were obtained using a hidden layer consisting of 22 hidden neurons and the bias input.

Data Selection and Preparation

A large and representative number of measurements of process attributes, key device parameters, and layout geometry's taken during the fabrication process is utilized to provide a statistical database for the neural network models. Test structures have been measured across the entire wafer in one test sweep. The parameter values are stored such that the reticle is identified by XXYY, and the structure within a reticle is identified with xxyy. This method of test structure identification allows for the tracking of parametric values for a specific device from one process stage to the next. The characters in the filename have been chosen in such a manner that the fabricator, process lot, boule source, wafer number, characteristic measured, and the processing stage at which the measurement was performed is easily identifiable. Substrate characteristics, which are taken prior to the process step which defines the XXYY reticle location, are reported in millimeters. Computer routines have been written and verified that reference the millimeter data to the reticle locations. The measured characteristic is then assigned the respective reticle XXYYxxyy.

The majority of the characteristics have been measured on the MESFET. For network training purposes, a density of six training vectors per reticle was chosen. All measured parameters within a reticle are referenced to six distinct MESFET locations. Training vectors were formed by assigning each of the non-MESFET characteristics to the nearest-neighbor MESFET. Training files were created for each of the fabrication process stages identified previously.

For this work, it was initially desirable to develop training files of a manageable size to train the neural network models. A training file consisting of data from each reticle would contain over 1200 training vectors. Yet, one desires to have training files which statistically represent the variations across a wafer. Due to the nature in which variations occur [1], it was determined that a horizontal slice of reticles across the wafer would satisfy both goals. A horizontal slice of 14 reticles across the middle of the wafer was chosen. This provided 84 training vectors. The data was analyzed and 14 training vectors, whose measurements indicated non-functional MESFETs, was discarded, i.e. Idss=0. Of the remaining 70 vectors, 51 were used to train the network models and 19 were used to test the network models. The training vectors were shuffled in a random manner before training and test files were created.

RESULTS

Upon completion of training, the developed models were tested. Each test vector is used as input to the respective MLPNN model. The resulting outputs represent the modeled process attributes or device characteristics for that respective fabrication stage. The modeled values have also been compared to the actual measurement.

TABLE 1. Relative error for all modeled CR stage characteristics.

MESFET	C-Idss	C-Rds	Rc	C-Rsh	O-Rsh	O-W	R-Idss	R-Rds
1	7.61%	4.81%	55.07%	2.88%	2.56%	0.20%	10.29%	2.32%
2	2.33%	2.20%	96.49%	2.86%	1.67%	0.71%	4.85%	8.21%
3	4.09%	4.08%	72.46%	13.70%	0.53%	0.32%	5.10%	5.74%
4	2.83%	2.27%	5.14%	0.59%	4.07%	1.96%	1.80%	2.36%
5	0.94%	2.16%	0.09%	2.24%	2.53%	1.13%	2.33%	4.25%
6	2.61%	1.78%	16.37%	2.43%	0.82%	0.11%	3.43%	1.26%
7	4.92%	3.61%	5.93%	5.56%	2.74%	0.18%	9.56%	3.16%
8	3.06%	0.43%	36.54%	1.03%	0.87%	0.35%	4.14%	2.32%
9	0.21%	1.91%	5.35%	3.01%	0.11%	0.72%	1.93%	1.65%
10	3.78%	3.40%	93.87%	8.43%	2.71%	0.66%	6.70%	1.59%
11	3.55%	1.34%	6.58%	0.17%	0.13%	0.13%	4.49%	5.77%
12	1.81%	0.39%	21.76%	3.03%	0.21%	0.44%	3.83%	0.86%
13	2.79%	0.24%	16.80%	0.99%	2.25%	1.29%	3.04%	0.25%
14	0.48%	1.07%	8.66%	1.25%	0.27%	0.30%	0.25%	0.57%
15	1.85%	2.32%	23.10%	0.27%	0.68%	0.30%	3.00%	5.42%
16	2.80%	3.20%	35.26%	0.11%	1.66%	0.48%	4.72%	5.98%
17	0.61%	0.57%	13.61%	3.94%	0.03%	0.49%	1.43%	4.67%
18	1.43%	1.34%	58.83%	4.58%	0.21%	0.13%	3.97%	1.86%
19	1.68%	0.59%	59.17%	1.07%	0.26%	0.11%	2.48%	1.78%

S=>CR Stage

It is important to remember that each test vector contains measurements of specific MESFETs or nearby test structures. Also, each modeled value is compared to the measurements made at the next stage on the same MESFET or test structure. For ease of discussion, from this point on each test vector and its resulting output will be referred to as MESFET characteristics.

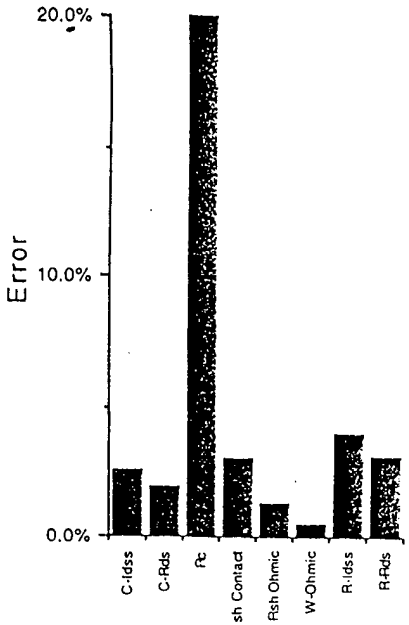
Table 1 shows the percent error of the modeled CR characteristics. Each row list the errors for a specific device. It can be seen that there exist a good agreement between most of the measured and modeled parameters except for contact resistance, Rc. The uncommon appearance of double digit errors indicates that the MLPNN did not train well on this parameter.

Fig. 2a shows the average error for each of the CR characteristics. The accuracy of the modeled values, with exception of Rc, is on average good to very good. Fig. 3a and 3b are scatter plots of the measured and modeled source-drain current at post-contact (C) and post-recess (R) respectively.

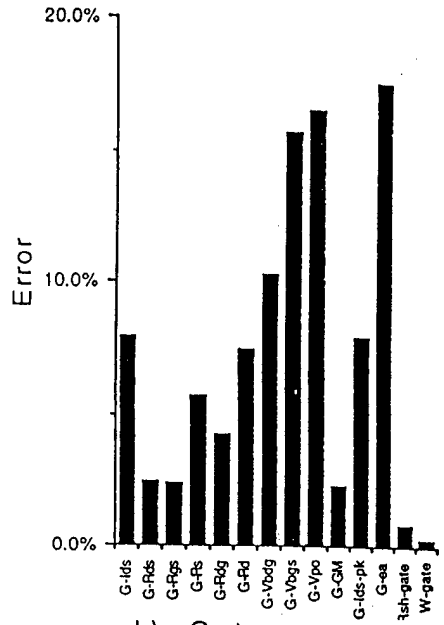
SCR=>G Stage

The tabulation of errors, such as shown in Table 1 for the CR parameters, is available for each subsequent stage. However, as the number of modeled parameters increases, the tables become rather large and will not be shown in this paper. Fig. 2b shows the average error for each of the 14 G-stage parameters. The errors associated with this stage are on average slightly higher than at the CR stage.

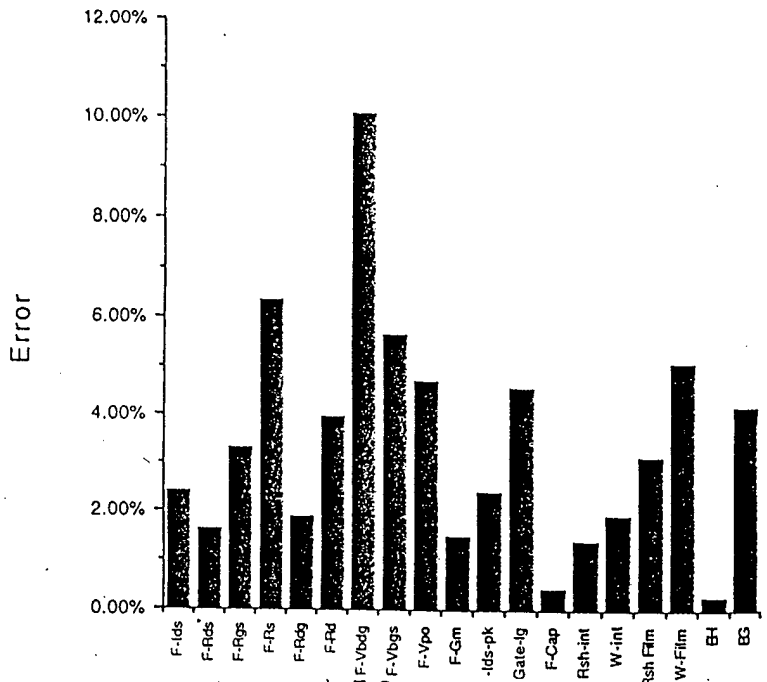
This increase in error could be attributed to the increase in the number of parameters the MLPNN is required to learn. The G-stage fabrication process is the most critical processing step with respect to the effect on device performance. Most of the G-stage parameters are either measured for the first time or the value shifts dramatically during the fabrication process. The MLPNN seems to have the most



a) CR-stage parameters



b) G-stage parameters



c) F-Stage parameters

FIGURE 2 - Average relative error of each process stage modeled.

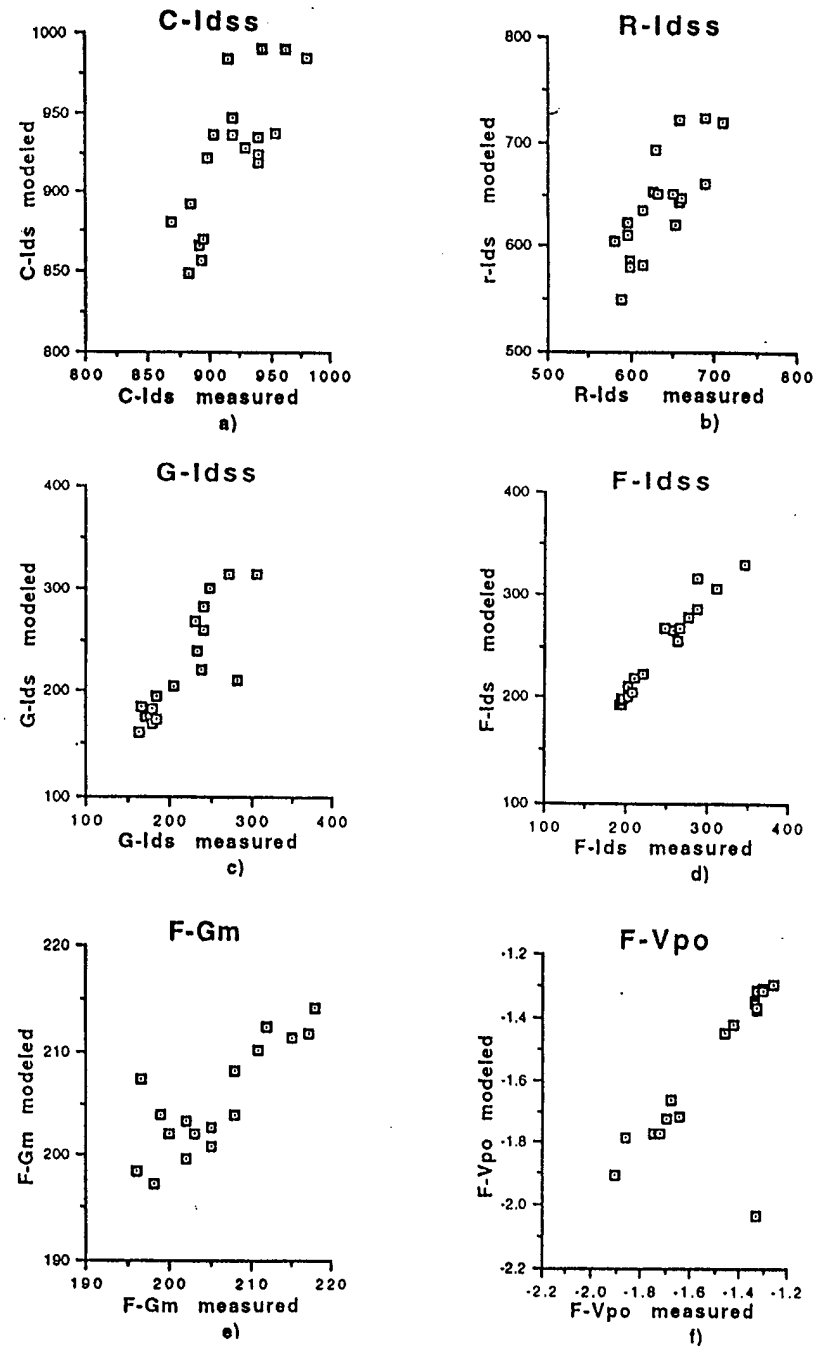


FIGURE 3 - Scatter plots of selected parameters.

trouble learning the relationships of the G-stage voltages. Fig. 3c shows a scatter plot of the G-stage measured and modeled source-drain current.

SCRG=>F Stage

It is rather common for the most extensive testing to take place after the final processing stage. This is where the key device characteristic are compared to the target values and it is determined if the devices are functional. In this work the best results were obtained from the F-stage model. Fig. 2c shows the average error for each of the F-stage parameters. All but one modeled parameter has an average error of less than about six percent. Many show an error of less than three percent. The drain-gate breakdown voltage, Vbdg, exhibits the largest average error of about ten percent.

The chart of Fig. 4 shows the average error for the key MESFET characteristics measured at both post-gate and at the final stage. There is a noticeable improvement in the MLPNN performance from the G to F stage. All but one parameter shows a decrease in average error and the errors indicate excellent agreement.

Fig. 3a-d show scatter plots of the measured and modeled source-drain current at each of the subsequent fabrication stages. C-Ids and R-Ids are outputs of the CR-stage model. The plots show good agreement between measured and modeled Ids, but the values do not fit as tightly about the diagonal axis as desired. As the MLPNN received more information on which to train, as with the G-stage model, there is a tighter grouping for G-Idss. The best fit along the diagonal axis is achieved from the F-stage model. This is expected since at this stage the MLPNN has the most process and device data on which to train. Fig. 3e and 3f are scatter plots of other key MESFET characteristics, pinch-off voltage and transconductance.

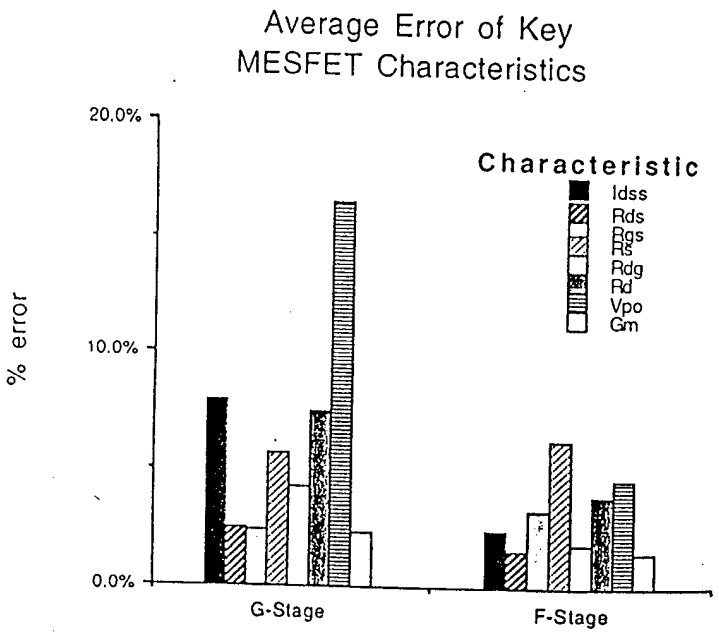


FIGURE 4 - Average error for the same characteristics at both the G and F stage.

CONCLUSIONS

This paper presents both a new methodology and new results for modeling of semiconductor process/device characteristics using multilayer perceptron neural networks. Measurements of characteristics taken at previous fabrication processing stages are used as input to a MLPNN and the next stage output values are modeled. This approach eliminates the need to statistically describe parametric variations across a wafer. This is accomplished by using the actual measurements as input and output pairs during the training of the MLPNN. The MLPNN inherently obtains the statistics of these variations. The data presented show the approach can provide accurate results.

More work is currently underway to determine the feasibility of extending this methodology to provide wafer-level yield analysis early in the fabrication process. For the approach to be practical in an industrial setting, the number of measurements required for successful model development need to be kept at a minimum. Therefore, further study of the sensitivity of the output parameters to the specific inputs needs to be performed. Inputs which do not significantly influence the MLPNN outputs may be pruned.

REFERENCES

1. D.C. Look, D.C. Walters, R.T. Kemerley, J.M. King, M.G. Mier, J.S. Sewell and J.S. Sizelove, "Wafer-Level Correlations of EL2, Dislocation Density, and FET Saturation Current at Various Processing Stages," *J. Electronic Material*, Vol. 18, No. 4, 1989.
2. C. Spanos, S. W. Director, "Parameter Extraction for Statistical Process Characterization," IEEE *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-1, No. 5, Jan. 1986, pp. 66-78.
3. M. Styblinski, L. J. Opalski, "Algorithms and Software Tools for IC Yield Optimization based on Fundamental Fabrication Parameters," IEEE *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-1, No. 5, Jan. 1986, pp. 66-78.
4. J.P. Spoto, W.T. Coston, and C.P. Hernandez, "Statistical Integrated Circuit Design and Characterization," IEEE *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-5, No. 1, Jan. 1986, pp. 90-103.
5. D.H. Rosenblatt, W.R. Hitchens, R.E. Anholt, and T.W. Sigmon, "GaAs MESFET Device Dependences on Ion-Implant Tilt and Rotation Angles," IEEE *Electron Device Letters*, Vol. 9, No. 3, March 1988.
6. P. Dobrilla, J.S. Blakemore, A.J. McCamant, K.R. Gleason, and R.Y. Koyama, "GaAs field-effect-transistor properties, as influenced by the local concentrations of midgap native donors and dislocations," *Applied Physics Letters*, Vol. 47, No. 6, 1985.
7. W. Davis "Process Variation Analysis Employing Artificial Neural Networks," *Proc. of the Intern. Joint Confer. on Neural Networks*, Baltimore, Maryland, June 7-11, 1992, vol. II, pp. 260-265.
8. D. Khera, M. E. Zaghloul, L. W. Linholm, C.L. Wilson, "A Neural Network Approach for Classifying Test Structure Results," Proc. of the 1989 *Intern. Confer. on Microelectronic Test Structures*, Vol. 2, No. 1, March 1989, pp. 200-204.
9. M. E. Zaghloul, D. Khera, L. W. Linholm, C. P. Reeve, "A Machine-Learning Classification Approach for IC Manufacturing Control Based on Test Structure Measurements," IEEE *Trans. on Semic. Manuf.*, Vol. 2, No. 2, May 89, pp.47-53.

IEEE

ISCA

INTERNATIONAL CONFERENCE ON CIRCUITS AND SYSTEMS 1994 LONDON, ENGLAND 30 MAY – 2 JUN 1994

Sensitivity Analysis for Minimization of Input Data Dimension for Feedforward Neural Network

Jacek M. Zurada, Aleksander Malinowski, Ian Cloete♦

University of Louisville, Louisville Kentucky 40292, USA

e-mail: jmzura02@ulkyvx.louisville.edu

♦University of Stellenbosch, Stellenbosch, South Africa

ABSTRACT

Multilayer feedforward networks are often used for modeling complex relationships between the data sets. Deleting unimportant data components in the training sets could lead to smaller networks and reduced-size data vectors. This can be achieved by analyzing the total disturbance of network outputs due to perturbed inputs. The search for redundant data components is performed for networks with continuous outputs and is based on the concept in sensitivity of linearized neural networks. The formalized criteria and algorithm for pruning data vectors are formulated and illustrated with examples.

INTRODUCTION

Neural networks are often used to model complex functional relationships between sets of experimental data. This is particularly useful when an analytical model of a process either does not exist or is not known, but when sufficient data is available for embedding relationships existing between two or more data bases into a neural network model. Representative data can be used in such a case to perform supervised training of a suitable neurocomputing architecture. Multilayer feedforward neural networks (MFNN) have been found especially efficient for this purpose [1, 2]. The minimization of redundancy in the training data is, however, an important issue and rather rarely addressed in the technical literature. MFNN considered here are trained using the popular error backpropagation technique in order to perform the feedforward process identification [3].

Let us consider a MFNN with a single hidden layer. The network performs a nonlinear and constrained mapping $o=\Gamma(x)$, where **o** (Kx1), and **x** (Ix1) are output and input vectors, respectively. It is assumed that certain inputs bear none, or little, statistical or deterministic relationships to outputs and input vectors could therefore be compressed. The objective of this study is to reduce the dimensionality of the input vector, **x**, and thus to prune the input data set, so that a smaller network can be utilized as a model of relationship between the data. Initial findings on this subject have been published in [4–6]. This paper introduces a more general and formal approach to reduction of input size of the network. The sensitivity approach can also be used to delete weights which are unimportant for neural network performance as it has been proposed in [7].

SENSITIVITIES TO INPUTS

Let us define the sensitivity of a trained MFNN output, o_k, with respect to its input x_i as

$$S_{x_i}^{o_k} \doteq \frac{\partial o_k}{\partial x_i} \tag{1a}$$

which can be written succinctly as

$$S_{ki} \doteq S_{x_i}^{o_k} \tag{1b}$$

By using the standard notation of an error backpropagation approach [3], the derivative of (1a) can be readily expressed in terms of network weights as follows

$$\frac{\partial o_k}{\partial x_i} = o_k' \sum_{j=1}^{J-1} w_{kj} \frac{\partial y_j}{\partial x_i} \tag{2}$$

where y_j denotes the output of the j-th neuron of the hidden layer, and o_k' is the value of derivative of the activation function o=f(net) at the k-th output neuron. This further yields

$$\frac{\partial o_k}{\partial x_i} = o_k' \sum_{j=1}^{J-1} w_{kj} y_j' v_{ji} \tag{3}$$

where y_j' is the value of derivative of the activation function y=f(net) of the j-th hidden neuron ($y_J'=0$ since the J-th neuron is a dummy one, i.e. it serves as a bias input to the output layer). The sensitivity matrix S (KxI) consisting of entries as in (3) or (1b) can now be expressed using array notation as

$$S = O' \times W \times Y' \times V \tag{4}$$

W (KxJ) and **V** (JxI) are output and hidden layer weight matrices, respectively, and **O'** (KxK) and **Y'** (JxJ) are diagonal matrices defined as follows

$$\begin{aligned} O' &\doteq diag(o_1', o_2', \ldots, o_K') \\ Y' &\doteq diag(y_1', y_2', \ldots, y_J') \end{aligned} \tag{5}$$

Matrix S contains entries S_{ki} which are ratios of absolute increments of output k due to the input i as defined in (1b). This matrix depends only upon the network weights as well as slopes of the activation functions of all neurons. Each training vector $x^{(n)} \in \mathcal{X}$, where $\mathcal{X}=\{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$ denotes the training set, produces different sensitivity matrix $S^{(n)}$ even for a fixed network. This is due to the fact that although weights of a trained network remain constant, the activation values of neurons change across the set of training vectors $x^{(n)}$, n=1, 2, ..., N. This, in turn, produces different diagonal matrices of

*This work was supported in part by the ONR Grant N00014-93-1-0855

derivatives O' and Y', which strongly depend upon the neurons' operating points determined by their activation values.

MEASURES OF SENSITIVITY OVER A TRAINING SET

In order to possibly reduce the dimensionality of input vectors, the sensitivity matrix as in (4) needs to be evaluated over the entire training set $\mathcal{X}$. Let us define the sensitivity matrix for the pattern x_n as $S^{(n)}$. There are several ways to define the overall sensitivity matrix, each relating to the different objective functions which need to be minimized.

The *mean square average sensitivities*, $S_{ki,avg}$, over the set $\mathcal{X}$ can be computed as

$$S_{ki,avg} \doteq \sqrt{\frac{\sum_{n=1}^{N}\left(S_{ki}^{(n)}\right)^2}{N}} \tag{6}$$

Matrix S_{avg} (KxI) is defined as $[S_{avg}]=S_{ki,avg}$. This method of sensitivity averaging is coherent with the goal of network training which minimizes the mean square error over all outputs and all patterns in the set.

The *absolute value average sensitivities*, $S_{ki,abs}$, over the set $\mathcal{X}$ can be computed as

$$S_{ki,abs} \doteq \sqrt{\frac{\sum_{n=1}^{N}\left|S_{ki}^{(n)}\right|}{N}} \tag{7}$$

Matrix S_{abs} (KxI) is defined as $[S_{abs}]=S_{ki,abs}$. Note that summing sensitivities across the training set requires taking their absolute values due to the possibility of cancelations of negative and positive values. This method of averaging may be better than (6) if sensitivities $S_{ki}^{(n)}$, n=1, ..., N, are of disparate values.

The *maximum sensitivities*, $S_{ki,max}$, over the set $\mathcal{X}$ can be computed as

$$S_{ki,max} \doteq \max_{n=1_N}\left\{S_{ki}^{(n)}\right\} \tag{8}$$

Matrix S_{max} (KxI) is defined as $[S_{max}]=S_{ki,max}$. This sensitivity definition allows to prevent pruning inputs which are relevant for the network only in a small percentage of input vectors among the whole training set. However, it can happen that a few fuzzy data entries in a large set can affect entries of sensitivity array by associating fuzziness with additional inputs. Those fuzzy results are masked in such a case by averaging in (6)–(7), and not by (8). Therefore the significance of inputs can be overestimated and some unimportant inputs may remain after reducing the dimension.

Any of the sensitivity measure matrices proposed in (6)–(8) can provide useful information as to the relative significance of each of the inputs in $\mathcal{X}$ to each of the outputs. For the sake of simplicity, however, only the matrix defined in (6) will be used in further discussion. The cumulative statistical information resulting from (6) will be used along with criteria for reducing the number of inputs to the smallest number sufficient for accurate learning. These criteria are formulated in the next section.

CRITERIA FOR PRUNING INPUTS

Inspection of the average sensitivity matrix S_{avg} allows to determine which inputs affect outputs least. A small value of $S_{ki,avg}$ in comparison to others means that for the particular k–th output of the network, the i–th input does not significantly contribute to output k, and may therefore be possibly disregarded. This reasoning and results of experiments allow to formulate the following practical rule: *The sensitivity matrices for a trained neural network can be evaluated for both training and testing data sets; the values of average sensitivity matrix entries can be used for determining the least significant inputs and for reducing the size of network accordingly through pruning unnecessary inputs.*

When one or more of the inputs have relatively small sensitivity in comparison to others, the dimension of neural network can be reduced by dropping them, and smaller–size neural network can be successfully retrained in most cases. The criterion used in this paper for determining which inputs can be pruned is based on the so called largest gap method.

In order to normalize the data relevant for comparison of significance of inputs, the sensitivity matrix defined in (6)–(8) has to be additionally preprocessed. The formulas often used for scaling are given in (9) and map each input into range [0 ; 1] and each output output into range [–1 ; 1]:

$$\hat{x}_i^{(m)} \doteq \frac{x_i^{(m)} - \min_{n=1_N}\left\{x_i^{(n)}\right\}}{\left(\max_{n=1_N}\left\{x_i^{(n)}\right\} - \min_{n=1_N}\left\{x_i^{(n)}\right\}\right)}$$
$$\hat{o}_k^{(m)} \doteq \frac{o_k^{(m)} - \frac{\left(\max_{n=1_N}\left\{o_k^{(n)}\right\} + \min_{n=1_N}\left\{o_k^{(n)}\right\}\right)}{2}}{\left(\max_{n=1_N}\left\{o_k^{(n)}\right\} - \min_{n=1_N}\left\{o_k^{(n)}\right\}\right)} \tag{9}$$

If input and output data scaling (9) was performed before network training, no additional operations on S_{ki} is required and we have

$$\hat{S}_{ki,avg} \doteq S_{ki} \tag{10}$$

Note that the scaling can be performed either on entries of S or S_{avg}. Experiments were performed also for scaling inputs into range [–1 ; 1]. Similar results were achieved for the same learning conditions. The latter scaling seems to fasten the learning convergence while accuracy and relations among sensitivities remain unchanged.

In case when network original inputs and outputs are not scaled to the same level, additional scaling (11) is necessary to allow for accurate comparison among inputs.

$$\hat{S}_{ki,avg} \doteq S_{ki}\frac{\left(\max_{n=1_N}\left\{x_i^{(n)}\right\} - \min_{n=1_N}\left\{x_i^{(n)}\right\}\right)}{\left(\max_{n=1_N}\left\{o_k^{(n)}\right\} - \min_{n=1_N}\left\{o_k^{(n)}\right\}\right)} \tag{11}$$

The significance of i–th input Φ_i across the entire set $\mathcal{X}$ is defined as:

$$\Phi_{i,avg} \doteq \max_{k=1_K}\left\{\hat{S}_{ki,avg}\right\} \tag{12}$$

Φ_{abs} and Φ_{max} can be evaluated similarly to Φ_{abs} defined in (12). In order to distinguish inputs with high and low importance, entries of Φ_i have to be sorted in descending order so

that:

$$\Phi_{i_{m+1}} \geq \Phi_{i_m}\ ,\ m = 1,\ ...,\ I-1 \qquad (13)$$

where i_m is a sequence of sorted input numbers. Let us define the measure of gap as (14)

$$g_{i_m} \doteq \frac{\Phi_{i_m}}{\Phi_{i_{m+1}}} \qquad (14)$$

and then find the largest gap using the formula (15).

$$g_{MAX} \doteq \max_{i_m}\{g_{i_m}\} \quad \text{and} \quad m_{CUT} \doteq m \text{ such that } g_{i_m} = g_{MAX} \qquad (15)$$

If condition (16) is valid, then the found gap between m_{CUT} and m_{CUT+1} is large enough.

$$C g_{MAX} > \max_{i_m \neq i_{m_{CUT}}} \{g_{i_m}\} \qquad (16)$$

Constant C from (16) is chosen arbitrarily within the reasonable range (e.g. C=0.5. The smaller C is, the stronger condition for existence of the acceptable gap is.) All inputs with index $\{i_{m+1}..i_{I-1}\}$ can be pruned with the smallest loss of information to the MFNN.

The gap method can be also applied for comparison among sensitivities of inputs to each output separately. For this purpose, a set containing candidates for pruning can be created for every output. Final pruning is performed by removing these inputs which can be found in every set determined previously for each output independently.

Certainly, S_{avg} can be evaluated meaningfully only for well trained neural networks. Despite this disadvantage, proposed criteria can still save computational effort when initial learning can be performed on smaller, but still representative subset of data. S_{avg} can be evaluated based either on data set used for initial training or on complete data set. Subsequently, newly developed neural network with appropriate inputs can be retrained using the full set of training patterns with reduced dimension.

NUMERICAL EXAMPLES

A series of numerical simulations was performed in order to verify the proposed definitions and the pruning criteria. In the first experiment a training set for a neural network was generated using four inputs $x_1..x_4$ and two outputs o_1 and o_2. Values of outputs were correlated with x_1 and x_2 for o_1, and with x_2 and x_3 for o_2. Input vectors **x** (4x1) were produced using a random number generator. The expected values of vector **d** (2x1) for the output vector **o** (2x1) were evaluated for each **x** using a known relationship **d**=F(**x**) where **d** is the desired (target) output vector for supervised training. The training set $\mathcal{X}$ consisted of N=81 patterns. A neural network with 4 inputs, 2 outputs and 6 hidden neurons (I=5, J=7, K=2) has been trained for the mean square error defined as in (17)

$$MSE \doteq \sqrt{\frac{\sum_{n=1}^{N}\sum_{k=1}^{K}\left(d_k^{(n)} - o_k^{(n)}\right)^2}{N}} \qquad (17)$$

equal 0.001 per input vector. Matrices of sensitivities were subsequently evaluated and S_{avg} produced at the end of training over the entire input data set $\mathcal{X}$.

The changes of sensitivity entries during learning are presented in Fig. 1. It can be seen that an untrained neural network

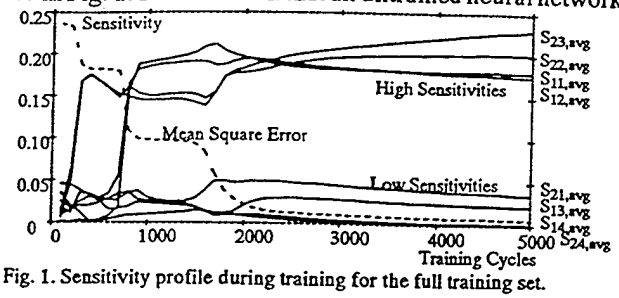


Fig. 1. Sensitivity profile during training for the full training set.

in the example has per average smaller sensitivities than after the training. During the training some of the average sensitivities $S_{ki,avg}$ increase, while others converge towards low values. Final values of sensitivities of the first output offer hints for deleting x_3 and x_4, and these for the second output indicate that x_1 and x_4 could be deleted. The only input which shows up in both sets candidates for deletion is x_4. Therefore, the fourth input to the network can be skipped and its dimension reduced to 3 (I=4).

After deleting x_4 from the learning data set the new network with 3 inputs was trained successfully with the same accuracy. The learning profiles for full and reduced input sets for the same learning conditions are compared in Fig. 2. Not only the

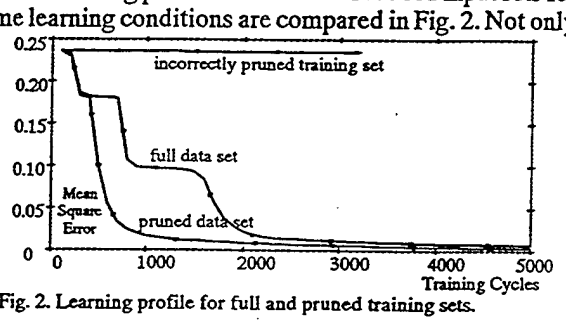


Fig. 2. Learning profile for full and pruned training sets.

network with 3 inputs trains within a smaller number of cycles, but each learning cycle is performed quicker due to the reduced input layer size.

If an input not recommended for pruning is erroneously deleted, the network was found unable to learn the data sets. The mean square error per pattern has remained at the level of approximately 0.25 as it is shown in Fig. 2. The entries of the sensitivity matrix remain at low level as it is shown in Fig. 3.

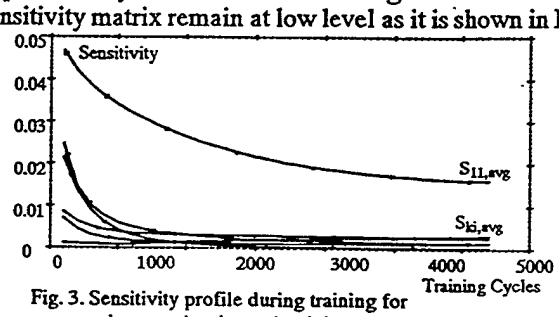


Fig. 3. Sensitivity profile during training for incorrectly trimmed training set.

There may still be some gap between entries, but it cannot be used for pruning because the MFNN has not learned vectors correctly and after input dimension reduction would not be able to learn more accurately. The gap which can be seen in Fig. 3 means that for the insufficient accuracy which was

achieved during the training, only one input could be left in the network without significant deterioration of performance.

The second experiment was performed using larger network and fuzzy data. MFNN had 20 inputs (I=21), 10 hidden neurons (J=26) and 4 outputs (K=4). There were N=500 patterns in the training set and several additional data sets of the same size for network performance evaluation. The network was successfully trained to the MSE error of 0.15. However, due to the fuzziness of the data MSE error for additional sets remained at the level of 0.20.

All outputs were strongly correlated with inputs x_1, x_2, x_3, x_4, x_6, x_8, and x_9. Input x_6 during data generation was multiplied by random numbers, while the influence of x_2 and x_4 on outputs was scaled down to remain small in comparison to other inputs (less than 0.05).

The input importances calculated using formulas (6)–(8) are shown in Fig. 4. Inputs x_2 and x_4 are placed even after sorting

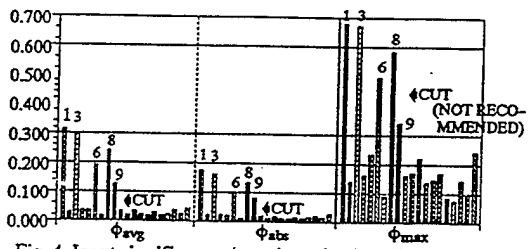


Fig. 4. Input significance ϕ evaluated using different overall sensitivities (6)–(8) and pruning criterion (16).

as less important than some of them which are not correlated at all. This occurred because of their low correlation to outputs, and they can be ignored as well as other not correlated for given MSE error as a final condition for training. The sequence of significance is the same for all proposed methods, however, the size of gaps are different in each case. Value C=0.5 prevents pruning using ϕ_{max} definition. Note that the maximum method does not give the clear clue where to set the level for purging due to fuzziness of the training data.

The result of initial training is shown in Fig. 5. It can be determined from this figure which inputs should remain after pruning. The network performance after pruning is shown in Fig. 6. No additional dimension reduction is possible because no large gap in input importances can be found. The speed of training has increased mostly because of reduction of the MFNN size (input dimension reduced by 4). The necessary number of cycles for training has also decreased, but not so dramatically as in the first experiment.

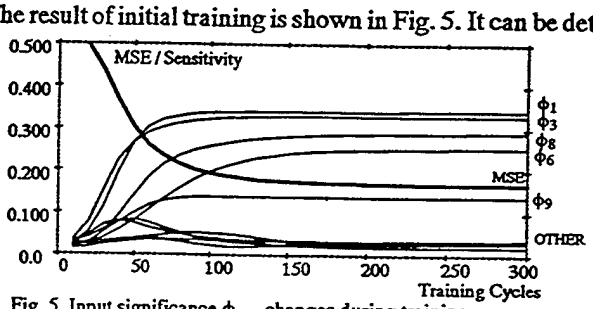


Fig. 5. Input significance ϕ_{avg} changes during training for the full training set.

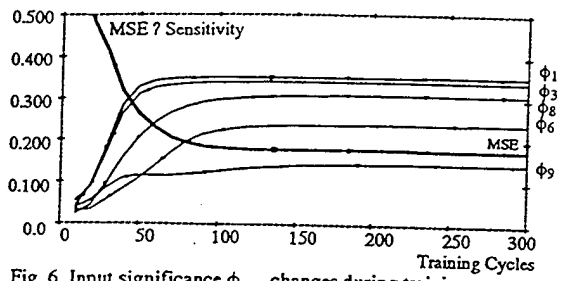


Fig. 6. Input significance ϕ_{avg} changes during training for the pruned training set.

CONCLUSIONS

Using the sensitivity approach for input layer pruning seems particularly useful when network training requires large amount of redundant data. In the first phase, network can be pre-trained until the training error decreases satisfactorily. Then sensitivity matrices can be evaluated and dimension of the input layer possibly reduced. Learning can subsequently be resumed until the training error reduces to acceptable low value. This process can be repeated, however, usually only the first execution yields significant improvement. Numerical experiments indicate that the effort of additional network retraining can be too high in comparison to benefits of further minimization.

Should the redundancy in training data vectors exist, the proposed approach based on the average sensitivity matrices for input data pruning allows for more efficient training. This can be achieved at a relatively low computational cost and based on heuristic data pruning criteria outlined in the paper. The approach can be combined with other improved training strategies such as increased complexity training [5]. Extension of the proposed sensitivity-based input pruning concept beyond continuous output values seems desirable for case of networks with binary outputs such as classifiers and other binary encoders.

BIBLIOGRAPHY

[1] K. M. Hornik, M. Stinchcombe, H. White, "Multilayer Feedforward Networks Are Universal Approximators", *Neural Networks*, vol. 2, 1989, pp.359–366.

[2] K. I. Funanashi, "On the Approximate Realization of Continuous Mappings by Neural Networks", *Neural Networks*, vol. 2, 1989, pp. 183–192.

[3] J. M. Zurada, *Introduction to Artificial Neural Systems*, West Publishing Company, St. Paul, Minn., 1992.

[4] L. Fu, T. Chen, "Sensitivity Analysis for Input Vector in Multilayer Feedforward Neural Networks", *Proc. of IEEE International Conference on Neural Networks, San Francisco*, CA, March 28–April 1, 1993, vol. 1, pp.215–218.

[5] I. Cloete, J. Ludik, "Increased Complexity Training", *Proc. of IWANN'93*, Sitges, Spain, June 9–11, 1993.

[6] J. M. Zurada, A. Malinowski, I. Cloete, "Sensitivity Analysis for Pruning of Training Data in Feedforward Neural Networks", *Proc. of First Australian and New Zeland Conference on Intelligent Information Systems*, Perth, Western Australia, December 1–3, 1993, pp. 288–292.

[7] E. D. Karnin, "A Simple Procedure for Pruning Back-Propagation Trained Neural Networks", *IEEE Trans. on Neural Networks*, vol. 1, No. 2, June 1990.

IEEE

ISCA

E INTERNATIONAL CONFERENCE ON CIRCUITS AND SYSTEMS 1994 LONDON, ENGLAND 30 MAY - 2 JUN 1994

Minimal Training Set Size Estimation For Neural Network-based Function Approximation

Aleksander Malinowski, Jacek M. Zurada, Peter B. Aronhime

Department of Electrical Engineering,

University of Louisville, Louisville, KY 40292, USA

e-mail: a0mali01@starbase.spd.louisville.edu, jmzura02@ulkyvx.louisville.edu

ABSTRACT

A new approach to the problem of n-dimensional continuous and sampled-data function approximation using two-layer neural network is presented. The generalized Nyquist theorem is introduced to solve for the optimum number of training examples in n-dimensional input space. Choosing the smallest but still sufficient set of training vectors results in the reduced learning time for the network. Analytical formulas and algorithm for training set size reduction are developed and illustrated by two-dimensional data examples.

INTRODUCTION

Neural networks as approximators of input/output relationships among many variables are currently under intense investigation, with emphasis on their approximation capabilities and performance for different network architectures and learning conditions [1]. Generalization and approximation without specifying equations and coefficients are indeed very promising features of neural networks, particularly in cases where the unknown model describing a plant is complex and training data abundant. Due to their ability of generalization, multilayer feedforward neural networks (MFNN) are commonly used for this purpose [2], [3].

Papers on the subject of approximation using MFNN were published lately [4], [5]; however, they do not focus on the size minimization of the training data set. Preliminary heuristic solutions to the training data set minimization problem along with single variable function examples have been published in [6] and [7]. The sampled-data function case based on results for continuous function are reported in [8].

This paper generalizes an analytical approach for multidimensional input space for continuous function case. An analytical approach based on the sampling theorem is applied to function approximation using MFNN. An analogy between time-dependent functions and single variable functions is made and then expanded into multidimensional space. In this way, an estimated minimum sampling frequency for MFNN training can be found for a required approximation accuracy. The results can be used for reducing the number of data entries required in a neural network training set. The experimental part of the paper illustrates the use of this method with a simple example.

SAMPLE DATA THEOREM FOR SAMPLING RATE EVALUATION

The well-known Sampling Theorem (non-periodic signal case) states that : *A function f(x) which contains no frequency components greater than f_0 Hz is uniquely determined by the values of f(x) at any set of sampling points spaced at most $1/(2f_0)$ seconds apart [9], [10].* Sampling rates defined for time signals can be extended to other independent variables so that the generalized theorem for function approximation can be obtained. Each dimension of the transform will then correspond to one dimension of the original domain.

Obviously, sampling with a certain frequency is needed to restore the signal from the samples taken. However, the theorem refers to the ideal case where the input signal has a finite high frequency boundary so that it can be accurately restored from samples taken using the inverse Fourier transform. Real-life signals are not band-limited, and other mechanisms than Fourier transforms are used for restoring them. This paper focuses on analysis of approximation conditions for particular network structures and neuron activation functions even though no theorem is available for approximation of signals of infinite bandwidth. The developed algorithm is based on the assumption that only a certain fraction of information about the function is necessary for the approximation with given required accuracy.

Let the continuous function to be approximated be given as f(x),

$$f(x) : \mathfrak{D} \rightarrow \Re, \text{ where } \mathfrak{D} \subset \Re^N$$

$$\mathfrak{D} \doteq (x_{MIN_1} \dots x_{MAX_1}) \times (x_{MIN_2} \dots x_{MAX_2}) \times \dots \times (x_{MIN_N} \dots x_{MAX_N}) \quad (1)$$

$$x \doteq [x_1, x_2, \dots, x_N]$$

and let B_i be the range of the i-th variable x_i:

$$B_i \doteq x_{MAX_i} - x_{MIN_i} \quad (2)$$

The multidimensional Fourier transform of f(x) is defined in the following way

$$F(\Omega) \doteq \frac{1}{(2\pi j)^N} \int_{x_{MIN_1}}^{x_{MAX_1}} \int_{x_{MIN_2}}^{x_{MAX_2}} \dots \int_{x_{MIN_N}}^{x_{MAX_N}} f(x_1, x_2, \dots, x_N) \, e^{2\pi j x_1 \omega_1} e^{2\pi j x_2 \omega_2} \dots e^{2\pi j x_N \omega_N} \, dx_1 dx_2 \dots dx_N \quad (3)$$

where $\Omega \doteq [\omega_1, \omega_2, \dots, \omega_N]$

The criterion for minimum sampling frequency estimation can be formulated in a number of ways. First, the basic formula for optimization should be defined as a norm evaluating the information density at particular frequencies. This norm as defined in (4) has a meaning of generalized energy density.

$$E_d(\Omega) \doteq |F(\Omega)|^2 \quad (4)$$

We also use function (5) for evaluating the amount of information enclosed by the frequency band Ω. In case of a multidimensional band-limited function, the energy, $E(\Omega)$, can be computed by integrating the generalized energy density (4) in the frequency domain in spherical [11], or more precisely,

ellipsoidal coordinates within an N-dimensional ellipsoid. For example, in the simplest case assuming that function F has isotropic properties in each dimension ($\omega=\omega_1=\omega_2=...=\omega_N$), this yields

$$E(\omega) \doteq \int_{r=0}^{1} \int_{\phi_1=0}^{2\pi} \int_{\phi_2=0}^{2\pi} \cdots \int_{\phi_{N-1}=0}^{2\pi} |F(\omega r \cos\phi_1 \cos\phi_2 \ldots \cos\phi_{N-1}, \quad (5)$$

$$\omega r \sin\phi_1 \cos\phi_2 \ldots \cos\phi_{N-1}, \ldots, \omega r \sin\phi_1 \sin\phi_2 \ldots \sin\phi_{N-1})|^2 J(.) d\phi_1 d\phi_2 \ldots d\phi_{N-1} dr$$

where $J(r, \phi_1, \phi_2, ..., \phi_N)$ is a term resulting from the change of the integration coordinates from cubic to spheric [12]. However, in general it cannot be assumed that the approximated function will have isotropic properties. It may then be reasonable to choose smaller sampling densities in some dimension. The function (5) becomes more complex due to different boundaries in each dimension

$$E(\Omega) \doteq \int_{r=0}^{1} \int_{\phi_1=0}^{2\pi} \int_{\phi_2=0}^{2\pi} \cdots \int_{\phi_{N-1}=0}^{2\pi} |F(\omega_1 r \cos\phi_1 \cos\phi_2 \ldots \cos\phi_{N-1}, \quad (6)$$

$$\omega_2 r \sin\phi_1 \cos\phi_2 \ldots \cos\phi_{N-1}, \ldots, \omega_N r \sin\phi_1 \sin\phi_2 \ldots \sin\phi_{N-1})|^2 J(.) d\phi_1 d\phi_2 \ldots d\phi_{N-1} dr$$

where $J(r, \Omega, \phi_1, \phi_2, ..., \phi_N)$ is a term obtained as previously from the change of the integration coordinates from cubic to spheric.

Let C_{INFO} called the *information rate factor* be the fraction describing the required minimum energy content of the signal sampled with frequency Ω, divided by the energy E_{TOT} of the original function (or function sampled with very high frequency). The information rate factor is a theoretical measure of the information amount needed to approximate a function with required accuracy. Function f(x) needs to be sampled with frequency Ω satisfying condition (7).

$$\frac{E(\Omega)}{E(\Omega_{MAX})} \geq C_{INFO} \quad (7)$$

where the frequency $\Omega_{MAX} \doteq [\omega_1, \omega_2, \ldots, \omega_N]_{MAX}$ is high enough, so that

$$|F(\Omega_{MAX})|^2 \approx 0 \text{ and } E(\Omega_{MAX}) \approx E_{TOT} \quad (8)$$

Let us now express the total number of samples in the training set. The number of samples taken per dimension, M_{Li}, is equal to

$$M_{Li}(\omega_i) \doteq (2B_i\omega_i + 1) \quad (9)$$

The total number of sampled data, M_L, can be expressed as

$$M_L(\omega_1, \omega_2, \ldots, \omega_N) \doteq \prod_{i=1}^{N} M_{Li}(\omega_i) \quad (10)$$

The objective is to search among vectors Ω which satisfy the condition (7) and minimize the value of M_L defined in (10). The vector $\Omega_{OPT} \doteq [\omega_1, \omega_2, \ldots, \omega_N]_{OPT}$ which is the solution to the given optimization problem contains the minimum sufficient sampling frequencies in the new training data set. The final sampling interval, Δx_i, is different for each dimension depending on the chosen frequency ω_i.

$$\Delta x_i = \frac{1}{2\omega_{OPTi}} \quad (11)$$

ALGORITHM FOR ESTIMATION OF THE INFORMATION RATE FACTOR

The last constant which has to be estimated is C_{INFO} from equation (7). This constant defines the minimum sufficient amount of information after narrowing the function frequency spectrum in terms of its energy.

Let us define mean square average error of approximation, MSE, as

$$MSE \doteq \sqrt{\frac{\sum_{p=1}^{P} \left(d^{(p)} - o^{(p)}\right)^2}{P}} \quad (12)$$

where P is the number of data entries, $d^{(p)}$ is the known function value for input vector $x^{(p)}$, and $o^{(p)}$ is the value computed by the neural network. Error defined in the sense of (12) is based on the energetic distance between the original and the approximated function and is also useful for expressing the training termination condition. In order to determine the value of MSE, it is necessary to know the average power of the original function, P_{TOT}. P_{TOT} can be calculated from the Fourier power spectrum in the frequency domain either in the x domain using formula (13) for the continuous case, or (14) in case of discrete data.

$$P_{TOT} = \frac{\int_{x_{MIN1}}^{x_{MAX1}} \int_{x_{MIN2}}^{x_{MAX2}} \cdots \int_{x_{MINN}}^{x_{MAXN}} f(x_1, x_2, \ldots . x_N)^2 dx_1 dx_2 \ldots dx_N}{\prod_{i=1}^{N} B_i} \quad (13)$$

$$P_{TOT} = \frac{\sum_{p=1}^{P} \left(d^{(p)}\right)^2}{P} \quad (14)$$

The terms of integration in equation (13) or summation in (14) evaluate the same energy, which is used in the denominators of condition (7).

The required approximation accuracy Ψ links together MSE and P_{TOT}.

$$\Psi = \frac{MSE}{\sqrt{P_{TOT}}} \quad (15)$$

The reason for normalizing the variables defined in (12) and (14) is to allow easy comparison of results of training and to evaluate the quality of the approximation without considering the number of patterns used each time. Finally we have the relation

$$C_{INFO} = 1 - \Psi \quad (16)$$

which links the final condition for training (12) with equations (7).

ALGORITHM FOR FINDING MINIMUM SUFFICIENT SAMPLING RATES

The following algorithm for finding the minimum sufficient sampling rates is based on the theoretical assumptions presented above:

STEP 1. Compute FFT of the function. $f(x) \Rightarrow F(\Omega)$
Note that the upper boundary of the Fourier transform in real life cannot be infinite. Condition (8) has to be satisfied.

STEP 2. If the frequency response is not small enough at the highest frequency in comparison to lower ones, i.e. (8) is not satisfied, the first step must be repeated for 5 to 10 times more frequent sampling in the appropriate dimensions.

STEP 3. Complete the information measurement function $E(\Omega)$ as in (6) and normalize it so that its maximum is equal to 1. $F(\Omega) \Rightarrow E(\Omega)$

STEP 4. Evaluate C_{INFO} and MSE for particular requirements of approximation accuracy Ψ using (16).

STEP 5. Check for what frequencies function $E(\Omega)$ from (6) reaches the levels evaluated in the STEP 4.
$\{\Omega\} \Leftarrow \{\Omega : E(\Omega) \geq C_{INFO}\, E_{TOT}\}$

STEP 6. Solve for l which produces minimum M_L in (10) using the set of $\{\Omega\}$ satisfying condition from STEP 5.
$\Omega_{OPT} \Leftarrow \Omega : M_L(\Omega) = \min\{M_L(\Omega)\}$ over all $\Omega \in \{\Omega\}$.

STEP 7. Choose the sampling steps slightly higher than those corresponding closely to frequencies computed in STEP 6.

STOP.

If there are problems with convergence, the MFNN architecture should be changed or the learning constants decreased. If the approximation error after completed training is excessive, sampling steps should be decreased by choosing more severe constraints than given in (16).

EXPERIMENTAL RESULTS

A series of experiments were conducted to confirm the theoretical results and to test the heuristic guidelines proposed for sampling rates. A MFNN with one hidden and one input layer has been used for single-variable function approximation. The experiments were performed for approximating one- [6-7], and two-dimensional functions using neural network architectures with different numbers of hidden neurons and for different final error conditions which provide more insight into the practical use of the method.

To prevent saturation of neurons and to provide similar conditions for each test, the scaling of input data was performed so that normalized input variables varied form 0 to 1. Since bipolar continuous neurons were used, it was necessary to scale functions to be approximated to the range between -1 and 1. Standard and modified (lambda learning [13]) error backpropagation algorithms were used for learning. Functions were first sampled with very high density for evaluating the discrete Fourier transform and for evaluating the approximation accuracy after completing training. Before each training with a new sampling step, a new learning data set was created and network weights were initialized once again. Each training was performed until it reached the MSE error set previously.

Theoretical estimations were compared with frequencies obtained from experiments. As anticipated, **there exists an optimal number of learning points for a given approximation accuracy. This number of points can be evaluated from the integral of its Fourier transform (6).**

In the following example the discrete number of samples per dimension is used instead of continuous frequency to make the theoretical results consistent with obtained from experiments. $l=[l_1, l_2, ..., l_N]$ is the discrete frequency in DFT domain and corresponds to continuous frequency Ω in the following way:

$$\omega_i \doteq \frac{l_i}{B_i} \qquad (17)$$

Fig. 1a shows an example function used for approximation. Fig. 1b depicts one quadrant of the frequency domain of the

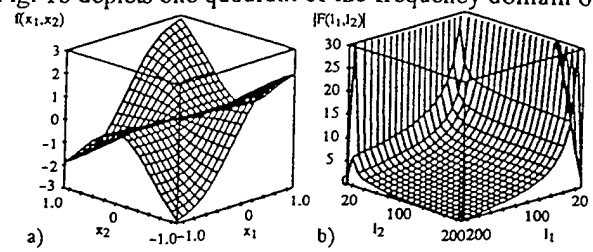


Fig. 1. Approximated function f(x) as in (18) (a), and its Fourier transform (b). P_{TOT}=1.12.

magnitude of its discrete Fourier transform. Approximated function was given by formula (18).

$$f(x) = \frac{x_1^2 x_2 + 5x_1 x_2^2}{x_1^2 + x_2^2 + 0.1}, \quad x_1 = -1..1,\ x_2 = -1..1 \qquad (18)$$

The normalized energy function of f(x) given by (18) covered by the frequency l is shown in Fig. 2. The normalized energy

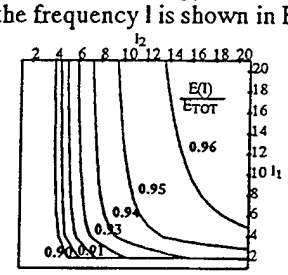


Fig. 2. Amount of energy $E_{NORM}(l)$ covered by bounded frequency spectrum. Profiles for E_{NORM}>0.90.

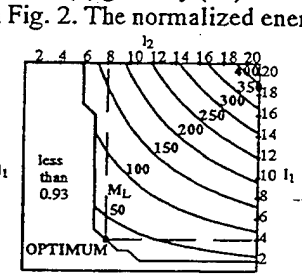


Fig. 3. Number of samples in learning set in area of normalized $E(l_1,l_2)>C_{INFO}$. Profiles for $C_{INFO}\geq 0.93$.

has been computed from the left side of condition (7).

The average power, P_{TOT}, was calculated for the given function using the formula (14); it is of value P_{TOT}=1.12. A MFNNs with 2 inputs, and 20 hidden neurons was trained to the error MSE=0.08 as defined by the formula (12). Ψ and C_{INFO} were then calculated using formulas (15) and (16), giving the values: Ψ=0.07, and C_{INFO}=0.93. The optimum number of samples for each dimension has been found from Fig. 3 by finding the minimum of M_L over frequencies satisfying condition (7) which gives the contour line bounding the domain of solution. This figure shows the contours for the number of data entries in the training set, M_L, for different l_1 and l_2 which satisfy the condition (7). The minimum of M_L can be seen at l_1=4 and l_2=8. It can be evaluated from equation (10). This corresponds to 9 samples for variable x_1 and 17 samples for variable x_2.

MFNNs with architectures described above were trained for different numbers of samples in each dimension to verify the theoretical results. The results of training are illustrated in Figs. 4-8. Fig. 4 and Fig. 5 show the quality of training in terms of MSE and the maximum error achieved during approximation verification based on a very large testing set (500x500 samples). It can be seen that the error decreases dramatically when $l_1\geq 2$ and $l_2\geq 3$. This corresponds to 5 and 7, respectively, samples per dimensions.

Fig. 6 shows the number of training steps required for the learning process, while Fig. 7 shows only the number of itera-

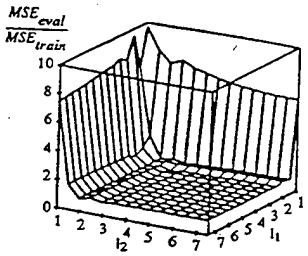


Fig. 4. Neural network performance (MSE) after training in versus sampling frequencies. MSE=0.08.

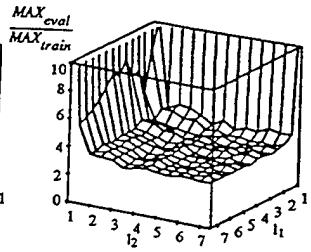


Fig. 5. Neural network performance (MAX) after training versus sampling frequencies l. MSE=0.08.

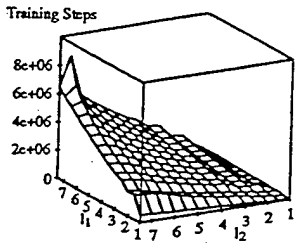


Fig. 6. Number of training steps versus sampling frequencies.

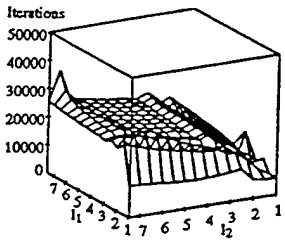


Fig. 7. Number of iterations versus sampling frequencies.

tions (cycles). After achieving certain frequencies of sampling the function to build a training set, the number of iterations does not increase or increases only slowly, while the overall number of steps still increases due to the growing number of data entries.

Fig. 8 summarizes the computational experiment. The num-

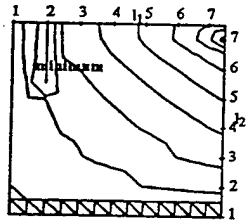


Fig. 8. Number of training steps versus sampling frequencies for area of sufficient learning;

ber of iterations for the sampling frequencies l providing accurate learning is displayed. Local minima can be observed for the frequencies l_1=2 and l_2=5 for the first MFNN and for l_1=2 and l_2=6 for the second. This corresponds to five and eleven, and five and thirteen samples per dimension, respectively. This is in agreement with four and eight samples per appropriate dimension. The obtained results are close enough to those evaluated previously using the derived theoretical algorithm and displayed in Fig. 3.

VI CONCLUSIONS

The results of the computational experiments and theoretical studies for continuous function case show that the generalized sampling theorem can be applied to the approximation problem using neural networks. The least possible, but still large enough for the sake of accuracy data set should be selected, and then other network parameters can be found through training [14–17]. Our results indicate that the least in size training sets can be found for any multidimensional functions basing on the knowledge of its frequency powers spectrum. Successfully trained neural networks capable of accurate approximation can be trained using a training set with sampling density of evaluated rate.

VII REFERENCES

[1] E. J. Hartman, J. D. Keeler, J. M. Kowalski, "Layered Neural Networks with Gaussian Hidden Units as Universal Approximations," *Neural Computation II*, 1990, pp. 210–215.

[2] S. Shekhar, M. B. Amin, "Generalization by Neural Networks," *IEEE Trans. on Knowledge and Data Eng.*, vol. 4, no. 2, April 1992, pp. 177–185.

[3] Y. Shin, J. Ghost, "Approximation of Multivariate Functions Using Ridge Polynomial Networks," *Proc. of International Joint Conference of Neural Networks*, Baltimore, Maryland, June 7–11, 1992, vol. 2, pp. 380 – 385.

[4] R. M. Sanner, J-J. E. Slotine, "Gaussian Network for Direct Adaptive Control," *IEEE Trans. on Neural Networks*, vol. 3, no. 6, November 1992, pp. 837–863.

[5] C.-H Choi, J. Y. Choi, "Construction of Neural Networks for Piecewise Approximation of Continuous Functions," *Proc. of the IEEE International Conference on Neural Networks*, San Francisco, CA, March 28–31, 1993, vol. 1, pp.428–433.

[6] J. M. Zurada, A. Malinowski, "Sampling Rate for Information Encoding Using Multilayer Neural Networks," *Proc. of International Joint Conference on Neural Networks*, Nagoya, Japan, October 25–29, 1993.

[7] A. Malinowski, J. M. Zurada, "Minimal Training Set Size for Neural Network-based Function Encoding," *Proc. of the Conference Artificial Neural Networks in Engineering*, St. Louis, Missouri, November 14–17, 1993, pp. 149–154.

[8] A. Malinowski, J. M. Zurada, "Minimal Training Conditions For Sampled-Data Function Encoding using Feedforward Neural Network", Submitted to The World Congress on Neural Networks, San Diego, CA, 1994.

[9] A. D. Poularikas, S. Seely, *Elements of Signals and Systems*, PWS-KENT Publishing Company, Boston, 1992.

[10] C. L. Philips, H. T. Nagle, *Digital Control Systems - Analysis and Design*, Prentice Hall Inc., New Jersey 1990.

[11] H. C. Andrews, W. K. Pratt, K. Caspari, *Computer Techniques in Image Processing*, Academic Press Inc. Ltd., London, 1970, pp. 157–161.

[12] T. M. Apostol, Mathematical Analysis, *A Modern Approach to Advanced Calculus*, Addison-Wesley Publishing Company, London 1957, pp. 270–275.

[13] J. M. Zurada, "Lambda Learning Rule," *Proc. of the IEEE International Conference on Neural Networks*, San Francisco, CA, March 28–31, 1993, pp. 1808–1811.

[14] S.-C. Huang, Y.-F. Huang, "Bounds on the Number of Hidden Neurons in Multilayer Perceptrons," *IEEE Trans. on Neural Networks*, vol. 2, no. 1, January 1991, pp. 47–55.

[15] W. Fakhr, M. Kamel, M. I. Elmasry, "Probability of Error, Maximum Mutual Information, and Size Minimization of Neural Networks," *Proc. of the International Joint Conference of Neural Networks*, Baltimore, Maryland, June 7–11, 1992, vol. 4, pp. 901–906.

[16] J. M. Zurada, *Introduction to Artificial Neural Systems*, West Publishing Company, St. Paul, Minn., 1992.

[17] J. Hertz, A. Krogh, R. G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing Company, 1990.

Appl. Math. and Comp. Sci., 1994, vol.4, No.3, 281–307

MULTILAYER PERCEPTRON NETWORKS: SELECTED ASPECTS OF TRAINING OPTIMIZATION†

JACEK M. ŻURADA*, ALEKSANDER MALINOWSKI*

Training of Multilayer Perceptron Neural Networks using the popular error back propagation method can be modified and its performance improved. The modified original generalized delta learning rule has been found to considerably enhance the learning process. In addition, input layer size can be reducible through evaluation of the network sensitivity over the training/test data set. Minimum set size estimation based on the sampling theorem can also be performed to determine the optimum number of training patterns.

1. Introduction

Multilayer Perceptron Neural Networks (MLPNN) account for the majority of today's applications of neural networks. They are used to express functional relationships between the sets of experimental data describing process identification and modelling of function, approximation, classification, prediction, one-step ahead control, and other tasks. MLPNN's ability to model experimental relationships and to embed knowledge from data into networks is due to the stochastic approximation based on learning within a rather flexible architecture. Although a layered architecture of an MLPNN with a single hidden layer fits into modelling of numerous tasks, a number of questions as to how to optimize the training and the network itself remain unanswered.

One important issue concerns the optimality of the generalized delta learning rule (EBPT). Although the EBPT algorithm has been widely used and hundreds of technical reports illustrate its successful applications, the lambda learning rule often offers a considerable improvement of learning. The paper outlines the generalized lambda learning algorithm for layered networks. It also focuses on visualization of learning and draws comparisons between the two learning approaches.

Minimization of redundancy in the training data is another important issue. When certain inputs bear none, or little, statistical or deterministic relationships to outputs, input vectors can be compressed. This makes it possible to reduce the dimensionality of the input vector, $\boldsymbol{x}$, through pruning of the input data set, so that a smaller network can be utilized as a model of relationship between the data. Initial findings on this subject have been published in (Cloete and Ludik, 1993; Fu and Chen, 1993; Żurada and Malinowski, 1993). This paper introduces a more formal approach to reduction of input size of the network.

† This work was supported in part by the ONR Grant N00014-93-1-0855
* Department of Electrical Engineering, University of Louisville, Louisville, Kentucky 40292, USA

A new approach to the problem of n-dimensional function approximation using MLPNN is also discussed. The generalized Nyquist theorem is used to look for the optimum number of learning patterns in n-dimensional input space. Choosing the smallest but still sufficient set of training vectors results in a reduced number of hidden neurons and learning time for the network. Analytical formulae and an algorithm for training set size reduction are developed and illustrated by two-dimensional data examples.

2. Lambda Learning Rule for MLPNN

Assume that a non-augmented input vector x has $n-1$ components. Lambda learning rule involves expansion of the learning space to $n+1$ dimensions. In this rule, in addition to weight learning, the steepness of the activation function undergoes adjustment in the negative gradient direction. The first observation of this useful property has been made in (Tawel, 1989) and, independently, in (Movellan, 1987). A number of simulations have confirmed that the method not only accelerates learning, but also improves generalizations and introduces a better participation of nodes in representing the input (Kruschke and Movellan, 1991).

Using the customary expression for error between the desired output value d and the actual one o, i.e.

$$E(\omega,\lambda)=\frac{1}{2}|d-o(\omega,\lambda)|^2 \tag{1}$$

we obtain the following weight and λ adjustments for a single neuron learning

$$\Delta\omega_i=\eta_1\frac{\partial E}{\partial\omega_i}=-\eta_1(d-o)f'(\lambda net)\lambda x_i \tag{2}$$

$$\Delta\lambda=\eta_2\frac{\partial E}{\partial\lambda}=-\eta_2(d-o)f'(\lambda net)net \tag{3}$$

where $f'(net)=o(1-o)$ and the activation value and function are, respectively,

$$net=w^Ty$$

$$f(\lambda,net)=(1+e^{-\lambda net})^{-1} \tag{4}$$

and η_1, η_2 are positive learning constants usually selected of arbitrary small values. Noticeably, expression (2) coincides exactly with the delta learning rule. Additional weight adjustment as in (3) is the essence of the lambda rule. It expresses such block changes of the value net that although the individual weights remain invariant, the overall impact of learning typically increases, since the adaptive gain factor in the exponent of (4) is suitably adjusted. This feature may be of particular importance when all weights are considerably too small or too large, and/or the neuron learning progresses slowly. Intuitively, (3) corresponds to a separate and independent weight scaling step, when all the multiplicative weights are adjusted up or down in a block-wise fashion in a direction which minimizes the current training error.

The lambda learning rule can now be easily generalized for a single or double layer learning. Assume that the output layer of a two-layer network undergoes the training. Here, steepness coefficients λ_i, $i=1,2,\ldots,K$, undergo adjustments in addition to the individual weights. With reference to Fig. 1, the adjusted learning variables can be obtained from (2), (3) as below (primes denote updated values)

$$\omega'_{kj}=\omega_{kj}+\eta_1(d_k-o_k)f'(net_k)\lambda_k x_j \tag{5}$$

$$\lambda'_k=\lambda_k+\eta_2(d_k-o_k)f'(net_k)net_k \tag{6}$$

where $j=1,2,\ldots,J$, and k is the neuron number, weights of which undergo adjustments. Understandably, k must be incremented from 1 through K in order for the entire layer to update both its weights and λ values for a single input pattern.

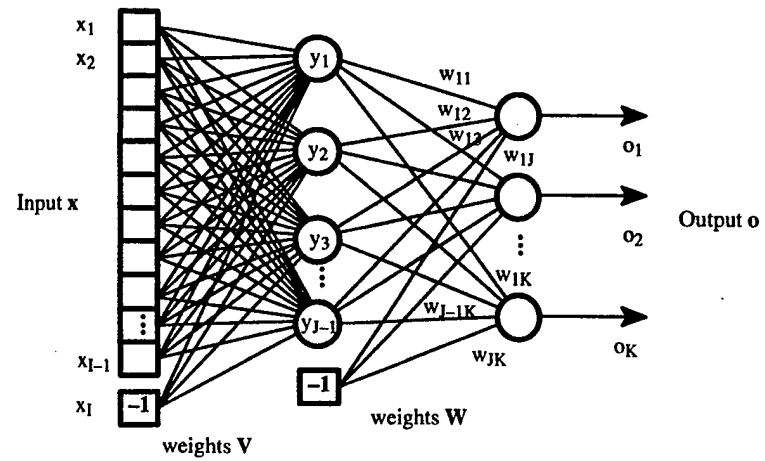


Fig. 1. MLPNN for illustration of lambda and generalized lambda learning.

The generalization below refers to the hidden layer learning, specifically, to learning of weights v_{ji} using the notation of Fig. 1. Obviously, the learning of the j-th neuron is performed such that v_{ji} and λ_j, $i=1,2,\ldots,I$, and $j=1,2,\ldots,J$, are adapted.

Using the following overall output error definition

$$E=\frac{1}{2}\sum_{k=1}^{K}(d_k-o_k)^2 \tag{7}$$

we obtain for weights and steepness coefficients the adjustments

$$\frac{\partial E}{\partial v_{ji}}=\frac{\partial E}{\partial net_j}\frac{\partial net_j}{\partial v_{ji}} \tag{8}$$

$$\frac{\partial E}{\partial \lambda_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial \lambda_j} \tag{9}$$

Further rearrangements using standard notation (Żurada, 1992) and based on derivations detailed in (Żurada, 1993) lead to the following weight adjustments in the generalized lambda learning rule

$$\Delta \nu_{ji} = \eta_1 f'(net_j) z_i \lambda_j \sum_{k=1}^{K} \delta_{ok} \omega_{kj} \tag{10}$$

$$\Delta \lambda_j = \eta_2 f'(net_j) net_j \sum_{k=1}^{K} \delta_{ok} net_k \tag{11}$$

where $\delta_{ok} = (d_k - o_k) f'(net_k)$.

3. MLPNN Training Algorithm with Lambda Learning

The complete learning algorithm for the network of Fig. 1 is given below:

BEGIN: Given are training pairs of vectors of inputs and desired outputs $\{\boldsymbol{z}_1, \boldsymbol{d}_1, \boldsymbol{z}_2, \boldsymbol{d}_2, \ldots, \boldsymbol{z}_p, \boldsymbol{d}_p\}$ where $\boldsymbol{z}_i$ is $(I \times 1)$, $\boldsymbol{d}_i$ is $(K \times 1)$ and $i = 1, 2, \ldots, P$. Note that the I-th component of each $\boldsymbol{z}_i$ is of value -1 since input vectors are augmented. Size $J-1$ of the hidden layer having outputs $\boldsymbol{y}$ is selected. Note that the J-th component of $\boldsymbol{y}$ is of value -1, since hidden layer outputs are also augmented; $\boldsymbol{y}$ is $(J \times 1)$ and $\boldsymbol{o}$ is $(K \times 1)$.

STEP 1: η_1, $\eta_2 > 0$, acceptable training error E_{max} chosen. Weights $\boldsymbol{W}$ and $\boldsymbol{V}$ are initialized at small random values; $\boldsymbol{W}$ is $(K \times J)$, V is $(J \times I)$, $q := 1$, $p := 1$, $E := 0$

STEP 2: Training step starts here. Input is presented and the layers' outputs are computed as in (1):

$$z := z_p, \qquad d := d_p, \qquad y_j := f(\boldsymbol{v}_j^T \boldsymbol{z}) \text{ for } j = 1, 2, \ldots, J$$

where $\boldsymbol{v}_j$, a column vector, is the j-th row of $\boldsymbol{V}$, and

$$o_k := f(\boldsymbol{w}_k^T \boldsymbol{y}) \qquad \text{for} \quad k = 1, 2, \ldots, K$$

where $\boldsymbol{w}_k$, a column vector, is the k-th row of $\boldsymbol{W}$.

STEP 3: Error value is computed: $E := 0.5(d_k - o_k)^2 + E$, for $k = 1, 2, \ldots, K$.

STEP 4: Error signal vectors δ_o and δ_y of both layers are computed. Vector δ_o is $(K \times 1)$, δ_y is $(J \times 1)$.

The error signal terms of the output and hidden layers in this step are, respectively

$$\delta_{ok} := (d_k - o_k)(1 - o_k) o_k \qquad \text{for} \quad k = 1, 2, \ldots, K$$

$$\delta_{yj} := y_j(1 - y_j) \sum \delta_o w_{kj} \quad \text{for} \quad j = 1, 2, \ldots, J, \quad \text{over } k = 1, 2, \ldots, K$$

STEP 5: Output layer weights and gains are adjusted:

$$w_{kj} := w_{kj} + \eta_1 \delta_{ok} \lambda_j$$

$$\lambda_k := \lambda_k + \eta_2 \delta_{ok} net_k \quad \text{for} \quad k = 1, 2, \ldots, K \quad \text{and} \quad j = 1, 2, \ldots, J$$

STEP 6: Hidden layer weights and gains are adjusted:

$$v_{ji} := v_{ji} + \eta_1 \delta_{yj} \lambda_j z_i$$

$$\lambda_j := \lambda_j + \eta_2 \delta_{yj} net_j \quad \text{for} \quad j = 1, 2, \ldots, J \quad \text{and} \quad i = 1, 2, \ldots, I$$

STEP 7: If $p < P$, then $p := p + 1$, $q := q + 1$, and go to Step 2; otherwise go to Step 8.

STEP 8: The training cycle is computed. For $E < E_{max}$ terminate the training session. Output weights $\boldsymbol{W}$, $\boldsymbol{V}$, and the cycle counter q, as well as the error E. If $E > E_{max}$, then $E := 0$, $p := 1$, and initiate the new training cycle by going to Step 2.

END.

4. Experimental Results

Simulation of learning efficiency using the lambda learning rule has been tested on a number of cases, including a bit-map classifier. Although the lambda learning rule-based algorithm is numerically about as complex as the classical EBPT, it can offer an improvement in the efficiency of training. Since the improvement comes at about no additional cost, the algorithm seems to be a promising one and likely to become important. The same learning constants and initial weights have been used for comparison of both algorithms (simulations are made for bipolar neurons' characteristics).

Learning using the EBPT and the lambda learning rule have been simulated and compared for the XOR problem. Figure 2 shows two typical learning profiles for the EBPT and lambda learning rule, and it indicates faster learning for the method employing adaptive gain. Two hidden-layer neurons have been used in this experiment. Figure 3 illustrates gain variations during training and corresponds to the bottom error curve of Fig. 2. It can be seen that the output neuron's gain changes sign during learning and its gain variations are substantially correlated with an associated error curve of Fig. 2.

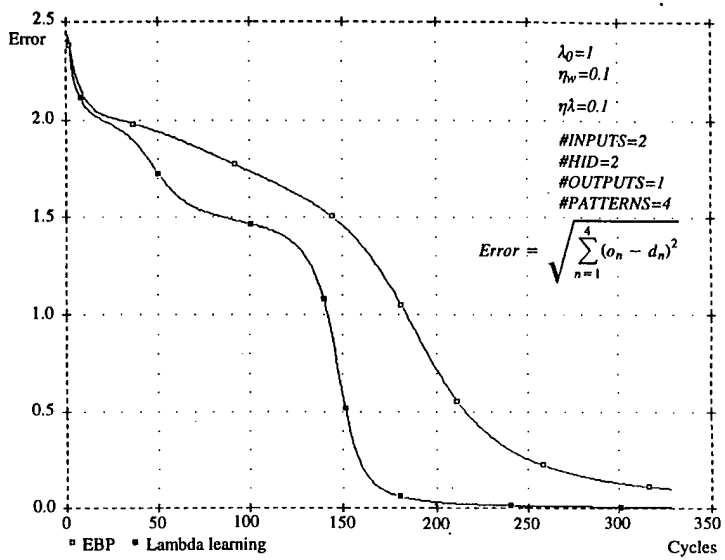


Fig. 2. Learning profile for the XOR problem.

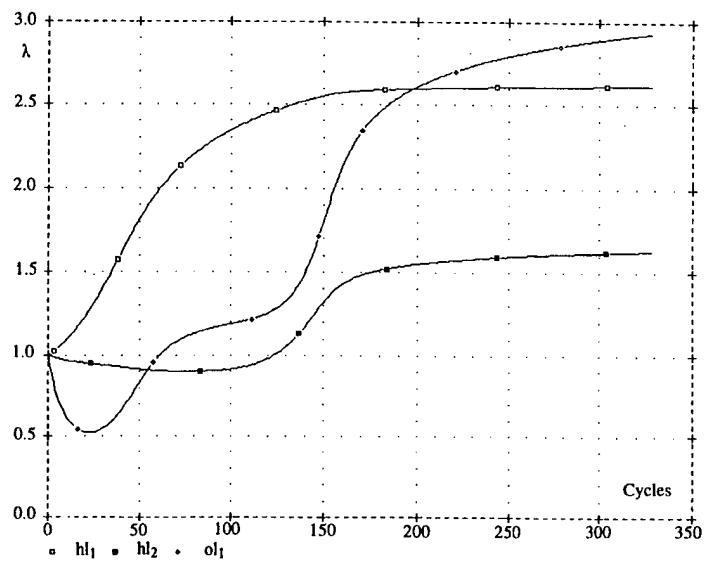


Fig. 3. Neurons' lambda variations during λ-learning for the XOR problem.

Efficiency of both algorithms has also been compared for training of a 36-class classifier of digits 0–9 and 26 letter characters with 8×10 binary input pixel field. The desired output vector has been the 7-bit ASCII code of each character. The architecture with 30 hidden nodes have been used. Figure 4 depicts two typical learning profiles produced for this application and indicates that the lambda learning method yields several times faster learning for the same final level of classifier's performance. Noticeably, Fig. 4 illustrates one case in which the EBPT method has not been successful and the lambda learning algorithm was.

In addition to the typical simulation results discussed in this section, some additional observations are in place. The lambda learning algorithm seems to be rather sensitive to initial weights and values. For small networks, such as XOR, $\lambda = 1$ and $\eta = 0.1$ yield seemingly best results. Initial weights uniformly spread within $\pm 1/\sqrt{(fan-in)}$ yield satisfactory results for initial value of $\lambda = 1$. It has also been found experimentally that both methods with weights initialized improperly result in persistent saturation of weights. In such a case no further learning occurs despite large output error value. Many simulations also indicated that lambda learning has led to quick and excellent solutions in cases when the standard EBPT has failed to produce acceptable results in reasonable time.

5. Sensitivity-based Approach for Input Vector Reduction

The minimization of redundancy which may be present in the training data is an important issue and rather rarely addressed in the technical literature. MLPNN are often used to model complex functional relationships between sets of experimental data. As such, they perform as function approximators which learn on a set of training patterns/examples (Hartman *et al.*, 1990; Shekhar and Amin, 1992).

Let us define the sensitivity of a trained MLPNN output o_k, with respect to its input x_i as

$$S_{x_i}^{o_k} \doteq \frac{\partial o_k}{\partial x_i} \tag{12}$$

which can be written succinctly as

$$S_{ki} \doteq S_{x_i}^{o_k} \tag{13}$$

By using standard notation of the EBPT, the derivative of (12) can be readily expressed in terms of network weights as follows

$$\frac{\partial o_k}{\partial x_i} = o_k' \sum_{j=1}^{J-1} w_{kj} \frac{\partial y_j}{\partial x_i} \tag{14}$$

where y_j denotes the output of the j-th neuron of the hidden layer, and o_k' is the value of derivative of the activation function $o = f(net)$ at the k-th output neuron. This further yields

$$\frac{\partial o_k}{\partial x_i} = o_k' \sum_{j=1}^{J-1} w_{kj} y_j' \nu_{ji} \tag{15}$$

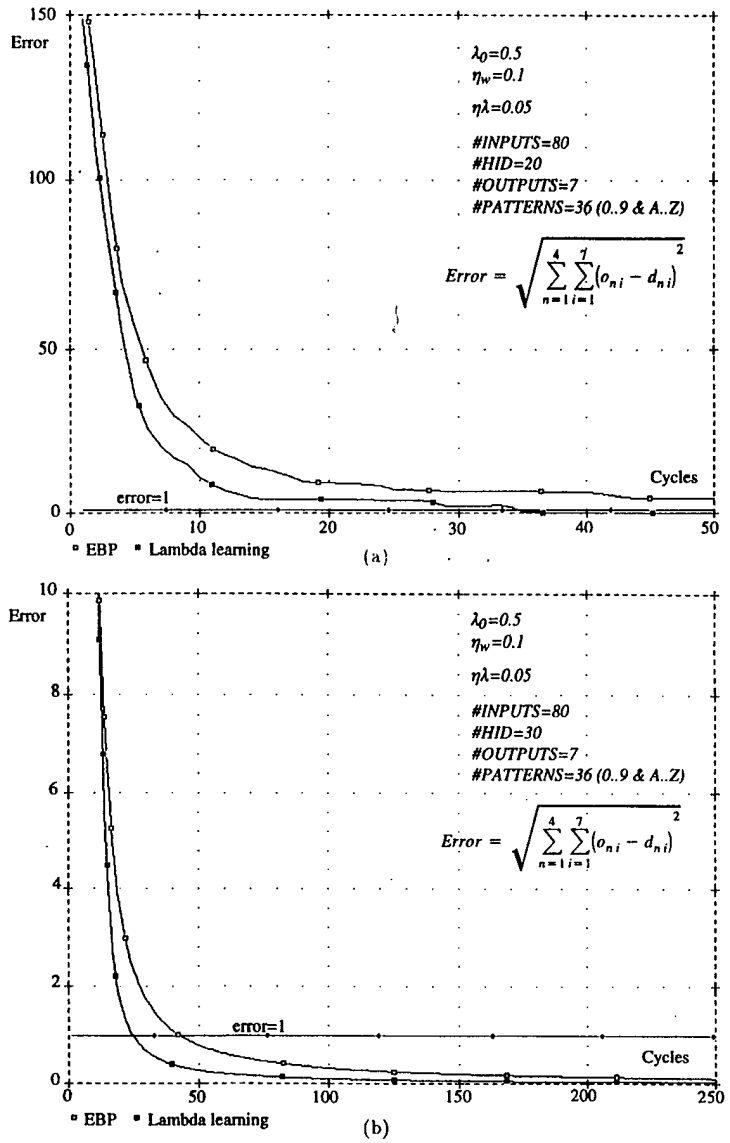


Fig. 4. Learning profiles for character classifiers (a) 20 hidden neurons (b) 30 hidden neurons.

where y'_j is the value of derivative of the activation function $y = f(net)$ of the j-th hidden neuron ($y'_J = 0$ since the J-th neuron is a dummy one, i.e. it serves as a bias input to the output layer). The sensitivity matrix $\boldsymbol{S}(K \times I)$ consisting of entries as in (15) or (13) can now be expressed using array notation as

$$\boldsymbol{S} = \boldsymbol{O}' \times \boldsymbol{W} \times \boldsymbol{Y}' \times \boldsymbol{V} \tag{16}$$

$\boldsymbol{W}(K \times J)$ and $\boldsymbol{V}(J \times I)$ are output and hidden layer weight matrices, respectively, and $\boldsymbol{O}'(K \times K)$ and $\boldsymbol{Y}'(J \times J)$ are diagonal matrices defined as follows

$$\begin{aligned} \boldsymbol{O}' &\doteq \mathrm{diag}(o'_1, o'_2, \ldots, o'_K) \\ \boldsymbol{Y}' &\doteq \mathrm{diag}(y'_1, y'_2, \ldots, y'_J) \end{aligned} \tag{17}$$

Matrix $\boldsymbol{S}$ contains entries S_{ki} which are ratios of absolute increments of output k due to the input i as defined in (13). This matrix depends only upon the network weights as well as slopes of the activation functions of all neurons. Each training vector $\boldsymbol{x}^{(n)} \in \mathcal{X}$, where $\mathcal{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\}$ denotes the training set, produces different sensitivity matrix $\boldsymbol{S}^{(n)}$ even for a fixed network. This is due to the fact that although weights of a trained network remain constant, the activation values of neurons change across the set of training vectors $\boldsymbol{x}^{(n)}$, $n = 1, 2, \ldots, N$. This, in turn, produces different diagonal matrices of derivatives $\boldsymbol{O}'$ and $\boldsymbol{Y}'$, which strongly depend upon the neurons' operating points determined by their activation values.

In order to possibly reduce the dimensionality of input vectors, the sensitivity matrix as in (16) needs to be evaluated over the entire training set $\mathcal{X}$. Let us define the sensitivity matrix for the pattern $\boldsymbol{x}_n$ as $\boldsymbol{S}^{(n)}$. There are several ways to define the overall sensitivity matrix, each relating to the different objective functions which need to be minimized.

The *mean square average sensitivities*, $S_{ki,avg}$, over the set $\mathcal{X}$ can be computed as follows

$$S_{ki,avg} \doteq \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(S_{ki}^{(n)}\right)^2} \tag{18}$$

Matrix $\boldsymbol{S}_{avg}(K \times I)$ is defined as $\boldsymbol{S}_{avg} = [S_{ki,avg}]$. This method of sensitivity averaging is coherent with the goal of network training which minimizes the mean square error over all outputs and all patterns in the set.

The *absolute value average sensitivities*, $S_{ki,abs}$, over the set $\mathcal{X}$ can be computed as follows

$$S_{ki,abs} \doteq \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left|S_{ki}^{(n)}\right|} \tag{19}$$

Matrix $\boldsymbol{S}_{abs}(K \times I)$ is defined as $\boldsymbol{S}_{abs} = [S_{ki,abs}]$. Note that summing sensitivities across the training set requires taking their absolute values due to the possibility of

cancelations of negative and positive values. This method of averaging may be better than (18) if sensitivities $S_{ki}^{(n)}$, $n = 1, \ldots, N$, are of disparate values.

The *maximum sensitivities*, $S_{ki,max}$, over the set $\mathcal{X}$ can be computed as

$$S_{ki,max} \doteq \max_{n=1,\ldots,N} \left\{ S_{ki}^{(n)} \right\} \tag{20}$$

Matrix $\boldsymbol{S}_{max}(K \times I)$ is defined as $\boldsymbol{S}_{max} = [S_{ki,max}]$. This sensitivity definition allows us to prevent deleting inputs which are relevant only in small percentage of inputs.

Any of the sensitivity measure matrices proposed in (18)–(20) can provide useful information as to the relative significance of each of the inputs in $\mathcal{X}$ to each of the outputs. For the sake of simplicity, however, mainly the matrix defined in (18) will be applied in further discussion. The cumulative statistical information resulting from (18) will be used along with criteria for reducing the number of inputs to the smallest number sufficient for accurate learning.

6. Algorithm for Pruning Inputs

Inspection of the average sensitivity matrix $\boldsymbol{S}_{avg}$ renders it possible to determine which inputs affect outputs least. A small value of $S_{ki,avg}$ in comparison to others means that for the particular k-th output of the network, the i-th input does not significantly contribute to output k, and may therefore be possibly disregarded. This property allows the formulation of the following pruning rule: *The sensitivity matrices for a trained neural network can be evaluated for both training and testing data sets; the values of average sensitivity matrix entries can be used for determining the least significant inputs and for reducing the size of network accordingly by pruning redundant inputs.*

When one or more of the inputs have relatively small sensitivity in comparison to others, the dimension of neural network can be reduced by suppressing them, and smaller-size neural network can be successfully used in most cases. The criterion used below for determining which inputs can be pruned is based on the so-called the largest gap method.

In order to normalize the data relevant for comparison of the significance of inputs, the sensitivity matrices defined in (18)–(20) have to be additionally preprocessed. The formulae needed for scaling are given in (21) and map each input into range [0, 1] as well as each output into range [−1, 1]:

$$\begin{aligned} \widehat{x}_i^{(m)} &\doteq \frac{x_i^{(m)} - \min\limits_{n=1,\ldots,N} \{x_i^{(n)}\}}{\left(\max\limits_{n=1,\ldots,N} \{x_i^{(n)}\} - \min\limits_{n=1,\ldots,N} \{x_i^{(n)}\} \right)} \\ \widehat{o}_k^{(m)} &\doteq \frac{o_k^{(m)} - \left(\max\limits_{n=1,\ldots,N} \{o_k^{(n)}\} + \min\limits_{n=1,\ldots,N} \{o_k^{(n)}\} \right)/2}{\left(\max\limits_{n=1,\ldots,N} \{o_k^{(n)}\} - \min\limits_{n=1,\ldots,N} \{o_k^{(n)}\} \right)} \end{aligned} \tag{21}$$

If input and output data scaling (21) was performed before network training, no additional operations on S_{ki} are required and we have

$$\widehat{S}_{ki,avg} \doteq S_{ki} \tag{22}$$

Note that the scaling can be performed either on entries of $\boldsymbol{S}$ or $\boldsymbol{S}_{avg}$. In the case when network original inputs and outputs are not scaled to the same level, additional scaling is necessary to allow for accurate comparison among inputs:

$$\widehat{S}_{ki,avg} \doteq S_{ki} \frac{\left(\max\limits_{n=1,\ldots,N} \{x_i^{(n)}\} - \min\limits_{n=1,\ldots,N} \{x_i^{(n)}\} \right)}{\left(\max\limits_{n=1,\ldots,N} \{o_k^{(n)}\} - \min\limits_{n=1,\ldots,N} \{o_k^{(n)}\} \right)} \tag{23}$$

The significance of the i-th input Φ_i across the entire set $\mathcal{X}$ is defined as follows:

$$\Phi_{i,avg} \doteq \max_{k=1,\ldots,K} \{\widehat{S}_{ki,avg}\} \tag{24}$$

In order to distinguish inputs, entries of Φ have to be sorted in descending order so that

$$\Phi_{i_{m+1}} \geq \Phi_{i_m}, \quad m = 1, \ldots, I-1 \tag{25}$$

where i_m is a sequence of sorted input numbers. Let us define the measure of gap as

$$g_{i_m} \doteq \frac{\Phi_{i_m}}{\Phi_{i_m+1}} \tag{26}$$

and then find the largest gap using the following formula

$$g_{max} \doteq \max_{i_m} \{g_{i_m}\} \quad \text{and} \quad m_{CUT} \doteq m \quad \text{such that} \quad g_{i_m} = g_{max} \tag{27}$$

If condition (28) below is valid, then the gap found between m_{CUT} and m_{CUT+1} is large enough:

$$C g_{max} > \max_{i_m \neq i_{m_{CUT}}} \{g_{i_m}\} \tag{28}$$

Constant C from (28) is chosen arbitraily within a reasonable range (e.g. $C = 0.5$; the smaller C, the stronger is the condition for existence of an acceptable gap). All inputs with indexes $\{i_{m+1} \ldots i_{I-1}\}$ can be pruned with the smallest loss of information to the MLPNN.

The gap method can be also applied for comparison among sensitivities of inputs to each output separately. For this purpose, a set containing candidates for pruning can be created for every output. Final pruning is then performed by removing these inputs which can be found in every set determined previously for each output independently.

$\boldsymbol{S}_{avg}$ can be meaningfully evaluated only for well trained neural networks. Despite this condition, it can still save computational effort when initial learning can be performed on smaller, but still representative subset of data. $\boldsymbol{S}_{avg}$ can be evaluated

based either on the data set used for initial training or on the complete data set. Subsequently, newly developed neural network with appropriate inputs can be retrained using the full set of training patterns of reduced dimension.

7. Experimental Results

A series of numerical simulations was performed in order to verify the proposed definitions and pruning criteria. In the first experiment a training set for a neural network was generated using four inputs $x_1, \ldots, x_4$ and two outputs o_1 and o_2. Values of outputs were correlated with x_1 and x_2 for o_1, and with x_2 and x_3 for o_2. Input vectors $\boldsymbol{x}$ (4×1) were produced using a random number generator. The expected values of vector $\boldsymbol{d}$ (2×1) for the output vector $\boldsymbol{o}$ (2×1) were evaluated for each $\boldsymbol{x}$ using a known relationship $\boldsymbol{d} = \boldsymbol{F}(\boldsymbol{x})$ where $\boldsymbol{d}$ is the desired (target) output vector for supervised training. The training set $\mathcal{X}$ consisted of $N = 81$ patterns. A neural network with 4 inputs, 2 outputs and 6 hidden neurons ($I = 5$, $J = 7$, $K = 2$) has been trained for the mean square error defined as follows

$$MSE \doteq \sqrt{\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \left(d_k^{(n)} - o_k^{(n)} \right)^2} \tag{29}$$

equal to 0.001 per input vector. Matrices of sensitivities were subsequently evaluated and $\boldsymbol{S}_{avg}$ produced at the end of training over the entire input data set $\mathcal{X}$.

The changes of sensitivity entries during learning are presented in Fig. 5.

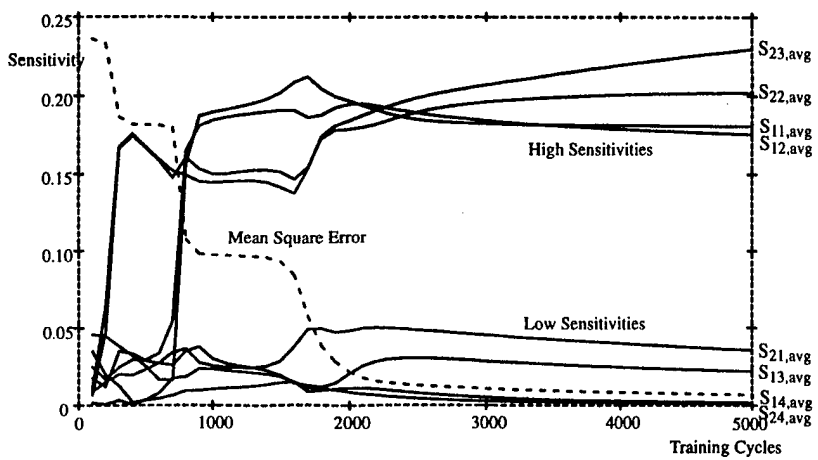


Fig. 5. Sensitivity profile for the full training data set.

It can be seen that an untrained neural network in the example has per average smaller sensitivities than after the training. During the training some of the average

sensitivities $\boldsymbol{S}_{ki,avg}$ increase, while the others converge towards low values. Final values of sensitivities of the first output offer hints for deleting x_3 and x_4, and these for the second output indicate that x_1 and x_4 could be deleted. The only input which occurs in both sets of candidates for deletion is x_4. Therefore, the fourth input to the network can be skipped and its dimension reduced to 3 ($I = 4$).

The second experiment was performed using a larger network and fuzzy data. MLPNN had 20 inputs ($I = 21$), 10 hidden neurons ($J = 26$) and 4 outputs ($K = 4$). There were $N = 500$ patterns in the training set and several additional data sets of the same size for network performance evaluation. The network was successfully trained to the MSE error of 0.15. However, due to the fuzziness of the training data, MSE error for additional sets remained at the level of 0.20.

All outputs were strongly correlated with inputs $x_1, x_2, x_3, x_4, x_6, x_8$, and x_9. Input x_6 during data generation was multiplied by random numbers, while the influence of x_2 and x_4 on outputs was scaled down to remain small in comparison with other inputs (less than 0.05).

Input significance coefficients calculated using formulae (18)–(20) are shown in Fig. 6.

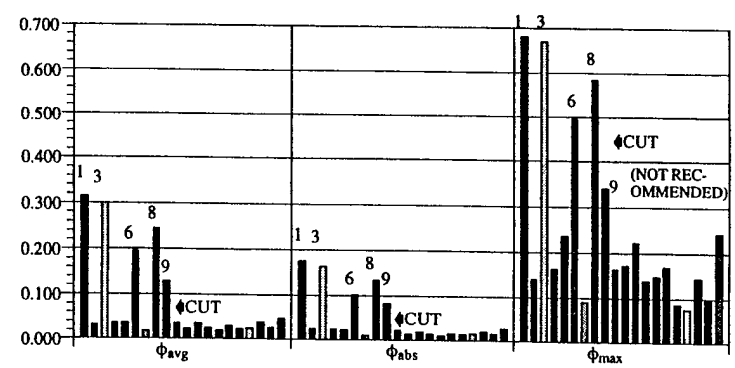


Fig. 6. Input significance coefficients ϕ for different sensitivities (18)–(20) and pruning criterion (28).

Inputs x_2 and x_4 are identified after sorting as less important than other, not correlated inputs. This is due to their low correlation with outputs. They therefore can be ignored along with other inputs which are not correlated for a given MSE error value. The sequence of significance is the same for all proposed methods, however, the sizes of gaps are different in each case. Value $C = 0.5$ prevents pruning using ϕ_{max} definition. Note that the maximum method does not give the clear clue how to choose the significance level for purging due to fuzziness of the training data.

The result of initial training is shown in Fig. 7. It can be determined from this figure which inputs should remain after pruning. The network performance after

pruning is shown in Fig. 8. No further input dimensionality reduction is possible since no large gap in coefficients ϕ can be found. The speed of training has increased mostly because of reduction of the MLPNN size (input dimension reduced by 4). The necessary number of the training cycles has also decreased, but not so dramatically as in the first experiment.

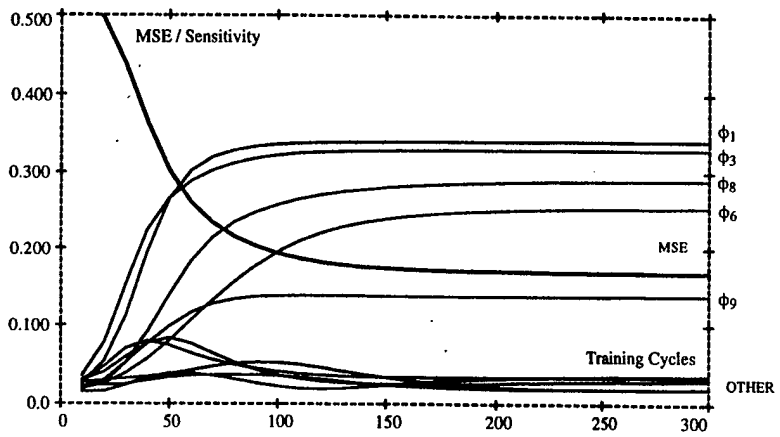


Fig. 7. Input significance ϕ_{avg} for the full training set.

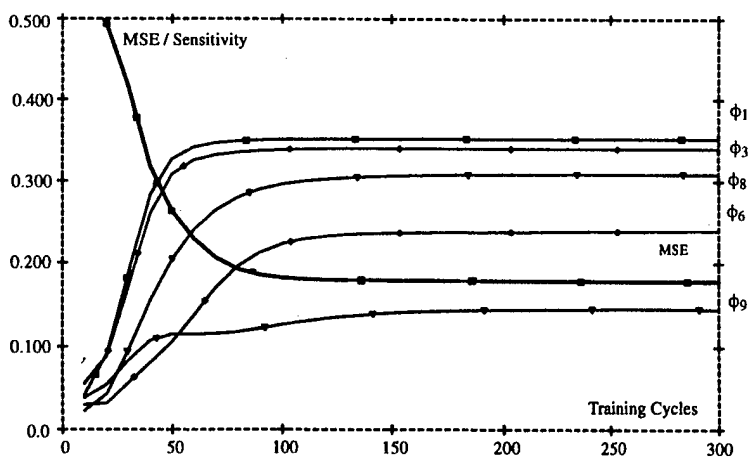


Fig. 8. Input significance ϕ_{avg} for the pruned training set.

Using the sensitivity-based approach for input layer pruning seems particularly useful when network training requires large amount of redundant, oversized data. In the first phase, network can be pre-trained until the training error decreases satisfactorily. Then sensitivity matrices can be evaluated and the dimension of the input layer possibly reduced. Learning can subsequently be resumed until the training error reduces to an acceptable low value. This process can be repeated, however, usually only the first execution yields significant improvement. Numerical experiments indicate that the effort of additional network retraining can be too high in comparison to benefits of further minimization.

Should the redundancy in training data vectors exist, the proposed approach based on the average sensitivity matrices for input data pruning allows for more efficient training. This can be achieved at a relatively low computational cost and based on heuristic data pruning criteria outlined in the paper.. The approach can be combined with other improved training strategies such as increased complexity training (Cloete and Ludik, 1993). Extension of the proposed sensitivity-based input pruning concept beyond continuous output values seems desirable for the case of networks with binary outputs such as classifiers and other binary encoders.

8. Minimum Sampling Rate for Information Encoding

An analytical approach allowing finding the optimum number of training examples for MLPNN can be outlined based on the Nyquist Sampling Theorem. The theorem states that *a function $f(x)$ which contains no frequency components greater than f_0 Hz is uniquely determined by the values of $f(x)$ at any set of sampling points spaced at most $1/(2f_0)$ seconds apart (Philips, 1990; Poularikas, 1992).* Sampling rates defined for time signals can be extended to other independent variables so that the generalized theorem for function approximation can be obtained. Each dimension of the transform will then correspond to one dimension of the original domain.

Obviously, sampling with a certain minimum frequency is needed to restore the signal from the samples taken. However, the theorem refers to the ideal case where the input signal has a finite high frequency boundary so that it can be accurately restored from samples using the inverse Fourier transform. Real-life signals are not band-limited, and we focus on the analysis of approximation conditions for MLPNNs.

An MLPNN trained using backpropagation algorithm can be regarded as a mean square error-based function approximation. However, no closed-form formulae for approximation polynomials are known before training. This makes it impossible to evaluate the sampling step theoretically. Therefore, the only hint for choosing the sufficient sampling interval is that it should be shorter than the Nyquist interval. The developed algorithm for non-band-limited functions is based on the assumption that only a certain fraction of information about the function is necessary for the approximation with a required accuracy.

Below we discuss the continuous function case. Let a continuous function to be approximated be known as $f(\boldsymbol{x})$,

$$f(\boldsymbol{x}) : \mathbb{D} \to \mathbb{R}, \qquad \mathbb{D} \subset \mathbb{R}^N$$

$$\mathbb{D} \doteq (x_{MIN_1} \ldots x_{MAX_1}) \times (x_{MIN_2} \ldots x_{MAX_2}) \times \ldots \times (x_{MIN_N} \ldots x_{MAX_M}) \quad (30)$$

$$\boldsymbol{x} \doteq [x_1, x_2, \ldots, x_n]$$

and let B_i be the range of the i-th variable, x_i:

$$B_i \doteq x_{MAX_i} - x_{MIN_i} \quad (31)$$

The multidimensional Fourier transform of $f(\boldsymbol{x})$ can now be expressed as

$$F(\boldsymbol{\Omega}) \doteq \frac{1}{(2\pi j)^N} \int_{x_{MIN_1}}^{x_{MAX_1}} \int_{x_{MIN_2}}^{x_{MAX_2}} \ldots \int_{x_{MIN_N}}^{x_{MAX_N}} f(x_1, x_2, \ldots, x_N) e^{2\pi j x_1 \omega_1} e^{2\pi j x_2 \omega_2}$$

$$\ldots e^{2\pi j x_N \omega_N} \, \mathrm{d}x_1 \mathrm{d}x_2 \ldots \mathrm{d}x_N \quad (32)$$

where $\boldsymbol{\Omega} \doteq [\omega_1, \omega_2, \ldots, \omega_N]$ is a vector of frequencies.

The criterion for minimum sampling frequency estimation can be formulated in a number of ways. First, the basic formula for optimization should be defined as a norm evaluating the information density at particular frequencies. The norm as defined in (33) has a meaning of generalized energy density:

$$E_d(\boldsymbol{\Omega}) \doteq |F(\boldsymbol{\Omega})|^2 \quad (33)$$

We can also use function (34) for evaluating the amount of information enclosed in the frequency band $\boldsymbol{\Omega}$. In case of a multidimensional band-limited function, the energy $E(\boldsymbol{\Omega})$, can be computed by integrating the generalized energy density (33) in the frequency domain in spherical (Andrews *et al.*, 1970), or more precisely, ellipsoidal coordinates within an N-dimensional ellipsoid. For example, in the simplest case assuming that function F has isotropic properties in each dimension ($\omega = \omega_1 = \omega_2 = \ldots = \omega_N$), we obtain

$$E(\boldsymbol{\Omega}) \doteq \int_0^{2\pi} \int_0^{2\pi} \ldots \int_0^{2\pi} \int_0^{1} |F(\omega r \cos\phi_1 \cos\phi_2 \ldots \cos\phi_{N-1}, \omega r \sin\phi_1 \cos\phi_2 \ldots \cos\phi_{N-1},$$

$$\ldots, \omega r \sin\phi_1 \sin\phi_2 \ldots \sin\phi_{N-1})|^2 r J(r, \phi_1, \phi_2, \ldots, \phi_N) \, \mathrm{d}\phi_1 \mathrm{d}\phi_2 \ldots \mathrm{d}\phi_{N-1} \mathrm{d}r \quad (34)$$

where $J(r, \phi_1, \phi_2, \ldots, \phi_N)$ is a term resulting from the change of the integration coordinates from cubic to spheric (Apostol, 1957). However, in general it cannot be assumed that the approximated function will have isotropic properties. It may then be

reasonable to choose smaller sampling densities in some dimension. The function (34) becomes then more complex due to the different boundaries in each dimension

$$E(\boldsymbol{\Omega}) \doteq \int_0^{2\pi} \int_0^{2\pi} \ldots \int_0^{2\pi} \int_0^{1} |F(\omega_1 r \cos\phi_1 \cos\phi_2 \ldots \cos\phi_{N-1}, \omega_2 r \sin\phi_1 \cos\phi_2 \ldots \cos\phi_{N-1},$$

$$\ldots, \omega_N r \sin\phi_1 \sin\phi_2 \ldots \sin\phi_{N-1})|^2 J(r, \Omega, \phi_1, \phi_2, \ldots, \phi_N) \, \mathrm{d}\phi_1 \mathrm{d}\phi_2 \ldots \mathrm{d}\phi_{N-1} \mathrm{d}r \quad (35)$$

where $J(r, \boldsymbol{\Omega}, \phi_1, \phi_2, \ldots, \phi_N)$ is a term obtained as previously after conversion of the integration coordinates from cubic to spheric.

Let C_{INFO} called the *information rate factor* be the fraction describing the required minimum energy content of the signal sampled with frequency $\boldsymbol{\Omega}$, normalized with respect to the total energy E_{TOT} of the original function (or function sampled with very high frequency). The information rate factor C_{INFO} can be seen as a theoretical measure of the information amount needed to approximate a function with a required accuracy.

Let function $f(\boldsymbol{x})$ be sampled with frequency $\boldsymbol{\Omega}$ satisfying the following condition:

$$\frac{E(\boldsymbol{\Omega})}{E(\boldsymbol{\Omega}_{MAX})} \geq C_{INFO} \quad (36)$$

where the frequency $\boldsymbol{\Omega}_{MAX} \doteq [\omega_1, \omega_2, \ldots, \omega_N]_{MAX}$ is high enough, so that

$$\frac{E(\boldsymbol{\Omega}_{MAX}) - E(\boldsymbol{\Omega})}{E(\boldsymbol{\Omega}_{MAX})} \ll C_{INFO} \quad (37)$$

Let us now compute the total number of samples in the training set. The number of samples taken per dimension M_{L_i}, is equal to

$$M_{L_i}(\omega_i) = |2B_i\omega_i + 1| \quad (38)$$

The total number of samples M_L can be expressed as

$$M_L(\boldsymbol{\Omega}) = \prod_{i=1}^{N} M_{L_i}(\omega_i) \quad (39)$$

Our objective is to search among vectors $\boldsymbol{\Omega}$ which satisfy the condition (36) and to minimize the value of M_L defined by (39). The vector $\boldsymbol{\Omega}_{OPT} \doteq [\omega_1, \omega_2, \ldots, \omega_N]_{OPT}$ which is the solution to the given optimization problem contains the minimum sufficient sampling frequencies in the new training data set. The final sampling interval, Δx_i, is expected to be of different value for each dimension depending on the chosen frequency ω_i:

$$\Delta x_i = \frac{1}{2\omega_{OPT_i}} \quad (40)$$

Let us redefine the results for continuous functions for discrete data sets. Let us consider sampling a plant characteristic for which no closed-form formula exists.

Assume that data is available with intervals $\delta x \doteq [\delta x_1, \delta x_2, \ldots, \delta x_N]$ on $\mathbb{D}$, so that there is a given data set $\mathcal{S}$ for estimation

$$\mathcal{S} \doteq \Big\{ ((x_{1_{k_1}}, x_{2_{k_2}}, \ldots, x_{N_{k_N}}),\ y_{k_1,k_2,\ldots,k_N}) \quad k_1 = 0, \ldots, K_1 - 1, \ldots, k_N = 0, \ldots, K_{N-1} \Big\} \tag{41}$$

where

$$x_{i_{k_i}} = x_{MIN_i} + \delta x_i k_i, \quad y_{k_i} = f(x_{1_{k_1}}, x_{2_{k_2}}, \ldots, x_{N_{k_N}}), \quad K_i = \left[\frac{B_i}{\delta x_i}\right] \tag{42}$$

The set $\mathcal{S}$ contains the entire information available about the function to be approximated. The final training set after calculations of minimum sufficient sampling density will be its subset. Now the information about the frequency domain can be obtained using the discrete Fourier transform (DFT) of $\mathcal{S}$ as follows:

$$F(\boldsymbol{l}) \doteq \sum_{k_1=1}^{K_1} \sum_{k_2=1}^{K_2} \cdots \sum_{k_N=1}^{K_N} y_{k_1 k_2 \ldots k_N} e^{2\pi j \left(\frac{k_1 l_1}{K_1} + \frac{k_2 l_2}{K_2} + \ldots + \frac{k_N l_N}{K_N}\right)} \tag{43}$$

where the vector $\boldsymbol{l}$ is the multidimensional discrete frequency equal to

$$\boldsymbol{l} = [l_1, l_2, \ldots, l_N] \quad \text{and} \quad l_i \in \{0, 1, \ldots, K_i - 1\} \tag{44}$$

The set $\mathcal{S}$ contains the function to be approximated of the band-limited spectrum due to the finite sampling density. Therefore, no information about higher frequency components exists. For this reason, initial sampling rates $\boldsymbol{\Omega}_{MAX}$ should be chosen carefully.

The norm for information density for discrete functions can be defined as follows:

$$E_d(\boldsymbol{l}) \doteq |F(\boldsymbol{l})|^2 \tag{45}$$

Let us note that l_i is an integer number, and it corresponds to the sampling frequency

$$\omega_i = l_i \Delta\omega_i \tag{46}$$

where

$$\Delta\omega_i = \frac{1}{2\delta x_i (K_i - 1)}, \qquad \omega_{MAX_i} = \frac{1}{2\delta x_i} \tag{47}$$

The final number of samples per dimension M_{L_i} can now be expressed as

$$M_{L_i} = \frac{B_i}{\delta x_i (K_i - 1)} l_i + 1 = 2l_i + 1 \tag{48}$$

The term M_L in (39), which is the number of samples available for training, can be evaluated as

$$M_L(\boldsymbol{\Omega}) = \prod_{i=1}^{N} (2l_i + 1) \tag{49}$$

More details of the discussion here is provided in (Malinowski *et al.*, 1994).

9. Simplified Approach for Estimating Minimum Sufficient Sampling Rates

A simplified expression for the multidimensional rectangle inscribed into the multidimensional ellipsoid is shown below (Andrews, 1970):

$$G(\boldsymbol{l}) = \sum_{k_1=1}^{l_1} \sum_{k_2=1}^{l_2} \cdots \sum_{k_N=1}^{l_N} |F(\boldsymbol{k})|^2 \tag{50}$$

The expression for $G(\boldsymbol{l})$ has to be normalized so that it approaches 1 at the maximum discrete frequency:

$$G_{NORM}(\boldsymbol{l}) = \frac{G(\boldsymbol{l})}{G(K_1 - 1, \ldots, K_N - 1)/2^N} \tag{51}$$

The requirement for sufficient information now replacing (36) and valid after narrowing the frequency boundary reduces to

$$G_{NORM}(\boldsymbol{l}) \geq C_{INFO} \tag{52}$$

The value of $\omega_{i_{MAX}}$ corresponds to the maximum discrete frequency $L_{i_{MAX}}$ which is

$$L_{i_{MAX}} = \left[\frac{K_i - 1}{2}\right] \tag{53}$$

The constant C_{INFO} needs to be estimated in order to be used in equations (36) and (52). This constant defines the minimum amount of information left after narrowing the frequency spectrum in terms of the function's energy. It can be estimated in the way described below.

Let us note that the error defined in the sense of (29) is based on the energetic distance between the original and the approximated function and is also useful for expressing the training termination condition. In order to determine the value of MSE, it is necessary to know the average power P_{TOT} of the original function. P_{TOT} can be calculated from the Fourier power spectrum in the frequency domain either in the x domain using formula (54) for the continuous case,

$$P_{TOT} = \frac{\int_{x_{MIN_1}}^{x_{MAX_1}} \int_{x_{MIN_2}}^{x_{MAX_2}} \cdots \int_{x_{MIN_N}}^{x_{MAX_N}} f(x_1, x_2, \ldots, x_N)^2 \, \mathrm{d}x_1 \mathrm{d}x_2 \ldots \mathrm{d}x_N}{\prod_{i=1}^{N} B_i} \tag{54}$$

or using (55) in case of discrete data:

$$P_{TOT} = \frac{1}{P} \sum_{p=1}^{P} (d^{(p)})^2 \tag{55}$$

The required approximation accuracy Ψ links together values of MSE and P_{TOT} as follows

$$\Psi = \frac{MSE}{\sqrt{P_{TOT}}} \tag{56}$$

The reason for normalizing the variables defined in (29) and (55) is to allow easy comparison of training results, and to evaluate the quality of the approximation without considering the number of patterns used each time. Finally, we have the relationship proposed after the inspection of the properties of the sampled signal

$$C_{INFO} = 1 - \Psi \tag{57}$$

which links the final condition for training (56) with equation (52).

The following algorithm is proposed for finding the minimum sufficient sampling rates (it is based on the theoretical assumptions presented above):

STEP 1: Sample plant characteristics with certain input data step with excessively oversampling to allow for sufficient generalization. Let $f(\boldsymbol{k}) := f(\boldsymbol{x})$ ($\delta\boldsymbol{x}$ is a vector of sampling intervals).

STEP 2: Compute the DFT of the sampled function, $F(\boldsymbol{l}) := f(\boldsymbol{k})$.

Note that the upper boundary of the Fourier transform depends on the sampling interval $\delta\boldsymbol{x}$. Condition (37) has to be satisfied. Therefore, if the sampling interval is too large, the significant part of the frequency response is lost.

STEP 3: If the frequency response is not small enough at the highest frequency in comparison with the lower ones, i.e. (37) is not satisfied, first two steps should be repeated; sampling for 5 to 10 times more frequent in the appropriate dimensions.

STEP 4: Complete the information measurement function $G(\boldsymbol{l})$ as in (50) and normalize it according to (51) so that its maximum is equal to 1.

$$G_{NORM}(\boldsymbol{l}) := F(\boldsymbol{l})$$

STEP 5: Evaluate C_{INFO} and MSE for particular requirements of approximation accuracy Ψ based on (57).

STEP 6: Check for what frequencies function $G(\boldsymbol{l})$ from (52) reaches the levels evaluated in STEP 5.

$$\{\boldsymbol{l}\} := \{\boldsymbol{l} : G_{NORM}(\boldsymbol{l}) \geq C_{INFO}\}$$

STEP 7: Solve for $\boldsymbol{l}$ which produces minimum M_L in (49) using the set $\{\boldsymbol{l}\}$ satisfying condition from STEP 6.

$$\boldsymbol{l}_{OPT} := \left\{\boldsymbol{l} : M_L(\boldsymbol{l}) = \min\{M_L(\boldsymbol{l})\} \quad \text{over all} \quad \boldsymbol{l} \in \{\boldsymbol{l}\}\right\}$$

STEP 8: Choose the sampling steps M_{L_i} slightly higher than those corresponding closely to frequencies computed in STEP 7.

STOP.

If problems with convergence are encountered, the MLPNN architecture should be changed or the learning constants decreased. If the approximation error after completed training is excessive, sampling steps should be decreased by choosing more severe constraints than given in (57).

10. Experimental Results

A series of experiments were conducted to confirm the theoretical results and to test the heuristic guidelines proposed for sampling rates. An MLPNN with one hidden layer has been used for single-variable function approximation. The experiments were performed for approximating one-, and two-dimensional functions using neural network architectures with different numbers of hidden neurons and for different final error conditions.

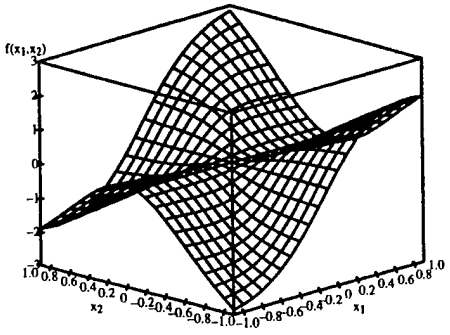
To prevent saturation of neurons and to provide similar conditions for each test the scaling of input data was performed so that normalized input variables varied form 0 to 1. Since bipolar continuous neurons were used, it was necessary to scale functions to be approximated to the range between -1 and 1. Standard and modified (lambda learning) EBPT methods were used for learning. Functions were first sampled with very high density for evaluating the discrete Fourier transform and for evaluating the approximation accuracy after completing the training. Before each training with a new sampling step, a new learning data set was created and network weights were initialized. Each training was performed until it has reached the MSE error set previously.

Theoretical estimations were compared with frequencies obtained from experiments. As anticipated, there exists an optimal number of learning points for a given approximation accuracy. This number of points can be evaluated from the normalized integral of its Fourier transform (52).

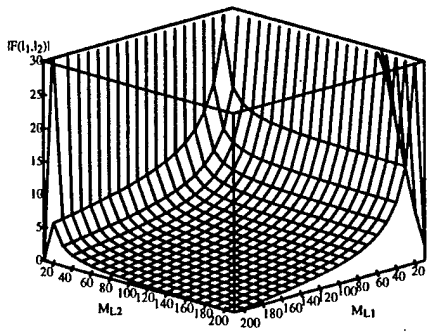
Figure 9(a) shows an example function used for approximation. Figure 9(b) depicts one quadrant of the frequency domain of the magnitude of its discrete Fourier transform. Function values were generated using formula (58) with the sampling interval $\delta x_i = 0.005$ in each dimension ($K_1 = K_2 = 400$ samples).

$$f(\boldsymbol{x}) = \frac{x_1^2 x_2 + 5x_1 x_2^2}{x_1^2 + x_2^2 + 0.1}, \quad x_1 \in (-1, 1), \quad x_2 \in (-1, 1) \tag{58}$$

The normalized energy function of $f(\boldsymbol{x})$ covered by the frequency $\boldsymbol{l}$ is shown in Fig. 10



(a)



(b)

Fig. 9. Approximated function $f(x)$ as in (53) (a), and its Fourier transform (b); ($P_{TOT} = 1.12$).

The average power P_{TOT} was calculated for the given function using formula (55); it is of value $P_{TOT} = 1.12$. MLPNN with 2 inputs, and 20 hidden neurons was trained to the error $MSE = 0.08$. Ψ and C_{INFO} were then calculated from equations (55) and (57) for $MSE = 0.08$ as $\Psi = 0.07$, and $C_{INFO} = 0.93$. The optimum number of samples for each dimension has been found from Fig. 11 by finding the minimum of M_L over frequencies satisfying condition (52) which shows the contour line bounding the domain of solution. This figure shows the contours for the number of samples in the training set M_L, for different l_1 and l_2 which satisfy condition (52). The minimum of M_L can be found at $L_1 = 4$ and $L_2 = 8$. It can be evaluated from equation (48). This corresponds to $M_{L_1} = 9$ samples for variable x_1 and $M_{L_2} = 17$ samples for variable x_2.

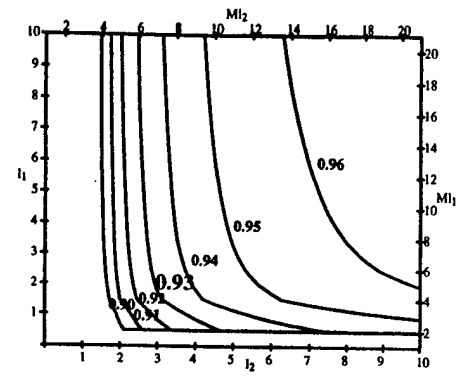


Fig. 10. Amount of energy $G_{NORM}(l)$ covered by bounded frequency spectrum ($E_{NORM} > 0.90$).

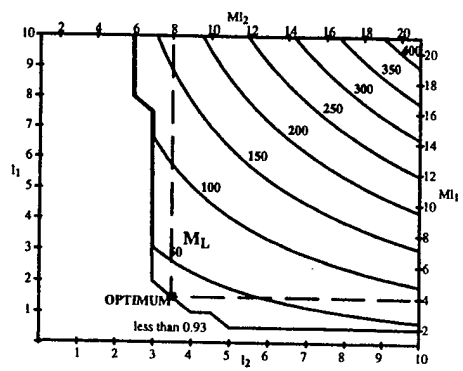
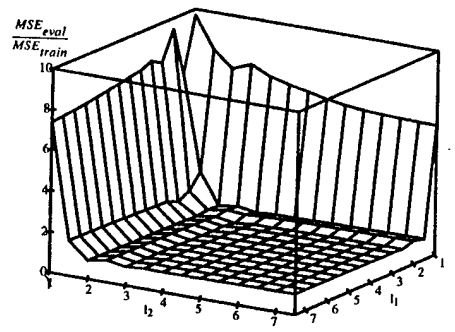
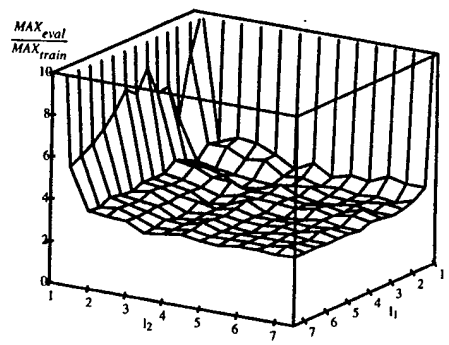


Fig. 11. Number of samples in learning set in area of normalized $G(l_1, l_2) > C_{INFO}$ ($C_{INFO} \geq 0.93$).

MLPNNs with architectures described above were then trained for different numbers of samples taken in each dimension to verify the theoretical results. The results are illustrated in Figs. 12–13. Figure 12 shows the quality of training in terms of MSE and the maximum error of approximation verification based on a very large test set (500×500 samples). It can be seen that the error decreases dramatically when $l_1 \geq 2$ and $l_2 \geq 3$. This corresponds to $M_{L_1} = 5$ and $M_{L_2} = 7$ samples per dimension, respectively.



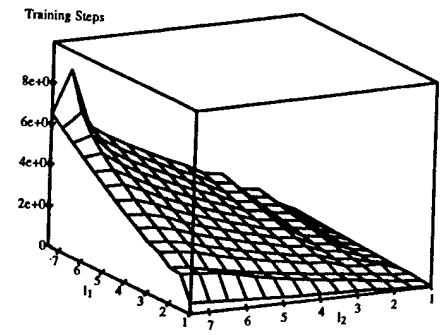
(a)



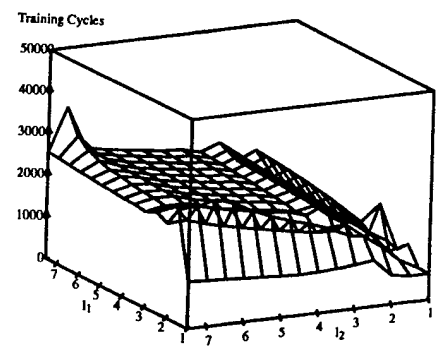
(b)

Fig. 12. Neural network performance (a) MSE (b) MAX.

Figure 13(a) shows the number of training steps required for the learning process, while Fig. 13(b) shows only the number of training cycles. After achieving certain frequencies of sampling the function to build a training set, the number of training cycles does not increase or increases only slowly, while the overall number of training steps still increases due to the growing number of training vectors.



(a)



(b)

Fig. 13. Number of training steps (a) and training cycles (b) versus sampling frequencies.

Figure 14 summarizes the computational experiment. It illustrates the number of training steps for the sampling frequencies l providing accurate learning. Local minima can be found for the frequencies $l_1 = 2$ and $l_2 = 5$. This corresponds to $M_{L_1} = 5$ and $M_{L_2} = 11$ samples per dimension, respectively. This is in agreement with $M_{L_1} = 4$ and $M_{L_2} = 8$ samples per appropriate dimension obtained by inspection of Fig. 11. The results are then rather close to those evaluated previously using the theoretical algorithm and shown in Fig. 11.

Both the results of the computational experiments and of theoretical studies show that the generalized sampling theorem can be applied to the approximation problem using neural networks. The smallest possible, but still large enough for the sake of accuracy data set should be selected, and then other network parameters can be

found through training. Our results indicate that the smallest training sets with only some data from the large measured real-life data sets are required in order to obtain successfully trained neural networks capable of accurate approximation.

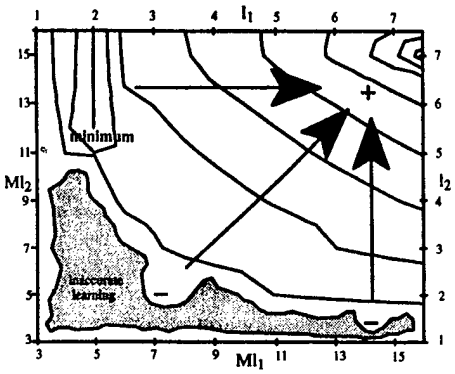


Fig. 14. Number of training steps versus sampling frequencies for area of satisfactory learning.

References

Andrews H.C., Pratt W.K. and Caspari K. (1970): *Computer Techniques in Image Processing.* — London: Academic Press Inc. Ltd.

Apostol T.M. (1957): *Mathematical Analysis. A Modern Approach to Advanced Calculus.* — London: Addison-Wesley Publishing Company, pp.270–275.

Cloete I. and Ludik J. (1993): *Increased complexity training.* — In: New Trends in Neural Computation, Eds. Mira J., Cabestany j. and Prieto A., in series Lecture Notes in Computer Science, Berlin: Springer-Verlag, pp.267–271.

Fu L. and Chen T. (1993): *Sensitivity analysis for input vector in multilayer feedforward neural networks.* — Proc. IEEE Int. Conf. *Neural Networks*, San Francisco (USA), March 28–April 1, pp.215–218.

Hartman E.J., Keeler J.D. and Kowalski J.M. (1990): *Layered neural networks with gaussian hidden units as universal approximators.* — Neural Computation, v.2, No.2, pp.210–215.

Kruschke J.K. and Movellan J.R. (1991): *Benefits of gain: speeded learning and minimal hidden layers in back-propagation networks.* — IEEE Trans. Systems, Man, and Cybernetics, v.21, No.1, pp.273–279.

Malinowski A., Żurada J.M. and Aronhime P.B. (1994): *Minimal training set size estimation for neural network-based function approximation.* — Proc. IEEE Int. Symp. *Circuits and Systems*, London (U.K.), May 28–June 2, pp.403–406.

Movellan J.R. (1987): *Self-regulated temperature in back-propagation networks.* — Presented at the Ninth Annual Berkeley-Stanford Conference, Berkeley, California.

Philips C.L. and Nagle H.T. (1990): *Digital Control Systems — Analysis and Design.* — New Jersey: Prentice Hall Inc.

Poularikas A.D. and Seely S. (1992): *Elements of Signals and Systems.* — Boston: PWS-KENT Publishing Company.

Shekhar S. and Amin M.B. (1992): *Generalization by neural networks.* — IEEE Trans. Knowledge and Data Eng., v.4, No.2, pp.177–185.

Tawel R. (1989): *Does the neuron 'learn' like synapse.* — In: Advances in Neural Information Processing Systems (Touretzky D.S., Ed.). – San Mateo (USA): Morgan Kaufmann, pp.169–176.

Żurada J.M. (1992): *Introduction to Artificial Neural Systems.* — St. Paul, (Minn.), West Publishing Company.

Żurada J.M. (1993): *Lambda learning rule for feedforward neural networks.* — Proc. IEEE Int. Conf. *Neural Networks*, San Francisco (USA), March 28–April 1, pp.1808–1811.

Żurada J.M., Malinowski A. and Cloete I. (1993): *Sensitivity analysis for pruning of training data in feedforward neural networks.* — Proc. First Australian and New Zeland Conf. *Intelligent Information Systems*, Perth, Western Australia, December 1–3, pp.288–292.

Żurada J.M., Malinowski A. and Cloete I. (1994): *Sensitivity analysis for minimization of input data dimension for feedforward neural network.* — Proc. IEEE Int. Symp. *Circuits and Systems*, London (U.K), May 28–June 2, pp.447–450.