Method for optimisation of mobile and wireless transceiver design

Duric, H.

Award date:
2002

TU/e technische universiteit eindhoven

Department of Electrical Engineering
Division of Telecommunication Technology and Electromagnetism

Method for Optimisation of Mobile and Wireless Transceiver Design

By Haris Duric

Master's thesis report
Master's thesis performed from February 4th to November 29th 2002
Master's thesis performed at Philips Semiconductors Systems Lab Eindhoven
Master's thesis supervised by prof.dr.ir G. Brussaard, TU/e
ir. M. Vlemmings, PS-SLE

Summary

In recent years, number of devices incorporating a telecommunication link has increased drastically. There is still an increase of mobile and wireless applications within new, dynamic, telecommunication markets. At the moment, the development of a *radio frequency integrated circuit* (RF-IC) can easily take up to two or three years, which has become unacceptable, both in duration and expenses. Currently, at the Philips Semiconductors Systems Lab Eindhoven, a new approach to the design of RF front-ends is being developed. Instead of making dedicated solutions, as up to this moment was the case, a set of standardised and configurable building blocks will be introduced. This will shorten the time-to-market and also increase the reuse of the *intellectual property* (IP), however, at the cost of higher power dissipation and larger chip areas. The targeted demonstrator is a transceiver compliant to the IEEE 802.11a standard, a *wireless local area network* (WLAN) standard that is situated in 5 GHz frequency band.

One of the first steps in the development of an RF platform is the choice of the transceiver architecture and the building block partitioning. Each building block will have up to two adjustment inputs, voltage or current controlled. By varying these inputs, the performance of the building blocks can be changed. This performance will be fully characterised as function of the adjustment inputs and stored in the measurement tables. For given product specifications, an optimising routine will have to find the optimum composition of the building blocks and their optimal adjustment settings.
The design of the optimisation method is the subject of this Master's thesis as a part of the author's study at the Department of Electrical Engineering at the Eindhoven University of Technology. During the Master's thesis the modern transceiver architectures and the system design parameters have been studied. As a result, a number of the parameters quantifying the amplification, noise, non-linearity and power dissipation have been identified as the parameters to be used for the system optimisation.
Based on the system architecture, a system model has been determined. The relationship between the performance of the individual building blocks and the total system performance is defined by the system equations.
The cost function determines the optimum compromise between the system parameters. As a result of this work, a cost function has been designed to translate the relative importance of the system parameters into a mathematical function. To find the optimum system performance, the cost function is minimised.
Several optimisation techniques have been studied, of which two were selected for implementation and brief comparison: Genetic Algorithms and Differential Evolution (DE). The DE has been improved for this problem and built into a RF system optimisation tool. The experimental results in MATLAB have proven the feasibility of the optimisation of the receiver chain of million operating points within a few seconds of calculation time.
In future, the system calculations could be further extended by using more advanced modelling and including the effects of imperfections, such as the *local oscillator* (LO) phase noise.

Preface

This report describes the work I performed during my Master's thesis at the Philips Semiconductors Systems Lab Eindhoven (PS-SLE). A new approach to RF transceiver design and the challenging optimisation tasks have appealed to me from the beginning. The mixture of disciplines has offered me a chance to improve my abilities in conducting a project that combines RF system design, telecommunications, optimisation techniques and computer science.

I appreciate the support of many people without whose help I could have never finished this thesis. First of all I would like to thank professor Brussaard for his frequent visits here at the SLE and his interest in the work I was doing. Many thanks I own to Marc Vlemmings for his excellent supervision and readiness to assist me whenever I encountered problems. Furthermore, frequent discussions with Martin Barnasconi, Tom Buss and Arno Neelen have lead to some fruitful ideas, which have helped me in determining the directions of the project. Finally, I would like to thank all the other colleagues who have made my stay at the SLE a pleasant time.

Haris Duric
Eindhoven, November 29th 2002

Contents

1 Introduction

Recently, much effort has been put on the integration of the telecommunication front-ends. For a semiconductor company to follow the large market request in shortening the time-to-market constraint for new products, a systematic design methodology has to be followed, starting form top-down design followed by a bottom-up verification [11].

At Systems Laboratory Eindhoven, department of Philips Semiconductors, a project designated 'RF Platform' has been started to meet these needs. The idea is to define and make a set of reusable building blocks, each representing a certain function. One should think of a *low noise amplifier* (LNA), a mixer, a frequency synthesizer, a *variable gain amplifier* (VGA) or a further integration of few of these functions as an example of a building block. A platform will be defined where these building blocks are combined into a system at higher abstraction level. An RF front-end, which is a part of every transceiver, is an example of such a high-level functionality. By introducing this platform, a generic front-end will be introduced.

The building blocks will be made such that it will be possible to tune their performance. Instead of a dedicated solution, the building blocks will be adjustable, covering a range of operating points.

The RF platforms project could be split into three major parts:

1. Mapping of the standard specifications to the building block specifications.
   Specifications of popular wireless and mobile standards can be projected to a multidimensional axis, where the extremes span the maximum design space. Examples of the quantities among the axes are frequency, temperature, bit error rate, maximum transmitter power, receiver sensitivity, etc. It will not be possible to cover this entire space with one set of building blocks.

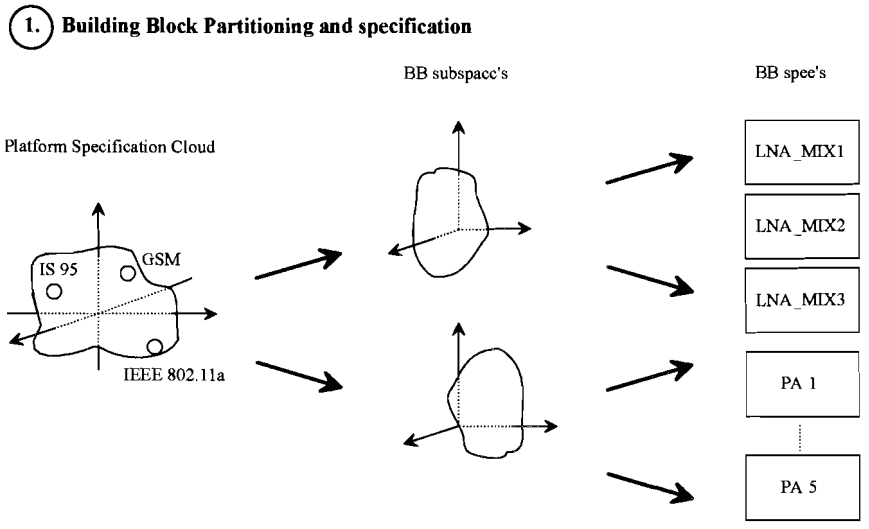


Figure 1: From the 'Platform Specification Cloud' to the building block specifications

Obviously, the different frequency allocations of different systems with their physical bandwidth limitations would not allow for such a wideband operation. The challenge is then to split the design space into a minimum number of building blocks that will be able to cover the entire space.

Figure 1 depicts this process. The entire design space is split into building block subspaces. Each subspace is than split into a number of different building block specifications. The building blocs will be configurable so they will not represent one solution, a point in space, but will be able to move within a subset of solutions.

2. Designing, manufacturing and characterising the building blocks.
   Second trajectory is to actually design and manufacture the building blocks. The performance of each building block will be made adjustable by two inputs. Afterwards, the blocks will be measured and their performance will be fully characterised as a function of the adjustment inputs.

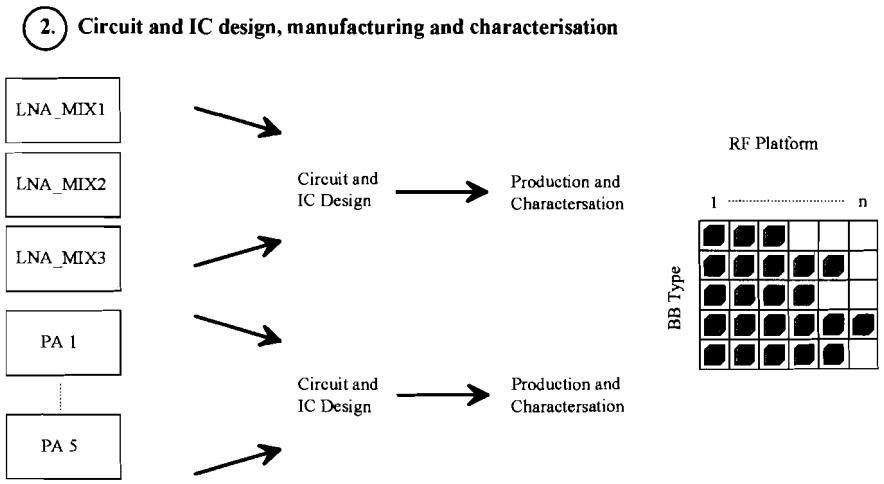


Figure 2: From the building block specifications to the building block die's

3. Optimisation of the system for a specific telecom standard and the verification of the performance.
   When a real-life system with defined specifications needs to be made, the optimum composition of the building blocks for that particular system must be determined. The building blocks are adjustable, so the optimum setting of the adjustment inputs has to be found. This graduation project concentrates on the optimisation part of this third trajectory, including the system modelling and system calculations. A time effective method needs to be found for finding the optimum system and its optimum operating point out of a huge number of alternative systems.
   In a later stage, the system will need to be assembled and the performance, as predicted by the optimisation method, needs to be verified.

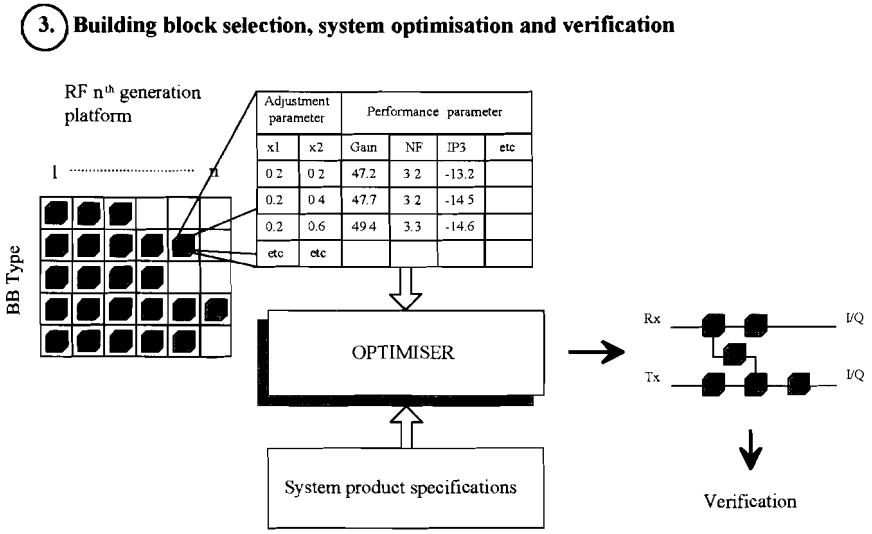


Figure 3: From die's to the optimal front-ends

2
Modern Transceiver Architectures

The task of the *radio frequency* (RF) part of the transceiver is to convert the antenna radio signals down to baseband and vice versa. The receiver part of the front-end is responsible for filtering, amplification and down-conversion, where the unwanted signals are suppressed and the wanted signals are amplified to a sufficient *signal-to-interference* (S/I) level and converted to sufficiently low frequencies to be sampled by a A/D converter.

Nowadays, most standards use digital modulation techniques where the actual modulation and demodulation is carried out by means of *digital signal processing*. If a form of *quadrature modulation* scheme is used, there is a need for distinguishing the *in-phase* (I) and *quadrature* (Q) parts of the signal since they are carrying different information.

In this chapter, only the receivers will be described since the transmitters basically perform the inverse operations.

2.1 Heterodyne Receiver

This classic receiver architecture is characterised by down-converting the signals in two steps; first form RF to some convenient frequency band, called the *intermediate frequency* (IF) band, and then from IF to baseband. This allows for extra filtering and amplification stages, which improve the signal quality but also increase the receiver's complexity and costs.

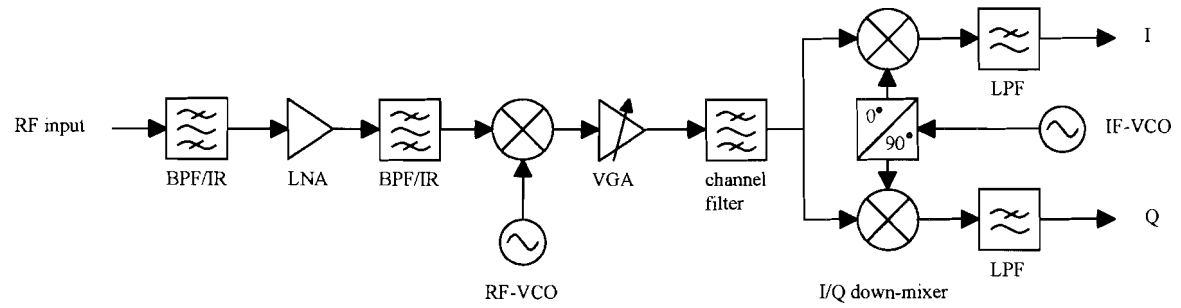


Figure 3: Heterodyne receiver architecture

The receiver can be operated in two modes: high-side and low-side injection, referring to the position of the *local oscillator* (LO) frequency with respect to the receive band. The principle of the high-side injection is depicted in Figure 4.

The IF band is situated around $f_{IF} = f_{LO,RF} - f_{RF}$. The transfer, $|H_{BPF}(f)|$, of the input *band-pass filter* (BPF) is also shown in Figure 4. The wanted signal, at f_{IF}, may be corrupted by the image signal, f_{image}, if the image is not suppressed enough by the BPF. The down-converted part of the image signal, conform $f_{IF} = f_{image} - f_{LO,RF}$, is then superimposed on the down-converted wanted signal.

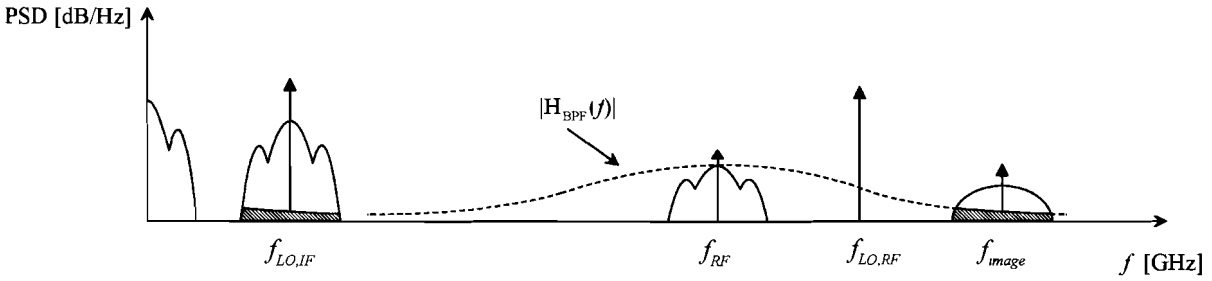


Figure 4: Power spectrum of heterodyne receiver

2.2 Zero-IF Receiver

This homodyne receiver architecture, shown in Figure 5, is basically the same as the heterodyne receiver with the difference that the down-conversion is performed in one step from RF to baseband. That is why it is often referred to as *direct conversion* receiver.

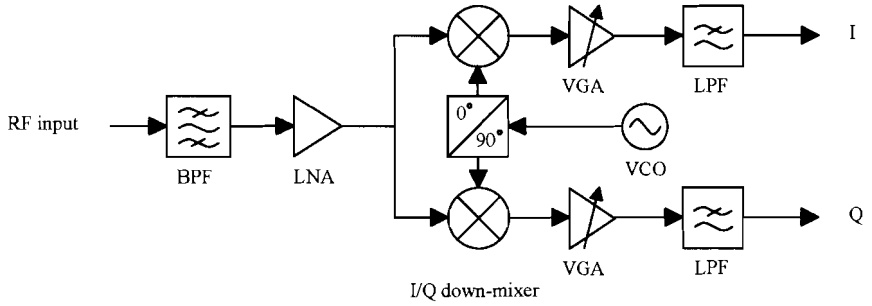


Figure 5: Homodyne receiver architecture

The receiver principle is shown in Figure 6. The image frequencies are located at $2 \cdot f_{LO}$. Usually, these frequencies are far in the spectrum and are suppressed enough by the BPF, so they do not corrupt the wanted signal.

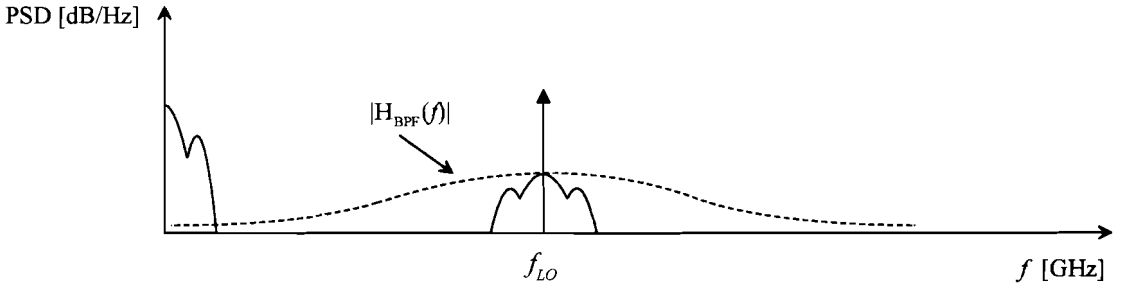


Figure 6: Spectrum of homodyne receiver

2.3 Near-Zero IF Receiver

Another important direct conversion receiver also uses homodyne architecture [5], as shown in Figure 5. The f_{LO} is generally slightly higher or lower than the RF band, which is converted in one step to the baseband, asymmetrical to the DC. The frequency offset is compensated and I/Q signals are restored by means of digital signal processing.

3
Building Block Partition

The architecture of choice for the RF-platforms project is the *Zero-IF* architecture for the receiver and the transmitter. A study has been performed where decisions were made for the architecture [17] and the building block partitioning. Both, they influence the modelling of the receiver and the transmitter chains and therefore are of interest for this thesis.

The building blocks have been partitioned according to their functions, physical position, semiconductor process technology, etc., basically using the following criteria [18]:

1. Full-duplex operation possibility.
2. Process technology aspects.
3. Reuse and flexibility.
4. Parasitic effects (crosstalk, pulling) and temperature effects.
5. Superheterodyne architecture should not be excluded.
6. Number of interconnections between blocks.

This has lead to division of the front-end into following five building blocks:

1. LNA – mixer
   This building block will combine the LNA, frequency divider with 0° and 90° phase outputs and the balanced I/Q mixer.

2. VCO – PLL
   This building block includes the *voltage-controlled oscillator* (VCO) together with the *phase lock loop* (PLL) circuitry.

3. IF – baseband
   This building block contains the VGA and baseband filters in the receiver path and amplifiers and filters in the transmitter path.

4. Upmixer – driver
   This building block combines the up-mixer and the driver for the *power amplifier* (PA).

5. Power Amplifier
   This building block contains the PA and the filter.

The transceiver architecture and building block partitioning are shown in Figure 7.

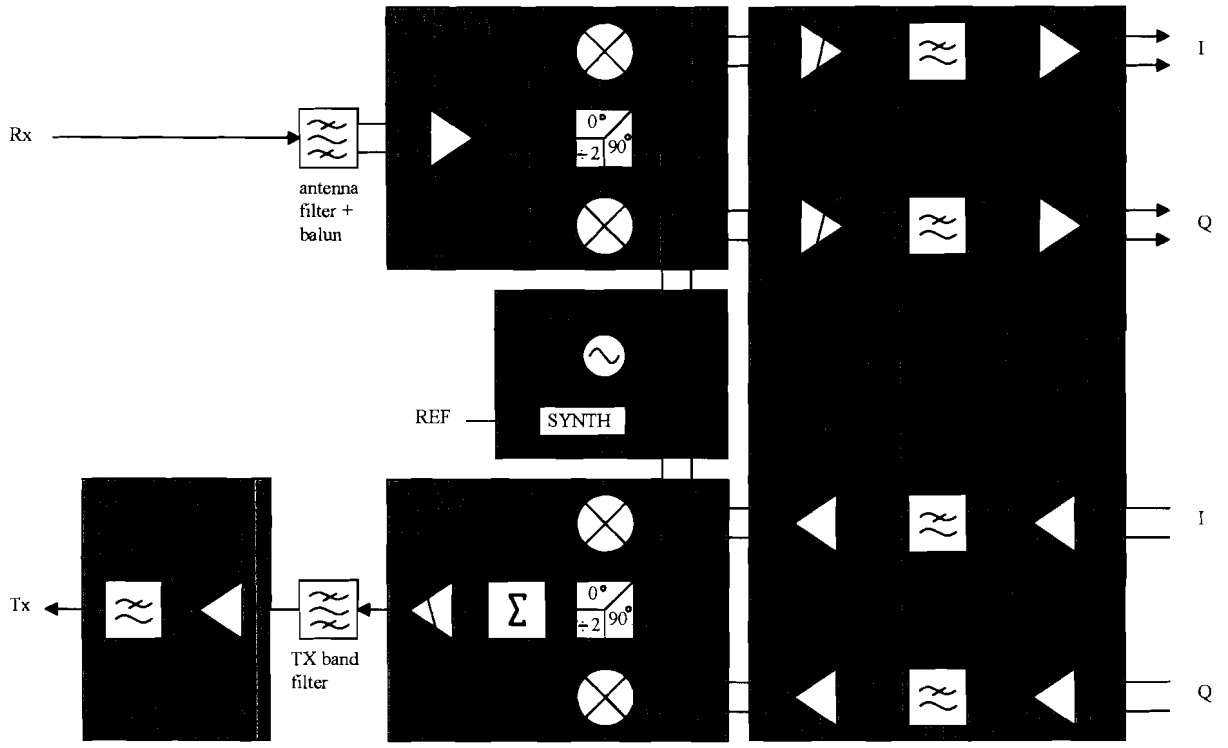


Figure 7: The RF-platform transceiver architecture

Each building block will have up to two adjustable inputs that can be used to tune the performance (not shown in Figure 7). The IF-baseband building block will be physically one block, but the receive-and transmit - circuitry will each have their own inputs.

In continuation of this thesis, the building blocks will be regarded as black boxes since their specific content is not important for this work. Their performance will be measured and therefore their behaviour will be known.

3.1 Building Block Measurement Parameters

After the specification, circuit design, IC-layout and manufacturing steps, the building blocks will be fully characterised and their performance will be stored in measurement tables. Since there are two inputs per building block, denoted x_1 and x_2 in Figure 8, each varying in ten steps, there will be 100 different operating points per building block. The *RF in* and *RF out* simply indicate the signal flows and do not necessarily represent asymmetrical terminals. The adjustment inputs can be either voltage or current controlled.

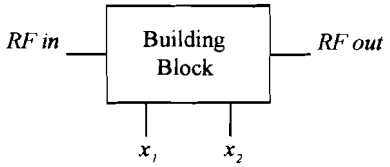


Figure 8: Building block abstraction

The VCO-PLL building block is different in the way that it has DC-in and RF-out terminals. What is of interest for this thesis is the influence of the building blocks in the signal path. In section 4.3 it will be discussed that the VCO-PLL may as well be represented by Figure 8.

The choice of the measurement parameters has influence on the system optimisation. A minimum set of mutually independent parameters is required. How the different parameters influence the system performance and what their choice is based on, will be discussed in chapters 4 and 5.

The parameters to be measured are (except VCO-PLL):

1. Gain (G)
2. Noise Figure (NF)
3. Input-referred 1-dB compression point (IP_{1dB})
4. Input-referred 2nd order intercept point (IIP2)
5. Input-referred 3rd order intercept point (IIP3)
6. DC current consumption (I_{DC})

Figure 9 shows an illustration of the (hypothetical) building block parameters as a function of the adjustment inputs x_1 and x_2.

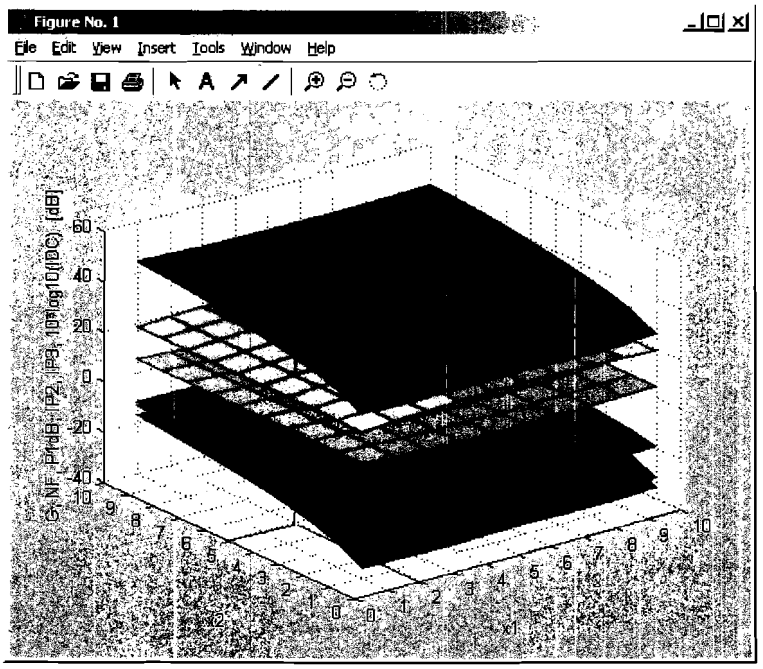


Figure 9: Example of the building block parameters as function of adjustment pins (not to scale)

A unique combination of x_1 and x_2 denotes an operating point. So at each operating point, a vector of n_{param} parameter values is appointed.

$$X = (x_1, x_2)^T \rightarrow (p_1, \ldots, p_{n_{param}})^T \tag{1}$$

At the present there are six parameters to be measured. A vector, e.g. $(2,5)^T$, as shown in Figure 9, points to a 6-dimensional building block parameter vector, which describes the performance at that operating point.

4
Models and System Calculations

A telecommunications system is often composed of several stages in cascade. To be able to predict and analyse the total equivalent system behaviour, system calculations are done.

4.1 Gain - Noise Model

An RF component may be modelled as a black box with a certain power gain and circuit generated noise density N, as shown in Figure 10.

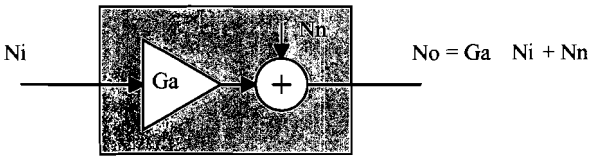


Figure 10: The gain-noise model

The total noise power density at the output of the component is the sum of the input noise power density N_o multiplied by the power gain Ga, and the noise introduced by the circuit.

$$No = Ga \cdot Ni + Nn \qquad \left[\frac{W}{Hz}\right] \tag{2}$$

The noise factor F is defined as the ratio of the input-referred noise density to the thermal noise density at the input of the device, at temperature of 290 K.

$$F = \frac{No}{Ga \cdot Ni} = 1 + \frac{Nn}{Ga \cdot Ni} \tag{3}$$

Widely used is the noise figure, which is the noise factor in units of dB:

$$NF = 10 \cdot \log(F) \qquad [dB] \tag{4}$$

The equivalent noise figure of N cascaded stages is given by the well-known Friis equation,

$$F = F_1 + \frac{F_2 - 1}{Ga_1} + \frac{F_3 - 1}{Ga_1 Ga_2} + \ldots + \frac{F_N - 1}{Ga_1 Ga_2 .. Ga_{N-1}} \tag{5}$$

4.2 Non-linearity Model

A problem with wireless and mobile applications is when a link is calculated, the exact magnitude of the signals is not known. Because the distance between the transceivers is variable, the received power can vary strongly while the receiver should be able to handle all these signal levels. If the receiver is too close to the signal source it could be driven into non-linear operation where harmonic distortion and intermodulation products are produced.

An RF component may be modelled by the following expression,

$$y[x(t)] = \beta_1 \cdot x(t) + \sum_{i=2}^{\infty} \beta_i \cdot [x(t)]^i \tag{6}$$

where $x(t)$ is the input waveform and β_i is the coefficient for the i^{th} order non-linearity.

There are three commonly used linearity measures, *1-dB compression point* (P_{1dB}), *second order intercept point* (*IP2*) and the *third order intercept point* (*IP3*).

4.2.1 1-dB Compression Point (P_{1dB})

The 1-dB compression point of a gain stage is defined as the input power level at which the power at the output of the stage, P_o, has degraded by 1 dB respective to the linear output. This is shown in Figure 11. P_{1dB} is often used as a measure of maximum applicable input signal.

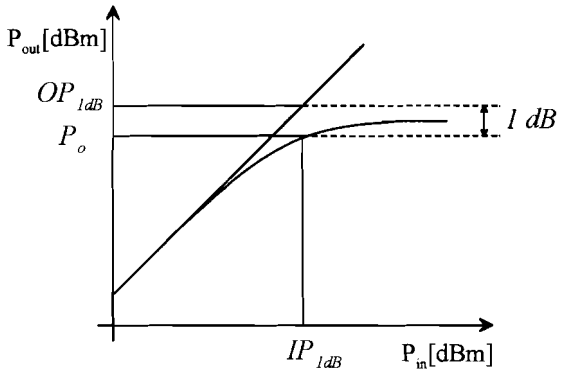


Figure 11: 1-dB compression point curve

It is desirable to predict the equivalent system P_{1dB} by using only the individual P_{1dB}'s of each stage and their respective gains. The linear power level, i.e. not in dB, at the input of N stages at which the system is at 1-dB compression is given by

$$\frac{1}{ip_{1dB}} = \frac{1}{ip_{1dB,1}} + \frac{Ga_1}{ip_{1dB,2}} + \frac{Ga_1 Ga_2}{ip_{1dB,3}} + \ldots + \frac{Ga_1 Ga_2 .. Ga_{N-1}}{ip_{1dB,N}} \qquad \left[\frac{1}{W}\right] \tag{7}$$

were G_a is the *small-signal* gain (and not the large signal gain which is 1 dB lower at P_{1dB}).

For this thesis a convention is introduced: uppercase symbols, e.g. IP_{1dB}, are used to denote the values in dB's and lower case, ip_{1dB}, are representing the linear values. This is different for the gain where the gain in dB is represented by symbol G, and its linear value by Ga. NF is the noise figure in dB and F is the linear value. The relationships between the dB and the linear values can be found in Appendix C.

The derivation of equation (7) is based on the assumption that all the individual building blocks as well as the entire system can be modelled by the cubic non-linearity. The derivation can be found in Appendix A.

The relationship between the input and the output P_{1dB} is given by

$$op_{1dB} = Ga \cdot ip_{1dB} \qquad [W] \tag{8}$$

4.2.2 Second Order Intercept Point (IP2)

The second-order intercept point is an imaginary point that is obtained by the extrapolation of the second-order intermodulation curve to the point where it intercepts the linear curve, shown in Figure 12.

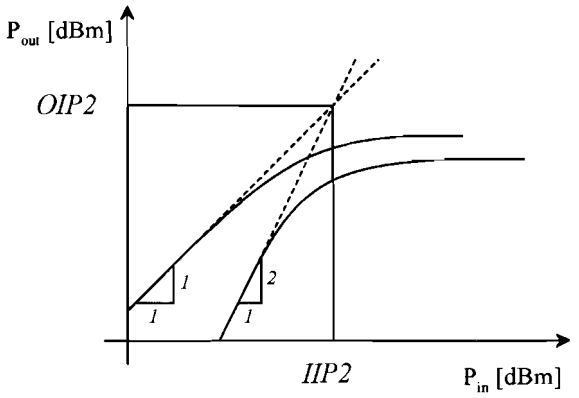


Figure 12: Second order intercept point

The following relationship allows for predicting the magnitude of 2nd order intermodulation products at the output of a stage for a given signal magnitude at the input.

$$OIM2 = 2 \cdot Po - OIP2 \qquad [dB] \tag{9}$$

The equivalent system input-referred *IP2* of *N* cascaded stages can be calculated from

$$\frac{1}{\sqrt{iip2}} = \sqrt{\frac{1}{iip2_1}} + \sqrt{\frac{Ga_1}{iip2_2}} + \sqrt{\frac{Ga_1 Ga_2}{iip2_3}} + \ldots + \sqrt{\frac{Ga_1 Ga_2 .. Ga_{N-1}}{iip2_N}} \qquad \left[\frac{1}{\sqrt{W}}\right] \tag{10}$$

The relationship between input and output *IP2* is given by

$$oip2 = Ga \cdot iip2 \qquad [W] \tag{11}$$

4.2.2 Third Order Intercept Point (IP3)

The 3rd order intercept point is another linearity measure that is also obtained by extrapolation, shown in Figure 13.

The relationship between the power of the fundamental at the output of the stage and the power of the 3rd order intermodulation products is given by

$$OIM3 = 3 \cdot Po - 2 \cdot OIP3 \qquad [dB] \tag{12}$$

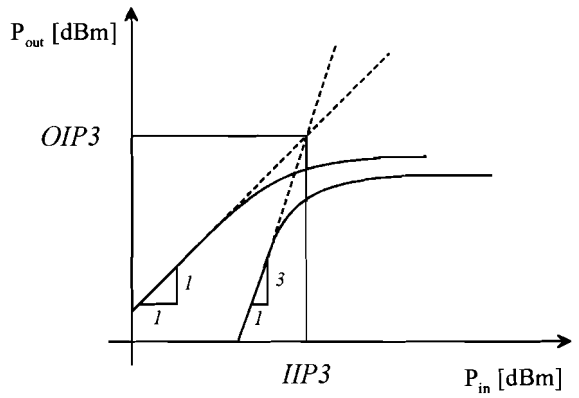


Figure 13: Third order intercept point

The equivalent system input-referred *IP3* of *N* cascaded stages is given by

$$\frac{1}{iip3} = \frac{1}{iip3_1} + \frac{Ga_1}{iip3_2} + \frac{Ga_1 Ga_2}{iip3_3} + \ldots + \frac{Ga_1 Ga_2 .. Ga_{N-1}}{iip3_N} \qquad \left[\frac{1}{W}\right] \tag{13}$$

and the relationship between input and output *IP3* is given by

$$oip3 = Ga \cdot iip3 \qquad [W] \tag{14}$$

From the derivation of the equations in Appendix A, the relationship between the *IIP3* and IP_{1dB} can be deduced. For the input voltages holds

$$\frac{A_{IP_{1dB}}}{A_{IIP3}} = \sqrt{\frac{\frac{4}{3}\left(10^{-\frac{1}{20}} - 1\right)\frac{\alpha_1}{\alpha_3}}{\frac{4}{3}\frac{\alpha_1}{\alpha_3}}} \approx 0.3298 \tag{15}$$

which yields for powers

$$IP_{1dB} \approx IIP3 - 9.6dB \tag{16}$$

4.3 VCO-PLL Model

According to the current architecture choice and the building block partitioning there are maximum three building blocks in a receiver or the transmitter signal path. The VCO-PLL building block is not encountered directly in the signal path but does certainly influence the system's performance and will have to be modelled.

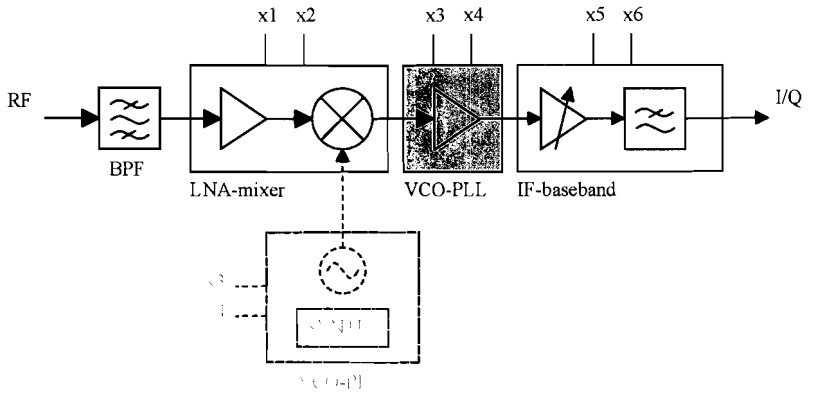


Figure 14: VCO-PLL modelling in the RF path

The conversion gain of the mixer depends on the output power of the VCO. Then the VCO-PLL building block may be modelled as an ideal gain stage between the LNA-mixer and the IF-baseband. The gain of the grey block, in Figure 14, is a function of the adjustment inputs of the VCO-PLL block.

According to [19], the oscillator phase noise causes the degradation of the SNR (hence BER) in an OFDM system. The phase noise of the building block may vary with the variation of the adjustment inputs and therefore could be used as a parameter for the optimisation. However, these effects are presently not covered by this work.

There are also other effects, such as the interferers- and LO self-mixing due to imperfect isolation [2]. This is described in Figure 15. These requirements are often specified in the standard and need to be met on the building block level. The effects shown in Figure 15 are determined by the physical positions and the geometry on the die itself. The physical properties cannot be changed by the adjustment inputs and are fixed for a building block. However, the effect on the signal could be modelled, which is not yet covered by this work.

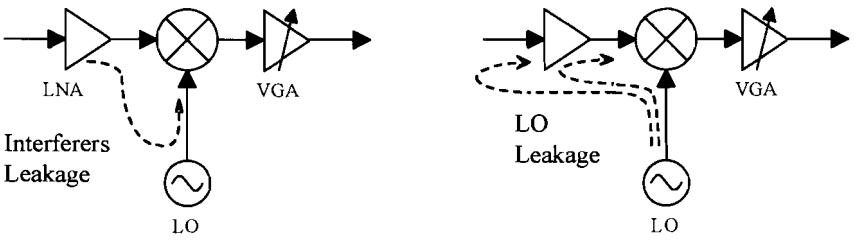


Figure 15: Interferers- and LO self-mixing

5
Optimisation Parameters and the Cost Function

Which parameters will be used for the system optimisation and what this choice is based on will be discussed in this chapter. A universal set of parameters, which are mutually as independent as possible, needs to be chosen in order to provide the highest possible flexibility.

5.1 Optimisation Parameters

Different standards use different parameters to describe the system requirements. One standard could provide the spectral masks and the sensitivity levels whereas another could demand, among others, some maximum *bit error rate* (BER) and minimum *adjacent channel power rejection* (ACPR). This diversity of parameters describing a wide range of quantities can be mapped to a basic set of system parameters.

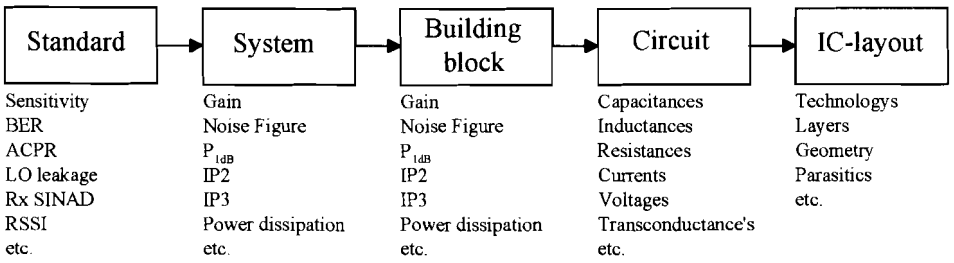


Figure 16: System abstraction levels

Except for the frequency translation, a front-end performs the following operations on the signal:

1. Amplifies the signal
2. Adds some noise
3. Adds distortion and intermodulation products

which are quantified by the following parameters:

1. Gain
2. Noise Figure
3. 1-dB compression point, 2^{nd} and 3^{rd} order intercept points, etc.

Consider the hierarchical representation of different abstraction levels, shown in Figure 16. The latter mentioned parameters are used to characterise the whole system and are also used to characterise the individual building blocks. This is a strong argument to choose these parameters as the parameters for the optimiser. The optimisation loop will circle between the building block and the system level, as shown in Figure 16.

The gain and noise figure are needed for the basic link budget calculations when designing the system.

The different linearity measures affect the system performance as follows. For Zero-IF receivers, the presence of two strong interferers near the receive band, shown in Figure 17, would result in an interfering signal, at $|f_1 - f_2|$, in baseband. Also a DC product is generated as a result of 2nd order intermodulation. This can cause problems if an interferer is switching on and off, which would generate a spectrum of *sin(x)/(x)* form around the DC.

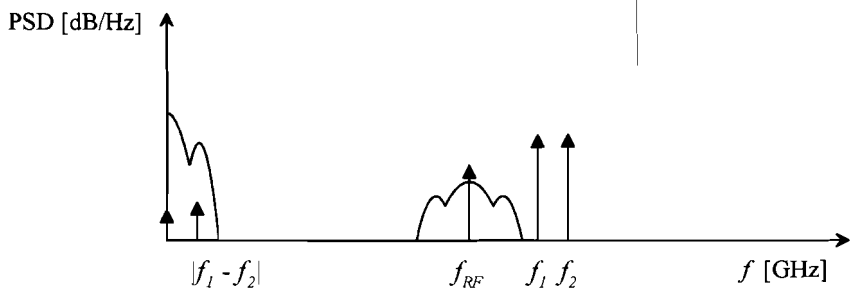


Figure 17: Influence of the IP2 in Zero-IF receivers

Similarly, the two strong interferes near the receive band would cause the in-band distortion due to the 3rd order intermodulation, as shown in Figure 18. When down-converted, the intermodulation product will be superimposed to the wanted baseband signal.

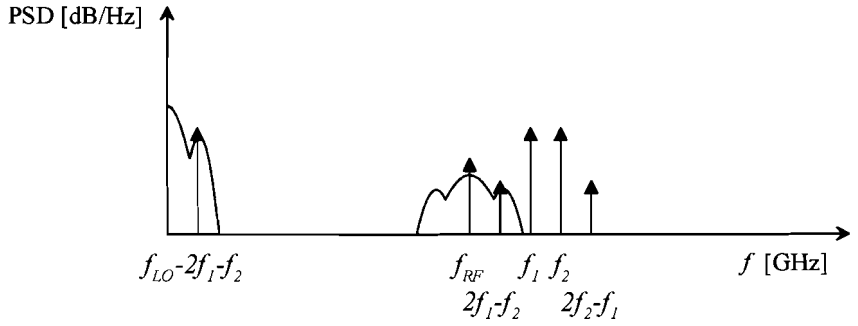


Figure 18: Influence of the IP3 in Zero-IF receivers

5.2 Cost Function

In general, an objective function is a function that is minimised or maximised. In this case the 'quality' of a system is measured by its performance and the objective function will be defined as a cost function. Minimising the costs optimises the system performance. The design of the cost function is described step-by-step.

Again, the cascaded noise factor is given by

$$F = F_1 + \frac{F_2 - 1}{Ga_1} + \frac{F_3 - 1}{Ga_1 Ga_2} + \ldots + \frac{F_N - 1}{Ga_1 Ga_2 .. Ga_{N-1}} \tag{17}$$

and the cascaded linearity measure *iip3*

$$\frac{1}{iip3} = \frac{1}{iip3_1} + \frac{Ga_1}{iip3_2} + \frac{Ga_1 Ga_2}{iip3_3} + \ldots + \frac{Ga_1 Ga_2 .. Ga_{N-1}}{iip3_N} \tag{18}$$

To increase the total system noise and linearity performance, the noise factor, F, needs to be minimised while simultaneously maximising the *iip3*. These are two contradictory demands, which can be understood from equations (17) and (18):

- Minimum overall F requires minimising individual noise factors (especially F_1) and maximising individual gains (especially Ga_1).

- Maximum overall *iip3* requires maximising individual *iip3*'s (especially $iip3_N$) and minimising individual gains (especially Ga_1).

To quantify the trade-off between the noise and linearity performance, a figure of merit called IP3-to-noise-figure ratio may be defined as [20]

$$IP3NR = \frac{iip3}{F} \tag{19}$$

There exists an optimal gain distribution that maximises *IP3NR*, which can be found by solving

$$df = \frac{\partial f}{\partial x_1}dx_1 + \frac{\partial f}{\partial x_2}dx_2 + \ldots + \frac{\partial f}{\partial x_n}dx_n = 0 \tag{20}$$

where $f = IP3NR$ and x_1 to x_n are Ga_1 to Ga_N.

The *IP3NR*, as defined in equation (19), assumes the equal importance of the noise and the linearity performance. In a real system this importance can be different. At instance, for the receiver chain, where the signals are weak, the noise may be as important as the linearity. But for a transmitter, when dealing with strong signals, the noise is of much less interest than the linearity. To be able to perform this trade-off in a controlled manner the weighting factors are introduced.

In units of dB, equation (19) is given by

$$IP3NR_{dB} = 10\log_{10}(iip3) - 10\log_{10}(F) \tag{21}$$

which allows for weighting the noise figure and *IIP3*

$$IP3NR_{dB,weight} = w_1 \cdot 10\log_{10}(iip3) - w_2 \cdot 10\log_{10}(F) \tag{22}$$

Normally in a system, it is required to minimize the F and to maximise the *IP3*, which can be combined in a single expression $maximise(IP3NR_{dB,weight})$, since

$$maximise(f(x)) = minimise(-f(x)) \quad \Leftrightarrow \quad minimise(f(x)) = maximise(-f(x)) \tag{23}$$

The weighted $IP3NR_{weight}$, again expressed as a linear factor yields

$$IP3NR_{weight} = \frac{iip3^{w_1}}{F^{w_2}} \tag{24}$$

Since the optimisation statement will be defined as minimisation rather then maximisation, a simple cost function is given by

$$f_{cost} = \frac{1}{IP3NR_{weight}} = \frac{F^{w_2}}{iip3^{w_1}} \tag{25}$$

which means that higher the value of the $IP3NR_{weight}$, the lower the costs, so better the system.

The total cost function should include not just the non-linearity and noise, but also the gain and the total power consumption. Some other cost-affecting parameters may also be included, such as chip costs per mm^2, but they are not relevant at this time. However, these kind of parameters are constant and do not vary with the adjustment parameters.

The trade-off part of the cost function may be formulated as follows:

$$f_{cost}(g(h(X))) = \frac{F^{w_2} \cdot I_{DC}{}^{w_3}}{\left(b \cdot iip3 + \bar{b} \cdot ip_{1dB}\right)^{w_1} \cdot iip2^{w_4} \cdot Ga^{w_5}} \tag{26}$$

where

$h : X \longrightarrow P$

$g : P \longrightarrow P_{sys}$

$X = (x_1, \ldots, x_{n_{param}})^T$

$P = (p_1, \ldots p_{n_{param}})^T$

$P_{sys} = (p_{sys,1} \ldots p_{sys,n_{param}})^T$

$b = 0 \vee 1$

The function h is defined by the measurements. For each vector from X there are six performance parameter values from P. The function g is defined by the system equations. For a building block performance parameter vector from P there is a system performance parameter P_{sys}.

Since there is a strict relationship between the $IP3$ and P_{1dB},

$$IP3 = P_{1dB} + 9.6\,dB \tag{27}$$

there is no sense in including both the parameters. Their dependency is well defined. Using the parameter b in equation (26) either one of two linearity measures can be selected for the optimisation. Nevertheless, if the cubic non-linearity model shows not accurate enough for modelling the P_{1dB} and $IP3$, than the system equations need to be revised and the cost function should be adapted to accommodate this.

The third important linearity measure, $IP2$, cannot strictly be related to $IP3$ and P_{1dB} and is therefore included as a separate parameter.

The DC current consumption, I_{DC}, is placed in the numerator indicating the higher costs for the higher current.

The gain has been placed in the denominator, meaning the higher the gain, the lower the costs. The gain is usually not a critical parameter in which case it could be specified as a constraint and not as an optimisation parameter (by setting w_5 equal to zero).

5.2.1 Constraint Function

Very often, the actual value of a system parameter is not important as long as certain constraint is satisfied. For example, a system may be optimised for the noise and linearity performance while demanding that the total power consumption of the system stays below a certain value.
In this investigation, a soft-constraint (penalty) approach will be followed. The constraint function introduces a distance measure from the feasible region, but is not used to reject unfeasible solutions, as it is in the case of hard-constraints.
The constraint function is given by:

$$f_{constr} = \begin{cases} 0 & for\ P \le P_C \\ a \cdot (P - P_C)^2 & for\ P > P_C \end{cases} \tag{28}$$

where P is the parameter value and the P_C is the respective parameter constraint. Changing the value of a allows for scaling of f_{constr}, as shown in Figure 19.

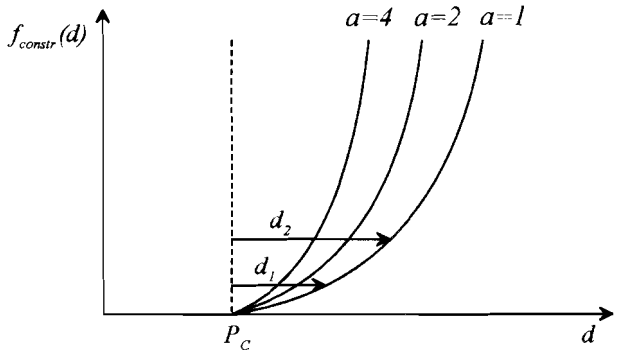


Figure 19: Squared distance penalty constraint

The total cost function consists of two parts:

- the trade-off section
- the constraints section

The cost function is defined as follows:

$$f_{cost}(g(h(X))) \hat{=} \begin{cases} \dfrac{F^{w_2} \cdot I_{DC}{}^{w_3}}{\left(b \cdot iip3 + \bar{b} \cdot ip_{1dB}\right)^{w_1} \cdot iip2^{w_4} \cdot Ga^{w_5}} & for \ P_i \le P_{Ci} \\ \dfrac{F^{w_2} \cdot I_{DC}{}^{w_3}}{\left(b \cdot iip3 + \bar{b} \cdot ip_{1dB}\right)^{w_1} \cdot iip2^{w_4} \cdot Ga^{w_5}} + \sum_{i=1}^{n_{param}} a_i (P_{Ci} - P_i)^2 & for \ P_i > P_{Ci} \end{cases}$$

where

$h : X \longrightarrow P$

$g : P \longrightarrow P_{sys}$

$X = (x_1, \ldots, x_N)^T$

$P = (p_1, \ldots p_{n_{param}})^T$

$P_{sys} = (p_{sys,1} \ldots p_{sys,n_{param}})^T$

$b = 0 \vee 1$

$a_i \in \mathbb{R}$ (29)

5.2.2 Cost Function Biasing and Normalising

In the trade-off section of the cost function the different quantities are traded. In order to normalise for the units and compensate for the scale of the absolute value, the cost function is biased.

For example, the *IIP3* of a building block may vary as function of x_1 and x_2 between –30 and -12 dBm whereas the *NF* of the same block may vary between 3 and 6 dB. The respective mean values represented as linear factors are $\approx 7.08 \cdot 10^{-3}$ mW and 2.82 (dimensionless). To let both the parameters to have a comparable influence on the cost function given in equation (20) and to normalise for the units, the weight factors are defined as

$$w_i = \lambda_i \cdot w_i' \quad (30)$$

where λ_i is in reciprocal units of the corresponding parameter.

6
Time Complexity Estimation

To find the optimal composition of the building blocks and the corresponding operating point, the cost function needs to be minimised. Before the appropriate algorithm can be proposed there should be insight in the complexity of the problem. Because the large numbers are at stake, careful examination needs to be performed in order to neither overestimate or to underestimate the problem.

Let's consider a system of k building blocks for deriving the equations. There is n number of different operating points per building block

$$\frac{\#\ of\ operating\ points}{building\ block} = adj1 \cdot adj2 = n \tag{31}$$

where *adj1* and *adj2* are the number of steps per corresponding adjustable parameter. In continuation of this chapter, n will be referred to as 'adjust-product'.

For k number of building blocks, the number of different operating points is given by

$$\#\ of\ operating\ points = \prod_{i=1}^{k} n_i \tag{32}$$

where i is the index of a building block.

Equation (32) is plotted in Figure 20 on the logarithmic scale for different values of k and with adjust-product as parameter. For example, if there are $k=5$ building blocks in a system and each of the adjustment inputs is varied in 10 steps (*adj1*adj2 = 100*), then there are 10^{10} different operating points.

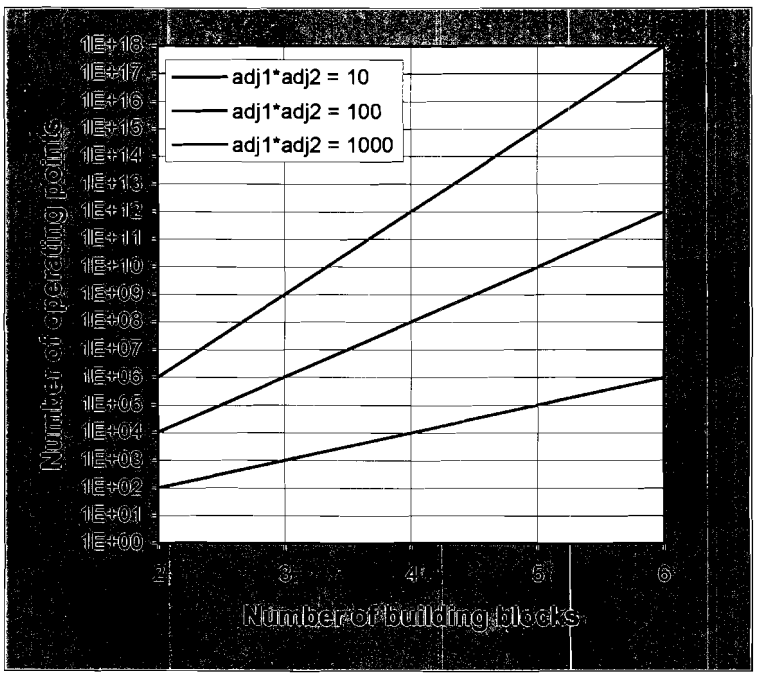


Figure 20: Number of operating points as function of the number of building blocks and with adjust-product as parameter

The next question is how long it would take to calculate the system performance for all the operating points.

Now the number of different operating points is known and if the time to calculate the system performance per operating point is also known, the total calculation time will be known.
The number of operating points can be calculated exactly, the calculation time per operating point is dependent on two quantities:

1. the number of the system parameters to be calculated and the system equations,
2. the operating system, CPU speed, used software, etc.

The number of the basic operations is counted for 5 system parameters: Gain, Noise Figure, P_{1dB}, *IP2*, and *IP3*. The basic operations that are counted are adding (or subtraction), and multiplication (or dividing).

Table 1: Noise figure basic operations

Noise Figure					
k (building blocks)	2	3	4	5	6
M (multiplication)	1	3	5	7	9
A (adding)	2	4	6	8	10

Table 2: 1-dB compression point basic operations

P1dB					
k (building blocks)	2	3	4	5	6
M (multiplication)	2	4	7	11	16
A (adding)	1	2	3	4	5

For *IP2* en *IP3* calculations is the same number of calculations needed as for P_{1dB}. The Gain is calculated in k multiplications. The I_{DC} is calculated in k additions.

Table 3 shows the total number of basic operations per operating point for the different number of building blocks.

Table 3: Number of basic operations per operating point

Operating point					
k (building blocks)	2	3	4	5	6
M (multiplication)	9	18	30	45	63
A (adding)	7	12	19	25	31

Estimation of time it would take to perform an operation is a difficult task since it depends on the number of factors such as the number format (integer, floating point), programming language, CPU, operating system, etc. A pragmatic approach to this problem is to perform a large number of basic operations on a real life machine and measure the elapsed time. For this purpose the following MATLAB script is executed on a network server.

The script fills two 1000x1000 matrices (each 1 million elements) with random numbers, starts a timer, multiplies the matrices and returns the elapsed time. The class of A and B is double array.

```
A = rand(1000,1000);
B = rand(1000,1000);
Tic;
C = A*B;
Toc;
```

It takes around 10 seconds to execute this script on the respective server. Result is also a matrix of 1000x1000 elements. There are in order of 10^3 of operations per element, multiplied by 10^6 elements, yields 10^9 operations for the whole matrix. This results into 10^{-8} sec calculation time per basic operation.

Figure 21 shows the expected calculation time for a different number of building blocks and different adjust-products. For the case of 5 building blocs and adjust product of 100, it would take, in baste case, almost two hours to calculate all the possibilities,

$$\frac{70 \cdot 10^{10} \cdot 10^{-8}}{3600} = 1.9 \; hours \tag{33}$$

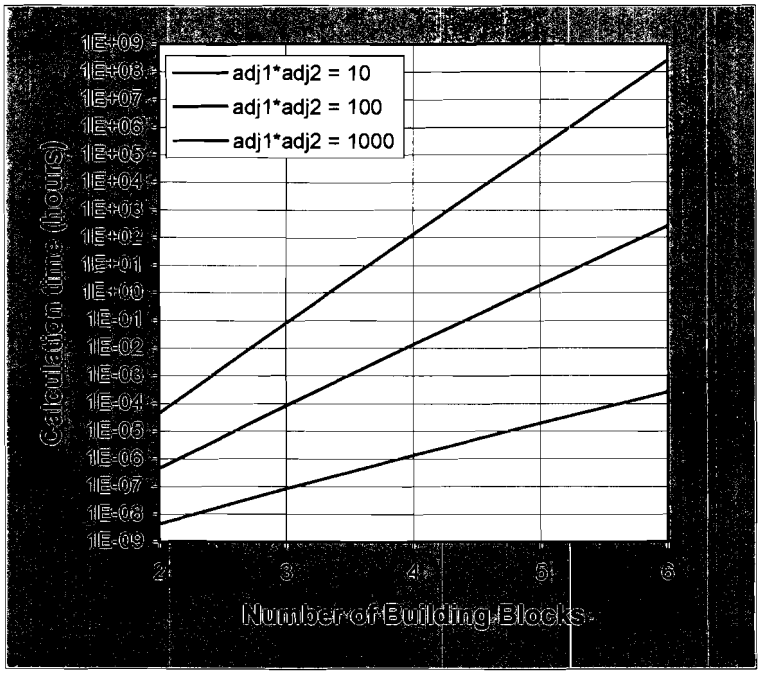


Figure 21: Calculation time as function of the number of building blocks and with adjust-product as parameter

One should be very careful with interpreting such a kind of graphs. If there were five building blocks and about 15 measurement points per parameter, instead of 10, the calculation time would be in order of days. But then, for five building blocks and the adjust-product of 1000 (approximately 33 measurement points per parameter) it would take more than 21 years to calculate all the possibilities.

7
Optimisation Techniques

The number of operating points, i.e. the number of alternative systems that need to be calculated, can easily become unmanageable, as shown in previous chapter. This holds for the case that all possible combinations are calculated. But in fact, there will be one optimal solution that has to be found in a minimum number of cost function (and thus also system) calculations. In this chapter, a summary of the techniques to do this and the classification of them will be given. At the end of the chapter, the choice of the most suitable technique for this problem will be discussed.

7.1 Optimisation Basics

Optimisation problems are made up of three basic ingredients:

- An **objective function** that is to be minimised or maximised. Very often a **cost function** is defined that incorporates the optimisation parameters, which is then minimised. For instance, in a transceiver design it could be desirable to maximise the linearity of the system and at the same time minimise the noise contribution.

- A set of **unknowns** or **variables** that affect the value of the objective function. These variables could be, for instance, the gains, noise figures and the IP3's of the individual stages in the system.

- A set of **constraints** that allow the unknowns to take on certain values but exclude others. For example, when a system is optimised for the linearity and noise figure, the limitation on maximum power dissipation of the total transceiver could be specified as a constraint.

The optimisation problem is then:

Find values of the variables that minimise or maximise the objective function while satisfying the constraints.

Mathematically we can write the constrained minimisation problem as follows [21],

$$\begin{array}{ll} \text{minimise} & f(X) \\ \text{subject to} & X \in \Omega \end{array} \tag{34}$$

Where f: $\mathbb{R}^n \rightarrow \mathbb{R}$ is a real valued function that is to be minimised, and is called *objective function* or a *cost function*. The vector X is an n-dimensional vector of independent variables, that is, $X = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$. The variables $x_1, \ldots, x_n$ are often referred as *decision variables*. The set Ω is a subset of $\mathbb{R}^n$, called *feasible set*.
The constraint set takes the form $\Omega = \{x : h(X) = 0, g(X) \leq 0\}$, where h and g are given functions.

7.2 Classification of Techniques

The methods for function optimisation can be divided into three types:

1. Methods using only the function values
   These methods, also known as direct search methods, use only the function values for the optimisation. An example of this category of methods is the Simplex method, which was first introduced in 1947 by Dantzig. Well-known variant is Nelder-Mead method: the method is described for minimisation of a function of n variables, which depends on the comparison of function values at the $(n + 1)$ vertices of a general simplex, followed by the replacement of the vertex with the highest value by another point [22]. Evolutionary algorithms, including Genetic algorithms [15], and Simulated Annealing [23] fall into this category.

2. Methods using first-order derivative
   These methods, also known as gradient methods, are based on the principle of finding the roots of the first derivative of the objective function. Well known are the Steepest descent- and Newton-Raphson method [24]. Conjugate gradient methods, like Fletcher-Reevs [25], are used for finding the minima in unconstrained optimisation problems.

3. Methods that require knowledge of second-derivative
   These methods, also known as Hessian methods, calculate (or approximate) the Hessian matrix (second-order derivative determinant) to perform the extremum test whether the function has a relative maximum or minimum at the stationary point for which the Hessian is calculated [25].

7.2.1 Algorithm Selection Criteria

To select the appropriate method for this problem the following criteria were used:

1. No explicit knowledge of the optimisation functions needed.
   This is crucial, since the functions describing the building block parameter dependencies on the adjustment inputs are not know. Instead, discrete measurement values are available.
2. Multidimensional search.
   There will be, at the moment twelve, parameters that are simultaneously optimised so the method should be time-efficient to support a multidimensional search.

Furthermore, the algorithm should be easy to implement.

Given the first requirement and the discrete measurement data, a direct search method will be most appropriate choice. Two direct search methods will be compared: Genetic Algorithms and Differential Evolution. Both methods use parallel search instead of single point search, which is why they are preferred above the other direct search methods.

8
Genetic Algorithms

Genetic Algorithms (GAs) are search algorithms based on the mechanics of natural selection and natural genetics, which were first introduced in 1975 by [15]. The applications of GAs are numerous in fields of science, engineering and economy. GAs can be used to search for (global) optima in constrained or unconstrained optimisation problems.

The next simple example, from [16], has been modified in some extent to demonstrate the idea and working principle of GAs.

Consider the constrained optimisation problem of maximising the function $f(x) = x^2$ on the integer interval [0, 31]. A traditional way to solve this problem would be to 'turn the x knob' until the objective function is maximised (if $f(x)$ was a cost function it would need to be minimised). With GAs, the first step is to code the parameter x as a finite-length string. There are many ways of coding the parameter x, however the most obvious way would be to use the 5-bit binary coding.

The next step is to generate an initial population. Let's assume there is an initial population of 4 strings:

0 1 1 0 0
1 1 0 0 0
0 1 0 0 0
1 0 0 1 1

This population could have been chosen at random by tossing an impartial coin for 20 times (4 x 5 bits).

In order to produce the next generation GAs use three main operators:

1. Reproduction
2. Crossover
3. Mutation

Reproduction is a process in which individual strings are copied according to their objective function values (biologists call this function the fitness function). One can think of the function f as some measure of profit, utility, or goodness that is to be maximised. Copying the strings according to their fitness values means that strings with a higher value have a higher probability of contributing one of more offspring in the next generation. This operator is the artificial version of natural selection, a Darwinian survival of the fittest.

In Table 4 the current population is evaluated. Summing the fitness over all four strings 1170 is obtained. The percentage of population total fitness is also shown in the table.

Table 4: Evaluation of a population

No.	String	Decimal string value	Fitness $f(x)$	% of total
1	01101	13	169	14.4
2	11000	24	576	49.2
3	01000	8	64	5.5
4	10011	19	361	30.9
Total			1170	100

An easy way to implement the reproduction operator in an algorithm would be to create a weighted roulette wheel as shown in Figure 22. For example, the string number 1 has a fitness value of 169, which represents 14.4 percent of the total fitness, so the reproduction probability is weighted with factor 0.144 each time the wheel is spun. To reproduce, the wheel is spun four times.

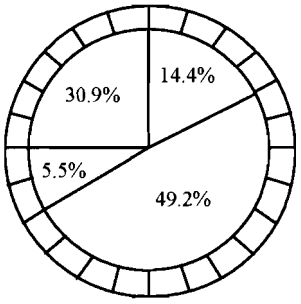


Figure 22: Roulette wheel reproduction operator implementation

After reproduction, simple crossover may proceed in two steps. First, members of the newly reproduced strings are randomly grouped in pairs. Secondly, each pair undergoes crossing over as follows: an integer position k along the string is selected uniformly at random between 1 and the string length minus one $[1, l-1]$. Two new strings are created by swapping all characters between positions $k + 1$ and l inclusively. For example, consider the first two strings from the initial population with $k = 4$

$A_1 = 0\ 1\ 1\ 0 \mid 1$
$A_2 = 1\ 1\ 0\ 0 \mid 0$

After crossover the new generation is represented by

$A'_1 = 0\ 1\ 1\ 0\ 0$
$A'_2 = 1\ 1\ 0\ 0\ 1$

The mechanics of reproduction and crossover are surprisingly simple, involving random number generation, string copies and some partial string exchange. Nevertheless, the information exchange is one of the features that give the GAs their power.
For example, each bit position in the string could represent some key feature. If the most significant bit is used to code the feature 'high gain' and least significant bit represents the feature 'low noise figure' than just two offspring where created where A'_2 has both the wanted qualities whereas A'_1 would probably fail to reproduce and die out because of the relatively low fitness. Despite the fact of somewhat fuzzy definition of the properties, such as

'high gain' and 'low noise figure', the exchange and combination of the features (or the ideas) is the basics of the innovation process where improvements are made regarding to the existing state.

The third operator, mutation, does occur with low probability and is performed on bit-by-bit basis. The effect of mutation is that some bits change from 0 to 1 (or vice versa) every now and than. Analogous to the natural processes, the mutation occurs very less frequently as reproduction and crossover.

Now all the operators are introduced, one step of genetic algorithm is simulated by hand to illustrate the working principle.

Each successive generation of the genetic algorithm begins with reproduction. Spinning the weighted roulette wheel generates the pool of mates, which are to be crossed over. Actual simulation of this process using the weighted coin tosses has resulted in string 1 and string 4 receiving one copy, string 2 receiving two copies and string 3 receiving no copies, as shown in Table 5. Comparing with the expected number of copies there is obtained exactly what is expected: the best get more copies, the average stay even and the worst die off.

Table 5: The first population simulation

	Initial population	***x* value**	**Fitness**	$P_i\{select\}$	**Expected count**	**Actual count**
String No.	**(randomly generated)**	**(unsigned integer)**	$f(x)=x^2$	$\frac{f_i}{\Sigma f}$	$\frac{f_i}{\overline{f}}$	**(from roulette wheel)**
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

With an active pool of strings looking for mates, simple crossover proceeds in two steps: (1) strings are mated randomly using coin tosses to pair the couples and (2) mated string couples cross over using the coin tosses to select the crossing sites. This is shown in Table 6.

The probability of mutation is assumed to be 0.001 for this example. With 20 transferred bit positions we should expect 20 · 0.001 = 0.02 bits to undergo mutation during a given generation. As a result, no bits have mutated in this generation.

It's clear that both in natural and GAs the mutation is a secondary effect whereas reproduction and crossover form a bulk of the processing power. Nevertheless, the mutation plays an important role for insuring that no valuable features ('good genes') are prematurely lost.

Table 6: The GA after one generation

	Mating pool after reproduction	Mate	Crossover site	After crossover		Fitness
String No.	**(crossover site shown)**	**(randomly selected)**	**(randomly selected)**	**new population**	***x* value**	$f(x)=x^2$
1	0 1 1 0 | 1	2	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	1	4	1 1 0 0 1	25	625
3	1 1 | 0 0 0	4	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	3	2	1 0 0 0 0	16	256
Sum						1754
Average						439
Max						729

The new population is now evaluated; notice how both the maximal and average performance have improved. Although only one step of the algorithm is evaluated and it is dangerous to conclude from this that the GAs are robust and always work, it clearly illustrates the algorithm mechanics and potentials.

Considering the fittest strings in the latest population, it can be concluded that strings beginning with two ones (1 1 * * *) perform the best. A *schema* [15] is a similarity template describing a subset of strings with similarities at certain string positions. These highly fit, short length schemas that are propagated generation to generation are called 'building blocks'. This similarity information contained in building blocks can be used for searching the solution space more efficiently.

Because the GAs are inspired by natural processes the terminology used is very often mixed with the terminology that is used in natural systems. Table 7 gives the comparison of natural and GAs terminology.

In natural systems, one or more *chromosomes* combine to form the total genetic prescription for the construction and operation of some organism. The total genetic package is called *genotype*. In artificial system the total package of strings is called *structure* (in this example the structure consists of only one string). In natural systems, the organism formed by the interaction of the total genetic package with its environment is called the *phenotype*. In artificial genetic systems, the structures decode to form a particular *parameter set, solution alternative* or *point* (in the solution space). In natural terminology, the chromosomes are composed of genes, which may take on some number of values called *alleles*. The position of the gene, its *locus*, is identified separately from the gene's function. For example, if we take the animal eye colour gene, we can talk of its locus, position 10, and its allele value, blue eyes. In artificial genetic search, the strings are composed of *features* or *detectors*, which can take on different values and may be located at different positions on the string.

Table 7: Comparison of natural and GAs terminology

Natural	**Genetic Algorithms**
chromosome	string
gene	feature, character or detector
allele	feature value
locus	string position
genotype	structure
phenotype	parameter set, alternative solution, a decoded structure
epistasis	non-linearity

The GAs differ from traditional methods in several aspects [21]

- They work with an encoding of the feasible set rather then the set itself.
- They search from a set of points rather than a single point at each iteration.
- They do not use derivatives of the objective function.
- They use operations that are random within each iteration.

9
Differential Evolution

The *Differential Evolution* (DE) algorithm is a heuristic search method that can be categorised into a class of floating-point encoded, evolutionary optimisation algorithms. It was introduced in 1995 by [6] and was originally designed to work with continuous variables.

Generally, a function to be optimised, f, is of the form:

$$f(X):\mathbb{R}^n \rightarrow \mathbb{R} \tag{35}$$

The optimisation target is to minimise the value of this objective function

$$\min(f(X)) \tag{36}$$

by optimising the values of its parameters

$$X = (x_1,...,x_{n_{param}})^T \qquad x \in \mathbb{R} \tag{37}$$

where X denotes a vector composed of n_{param} objective function parameters. Usually, the parameters of the objective function are subject to lower and upper boundary constraints, $x^{(L)}$ and $x^{(U)}$, respectively:

$$x_j^{(L)} \le x_j \le x_j^{(U)} \qquad j = 1,...,n_{param} \tag{38}$$

As with all evolutionary algorithms, DE works with a population of solutions, rather than a single solution for the optimisation problem. Population P of generation G contains n_{pop} solution vectors called individuals of the population. Each vector represents a potential solution for the optimisation problem.

$$P^{(G)} = X_i^{(G)} \qquad i = 1,...,n_{pop}\ , G = 1,...,G_{\max} \tag{39}$$

So, the population P of generation G contains n_{pop} individuals each containing n_{param} parameters:

$$P^{(G)} = X_i^{(G)} = x_{i,j}^{(G)} \qquad i = 1,...,n_{pop}\ , j = 1,...,n_{param} \tag{40}$$

In order to establish a starting point for optimisation, the population must be initialised. Often there is no more knowledge available about the location of a global optimum than the boundaries of the problem variables. In this case, a natural way to initialise the population $P^{(0)}$ (initial population) is to seed it with random values within the given boundary constrains

$$P^{(0)} = x_{i,j}^{(0)} = r_{i,j}(x_j^{(U)} - x_j^{(L)}) + x_j^{(L)} \qquad i = 1,...,n_{pop}\ , j = 1,...,n_{param} \tag{41}$$

where r denotes a uniformly distributed random value within range [0,1].

The population reproduction scheme of DE is different from the other evolutionary algorithms. From the first generation forward, the population of the following generation $P^{(G+1)}$ is created in the following way on basis of the current population $P^{(G)}$. First, a temporary (trial) population for the subsequent generation, $P_i'^{(G+1)}$, is generated as follows:

$$x_{i,j}^{'(G+1)} = \begin{cases} x_{C_i,j}^{(G)} + F \cdot \left(x_{A_i,j}^{(G)} - x_{B_i,j}^{(G)} \right) & \text{if} \quad r_{i,j} \le C_r \vee j = D_i \\ x_{i,j}^{(G)} & \text{otherwise} \end{cases} \tag{42}$$

where

$i = 1,\ldots, n_{pop}$, $j = 1,\ldots, n_{param}$

$D = 1,\ldots, n_{param}$

$A = 1,\ldots, n_{pop}$, $B = 1,\ldots, n_{pop}$, $C = 1,\ldots, n_{pop}$, $A_i \neq B_i \neq C_i \neq i$

$C_r \in [0,1]$, $F \in [0,1]$, $r \in [0,1]$

A, B and C are randomly chosen indexes referring to three randomly chosen individuals of the population. They are mutually different and also different from the running index i (for each individual). A new value for random number r is assigned for each value of index j.
F and C_r are DE control parameters, together with the population size parameter, n_{pop}. All three parameters stay constant during the search process. F is a real-valued factor that controls the amplification of differential variations and C_r is a real-valued crossover factor controlling the probability to choose the mutated value for x instead of the current value. Both F and C_r affect the convergence velocity and robustness of the search process. Their optimal value depends on objective function, $f(X)$, characteristics and on the population size n_{pop}. The selection scheme of DE is based on so called *greedy criterion*. On basis of the current population $P^{(G)}$ and the temporary population $P_i^{'(G)}$, the population of the next generation $P^{(G+1)}$ is created as follows:

$$X_i^{(G+1)} = \begin{cases} X_i^{'(G+1)} & \text{if} \quad f_{cost}(X_i^{'(G+1)}) < f_{cost}(X_i^{(G)}) \\ X_i^{(G)} & \text{otherwise} \end{cases} \tag{43}$$

Thus each individual of the temporary population is compared with its counterpart in the current population. The one with the lower value of cost function $f_{cost}(X)$ will survive to the population of the next generation. As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation.

9.1 Discrete Differential Evolution

As previously mentioned, the DE algorithm was originally designed to work with continuous variables. In this project, the objective function parameters (the building block adjustment parameters) will have, most probably, 10 discrete steps per each parameter. These vectors, representing an operating point, are pointing to a building block parameter vector

$$X_i = (x_1, x_2)^T \rightarrow (p_1,\ldots, p_{n_{param}})^T \qquad i = 1,\ldots, n_{pop} \tag{44}$$

where n_{param} is the number of measured parameters per operating point.
The values of p_1 to $p_{n_{param}}$ will be obtained by measurements, which will always have some limited accuracy. So it can be argued that, having discrete value steps with limited accuracy, there is no need for continuous optimisation with a high precision.

Instead of working with real values of the adjustment parameters, the index (integer) values can be used. For instance if x_1 varies in 10 equidistant steps in the interval [0.2,2], the index value could be used for the optimisation rather than the value itself:

i	1	2	3	4	5	6	7	8	9	10
$x_1(i)$	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0

The conversion from the index to the real value is trivially done by the multiplication of the index value by the stepsize:

$$x_1(i) = i \cdot stepsize \tag{45}$$

When adapting the DE algorithm to work with integers, the first population is initialised with the integer values (on the grid), but the subsequent trial vectors can assume real values. This is caused by the real-valued amplification factor F, which can produce the vectors that point in-between discrete measurement steps where no building block parameter values are available, as shown in Figure 23.

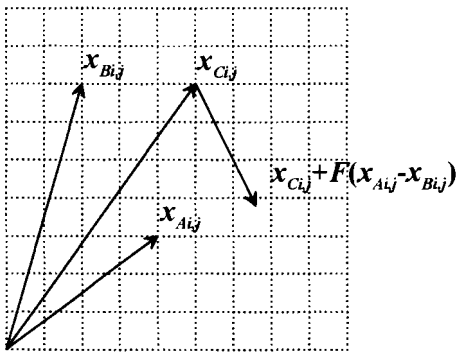


Figure 23: Real-valued vector coordinates

There are several possibilities for solving this problem:

1. The objective function can be made continuous by curve fitting.
2. The objective function can be made continuous by interpolating the in-between values.
3. The DE algorithm could be made discrete.

The first two options are making the discrete measurement data appear continuous while keeping the DE algorithm unchanged. The third option keeps the data discrete while the algorithm needs to be modified to work with these data. Since the integer values are used rather than their real-valued counterparts, the discretisation is easily realised by rounding-off the values to the nearest integer

$$x_{i,j}^{(G)} = round(x_{i,j}^{(G)}) \qquad i = 1,\ldots,n_{pop}\ ,\ j = 1,\ldots,n_{param} \tag{46}$$

where function *round* rounds a real value, either up or down, to the nearest integer.
However, doing so will limit the precision to the discrete steps of the adjustment parameters whereas the optimisation method should be able to produce the optimum solution that lies between the adjacent measurement steps.

To enhance the precision, the data can be interpolated prior to be fed to the optimiser. This provides a significant time saving in comparison with the second option, where the CPU-time consuming interpolation is performed within each iteration and for each trial vector.

9.1.1 Boundary Constraints Handling

It is important to notice that the reproduction operator is able to produce the vectors that fall outside the valid range of the adjustment parameters. Figure 24 shows an example of how such a vector can be constructed.

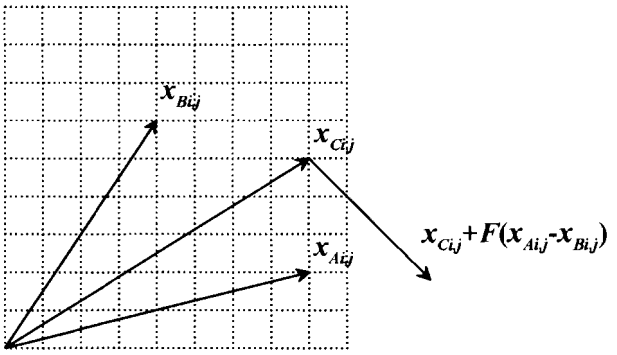


Figure 24: Out-of-range vector generation

A simple solution for this problem has been found: disregard these vectors. This is also the most time effective solution since it is relatively cheep, in terms of consumed time, to generate trial vectors. Another solutions, such as rounding-off the out-of-range coordinates to the borderline would interfere with the normal course of the algorithm. In that case, the values at the border would have a higher probability of being appointed which would affect the random character of the algorithm.

Another thinkable solution is to flip the invalid coordinates around the crossed boundary so the vector does not have to be thrown away. However, it would cost more time to correct for this than to ignore the respective vector and generate a new one in the next iteration.

If more than, e.g. 3/4 of trial vectors, fall outside the valid range the miss-convergence would become highly probable. A safeguard has been introduced to prevent this. It has been implemented in the form of a simple counter that keeps up the number of the in-range vectors. If the percentage of valid vectors is lower than, e.g. 20%, the value of F should be decreased.

10
Implementation

Several tools were explored for the possibilities of the implementation. Regarding to the nature of the problem and the available data, the choice is made to embed the optimisation method into MATLAB environment. MATLAB was selected because it offers a high-performance numerical computation and visualisation. It uses high-level programming language and it offers many functions that are already implemented in different toolboxes.

Two algorithms were selected for the implementation and brief comparison: Genetic Algorithms and Differential Evolution.

10.1 Implementation of the Genetic Algorithm

The Genetic Algorithm has been implemented in a MATLAB code according to the flow chart in Appendix B. To make the start more fluent, the simple function $f(x)=x^2$ on the integer interval [0, 31] has been used as the objective function to be maximised. This function was used in Chapter 8 to explain the working principle of GAs.

The parameter x is encoded with 5-bit binary coding. The algorithm has been implemented in a *graphical user interface* (GUI). This makes it easy to monitor what is going on and to debug the code.

As a result of the reproduction operator, which is the equivalent of the Darwinian model of natural selection, the best (highly fit) members from the population receive the most copies. In this way a mating pool is created were the mates are randomly paired and the crossover site is randomly chosen. If the mates in a couple are identical, which is highly probable since the fittest members get the most copies, the crossover operator has no effect on them. If all the members of a given population are identical, the non-effectiveness of the crossover operator causes the algorithm to halt at the value represented by these members. This situation is shown in Figure 25. The stars mark the maximum value of the population fitness whereas the continuous line shows the average fitness. All the members of the population have reached the same value, decimal 19, and from 10^{th} iteration the population does not evolve any further.

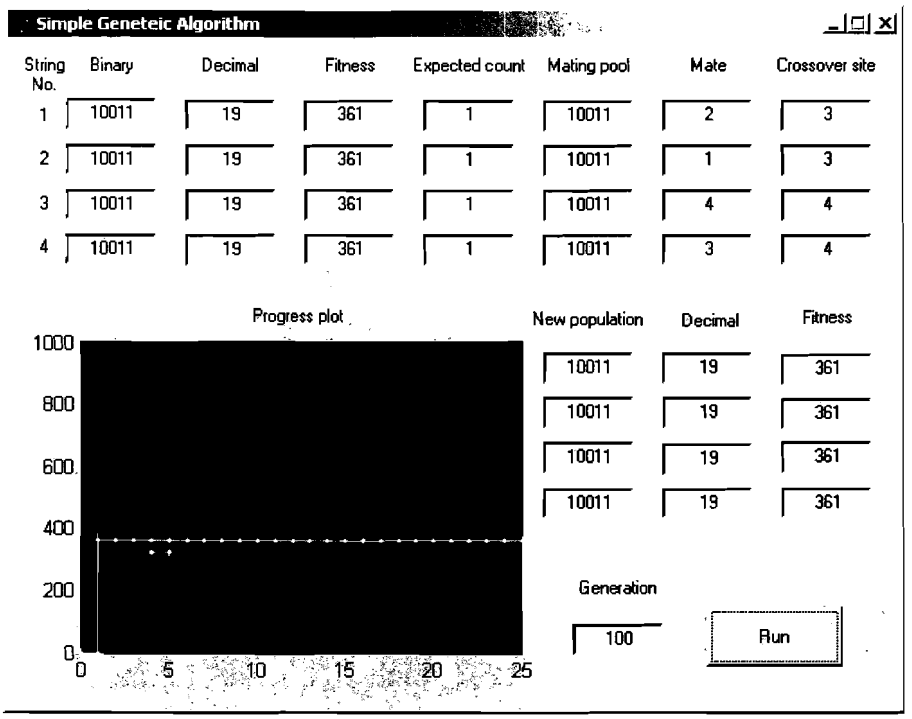


Figure 25: Simple Genetic Algorithm screenshot

A way to sample a state away of the average of the population and to escape from this miss-convergence is provided by the mutation operator. The mutation operator, occasionally flipping random bits in the population, causes individuals to sample the search space with the integer precision. The new states are typically sub-optimal relative to the population average; such a mutation has occurred in Figure 26 at 4th generation. These random changes can, though infrequently, increase the average fitness of the population, as shown in Figure 26 at 15th generation. From that point the population has evolved towards the global solution by the normal reproduction and crossover processes.

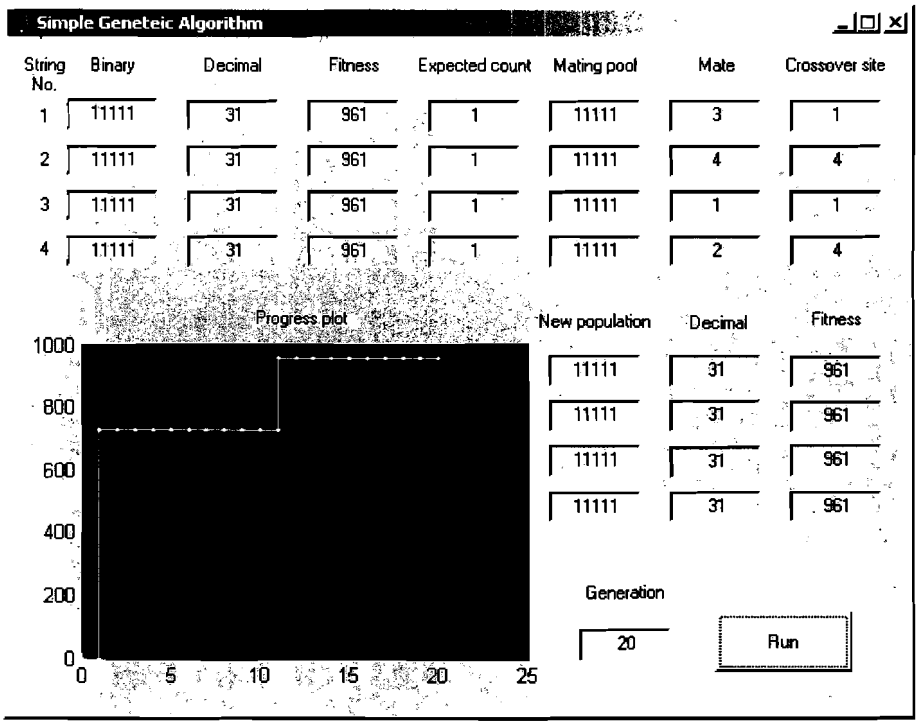


Figure 26: GA with mutation probability 0.01

The convergence is dependent on the algorithm control parameters: population size and mutation probability, P_m. Increasing the population size increases the chance and speed of finding the global optimum.

The mutation probability should be chosen carefully and remain low (e.g. 0.01) because the high mutation rate introduces 'noise' into convergence process, which has the effect that no steady state can be reached. This is shown in Figure 27, where the number of iterations is set to 100 and P_m is set to 0.1. As a result, the population average shows large deviations and does not converge.
For the case that P_m = 0.5, the GA becomes the simplest form of the evolutionary algorithms: Random Walk.

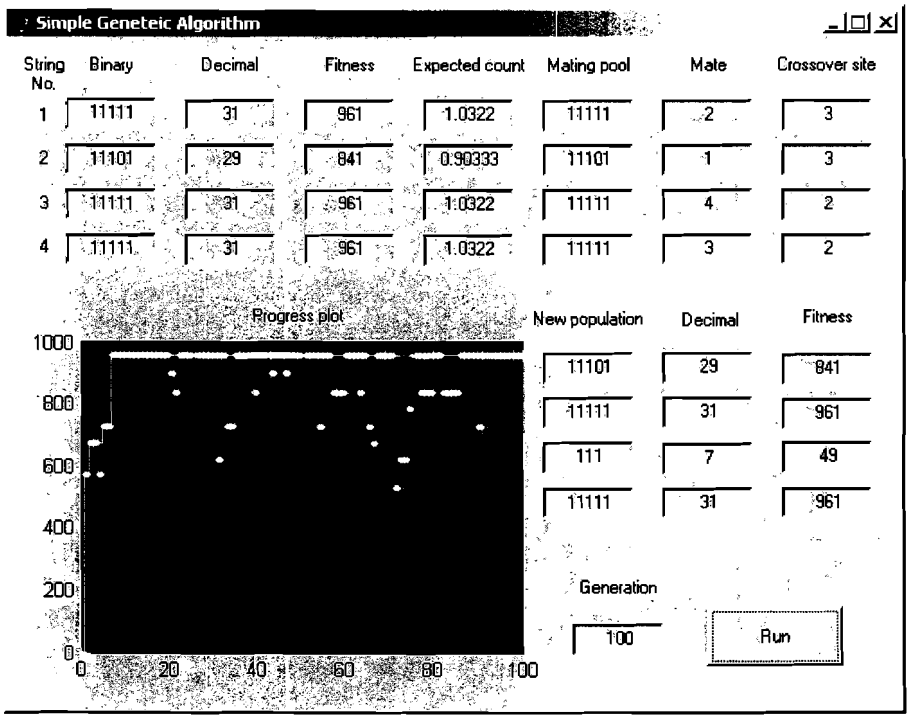


Figure 27: Screenshot of non-converging GA caused by high mutation rate

10.1.1 Reinsertion and Elitist Strategy

The mutation operator follows the crossover operator. It is possible that the fittest member ever, that has just been created by the crossover operator, is lost due to the mutation operator. This could happen if, e.g. the most significant bit were inverted.
To prevent this from happening a reinsertion can be performed. The best member ever is stored after the crossover operator. After the mutation, the population is screened if it still contains the best member. If not, it can be reinserted, so the 'good genes' are not lost. In this investigation, the best member is reinserted on the position of the member with the lowest fitness. As a result, on average, the optimum value was quicker reached.

Another variant, found in literature [26], copies the best member a few times into each new population. This ensures its reproduction and propagation into subsequent generations. If one or more of the fittest individuals are deterministically allowed to propagate through successive generations than the GA is said to use the *elitist strategy*.

10.1.2 Termination of the GA

Because the GA is a stochastic search method, it is difficult to formally specify convergence criteria. As the fitness of a population may remain static for a number of generations before a superior individual is found, the application of conventional termination criteria becomes problematic. A common practice is to terminate the GA after a pre-specified number of generations and the best member should contain the optimum solution. This, however, provides no certainty that the global optimum has been found.

10.2 Implementation of the Differential Evolution Algorithm

The Differential Evolution has also been implemented in a MATLAB code, whose basic code has been made available through [27]. The GA and the DE differ in the way they generate subsequent generations.
The GAs use the reproduction, crossover and the mutation operators to create the new vectors. With DE, the weighted difference vector serves as the source of random variations for a third vector

$$x_{new} = x_3 + F \cdot (x_1 - x_2) \tag{47}$$

which has previously been formalised in equation (42).

Figure 28 depicts the process of differential evolution [27]. First, a target vector is chosen. Second, two random vectors from the current population are selected. Subsequently, these vectors are subtracted, amplified by the factor F, and are added to a third vector, as prescribed by (47). The following operation is the crossover, where randomly chosen coordinates of the newly created vector are exchanged with the coordinates of the target vector, with the crossover probability C_r.

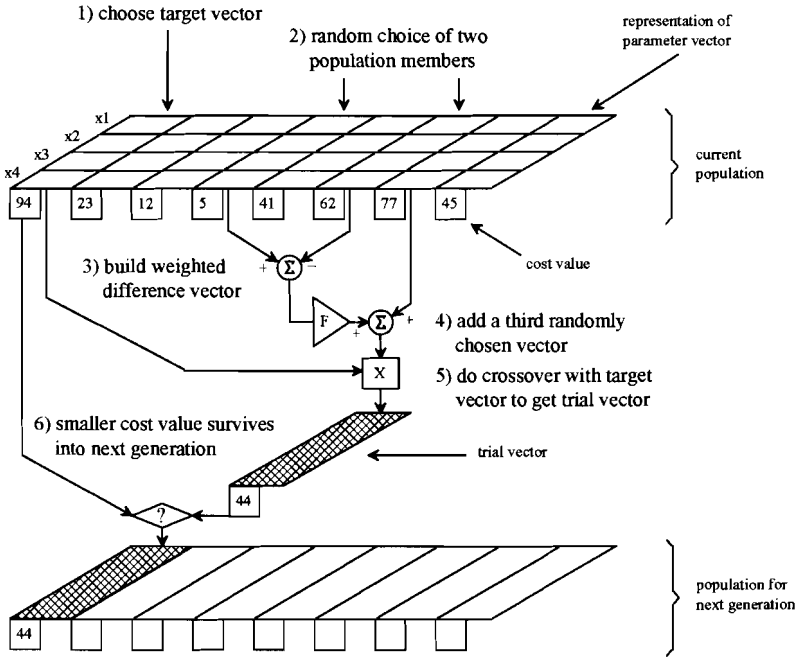


Figure 28: Generation of subsequent populations in DE algorithm

The resulting trial vector is then compared with the target vector. If the cost value of the trial vector is lower than the cost value of the target vector, the target is replaced by the trial in de next generation. This selection is based on so-called *greedy* criterion; the vectors with low cost value replace the vectors with a high cost value.

There are different strategies possible depending on the random vectors generation. If x_3 in equation (47) is the best member then the algorithm uses so called 'Best member' strategy. The best member, i.e. vector with the lowest cost value so far, is copied n_{pop} times and each difference vector is added to the best member.
If the third vector, x_3, is randomly chosen out of the population members then the algorithm uses the 'Random' strategy.

After DE successfully minimised $f(x) = -x^2$, function 'Peaks', which is standard available in MATLAB, was used as a test function. Function Peaks is defined as

$$z(x_1, x_2) = 3 \cdot (1 - x_1)^2 \cdot e^{-x_1^2 - (x_2 + 1)^2} - 10 \cdot \left(\frac{x_1}{5} - x_1^3 - x_2^5 \right) \cdot e^{-x_1^2 - x_2^2} - \frac{1}{3} e^{-(x+1)^2 - x_2^2} \qquad (48)$$

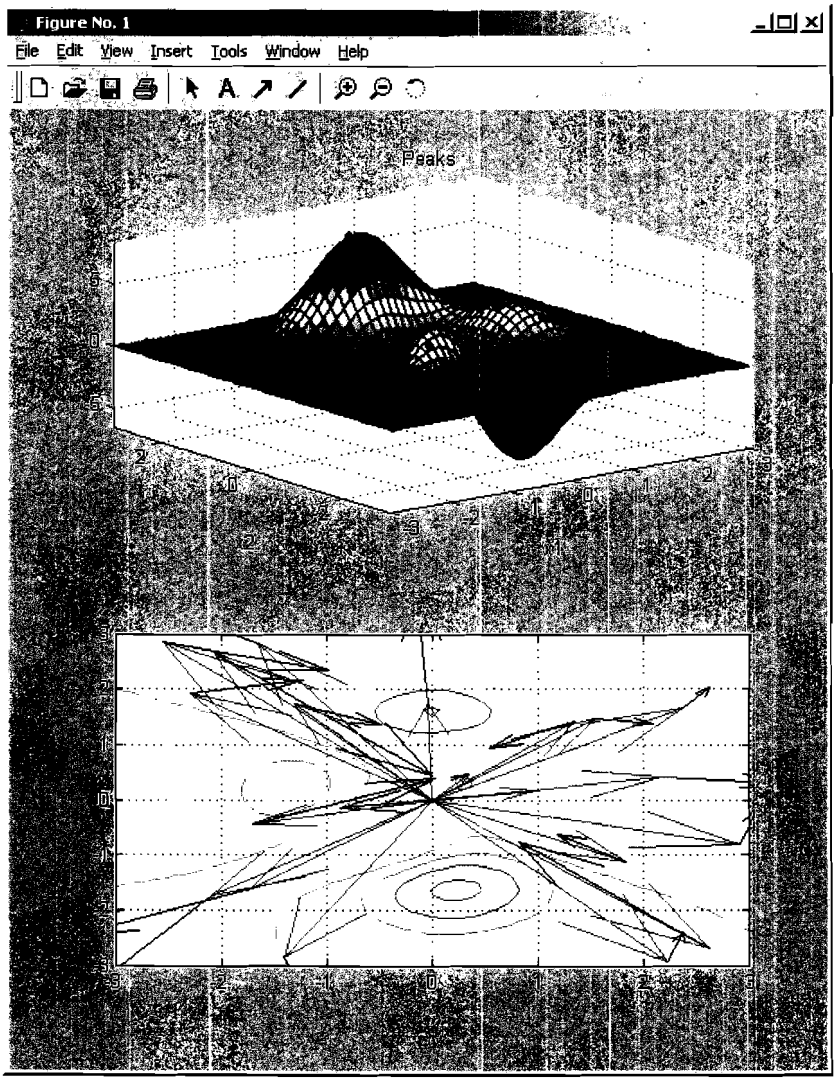


Figure 29: Screenshot of first population in Differential Evolution

Although the search space is relatively small and the function is dependent on a 2-dimensional parameter vector, a lot can be learned about the dependency of the convergence process on the algorithm control parameters.

The upper subplot in Figure 29 shows the 'cost value' of function peaks in 3-dimensional space. The blue dots on the surface represent the initial population of 20 randomly chosen members. The magenta dot shows the position of the initial 'best member', i.e. the vector with the lowest cost value. This vector lies in the local minimum valley and is a nice test case of the algorithm's ability to escape local minima.

The lower subplot shows the same function in a two-dimensional space where the cost value is represented by the contour plot. The red vectors show the points that are sampled by the initial population, pointing to the positions of blue dots in the upper subplot of Figure 29. The black vectors represent the weighted difference vector from equation (47), pointing to the new test points.

The algorithm control parameters in this example run are: $n_{pop} = 20$, $F = 0.4$, $C_r = 0.6$ (number of population members, weight factor, crossover probability).

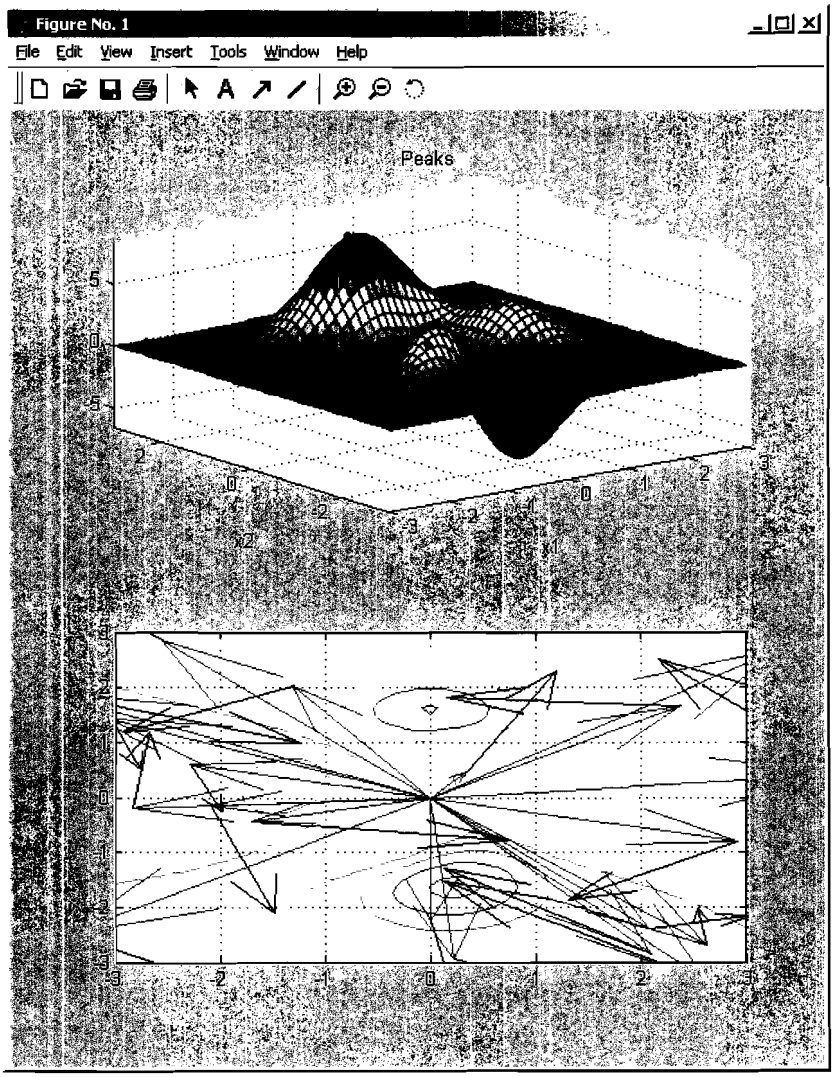


Figure 30: Screenshot of a DE run after 3 iterations

When a new 'best member' is found during the optimisation, its position is plotted as a red dot in the upper subplot. The blue dots of the initial population stay unchanged.
The lower subplot is updated each iteration with the current population vector field (red vectors) and new weighted difference vectors (black vectors).

Figure 30 shows the state after 3 iterations. The current best member is already situated near the global minimum. The figure also demonstrates a successful escape from the local minimum.

Figure 31 shows the situation after 20 iterations. There are two cheap areas in the function Peaks; those are the areas of the local and the global minima, the blue areas in the contour plot. What can clearly be seen is that all 'expensive' vectors from Figure 30 are replaced by the 'cheaper' ones, so the complete population evolves towards the cheap areas.

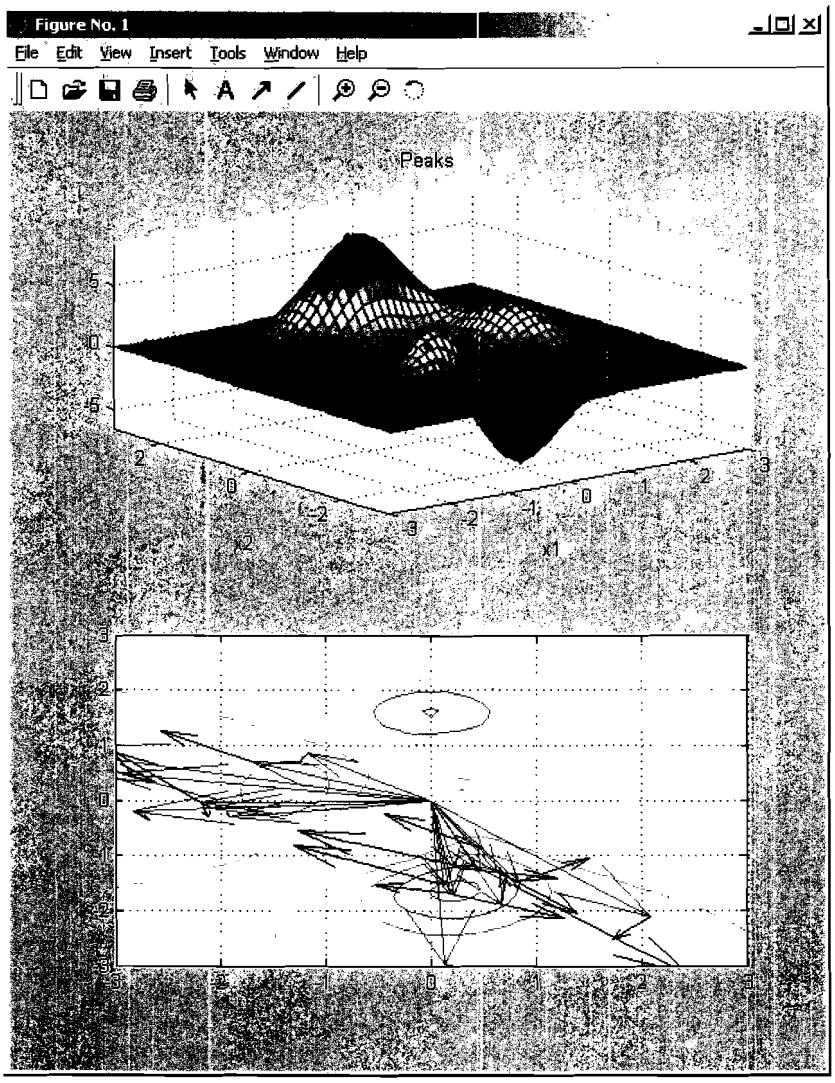


Figure 31: The population evolves towards the cheap areas

After 10 more iterations, the total population has evolved in the area of the global minimum, as shown by Figure 32. So all the population members have concentrated in the close proximity of each other within the area of the global minimum. One of the consequences of this is that the magnitude of the difference vector gets increasingly smaller, which allows for even finer search in the increasingly smaller area.

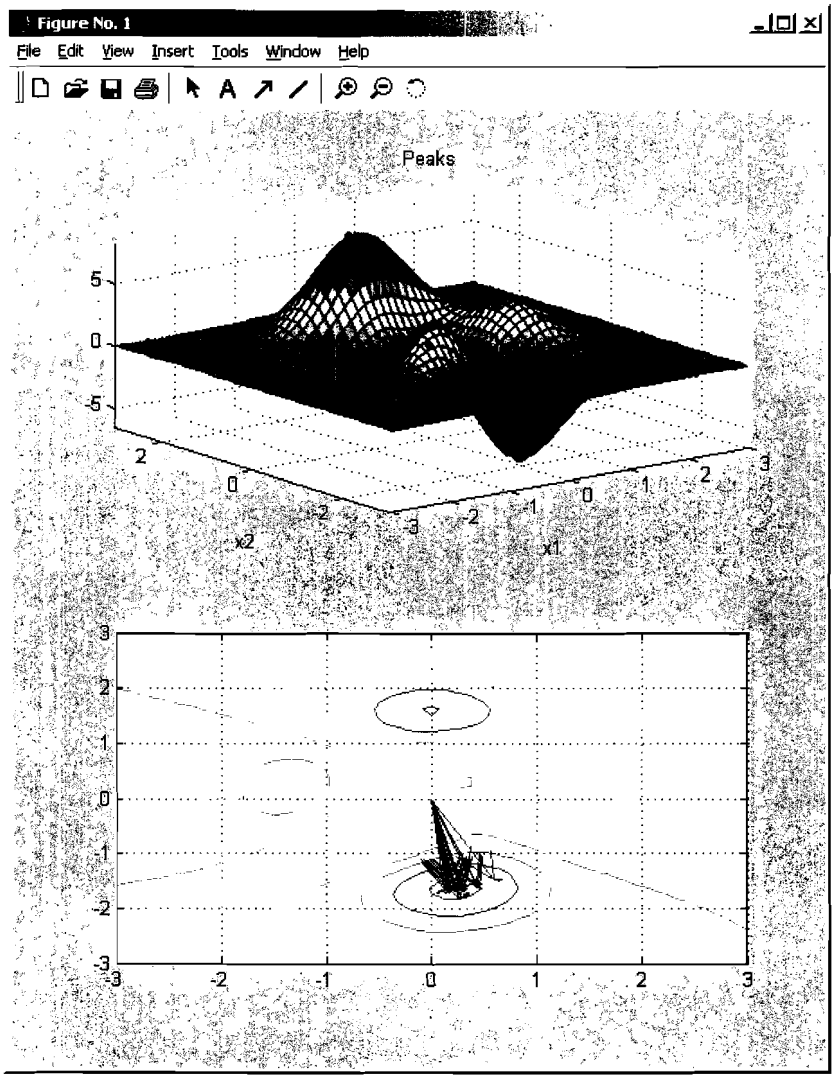


Figure 32: Whole population concentrates in the global minimum area at 20th iteration

After in total 50 iterations the whole population converges to a single point, the global minimum, as shown in Figure 33.

Except the graphical output, the numerical state is displayed each 10 iterations in the MATLAB control window. The output of this example run is shown below. The cost value of the best member is displayed in the field 'Best' and the vector coordinates are given by `x(1)` and `x(2)`.

```
Iteration: 10,  Best: -6.113038,  F: 0.400000,  CR: 0.600000,  NP: 20
x(1) = 0.447771
x(2) = -1.614524
Iteration: 20,  Best: -6.467597,  F: 0.400000,  CR: 0.600000,  NP: 20
x(1) = 0.287428
x(2) = -1.678929
Iteration: 30,  Best: -6.550152,  F: 0.400000,  CR: 0.600000,  NP: 20
x(1) = 0.219107
x(2) = -1.623089
Iteration: 40,  Best: -6.551053,  F: 0.400000,  CR: 0.600000,  NP: 20
x(1) = 0.227819
x(2) = -1.623274
```

```
Iteration: 50,  Best: -6.551133,  F: 0.400000,  CR: 0.600000,  NP: 20
x(1) = 0.228274
x(2) = -1.625775
Number of function evaluations:  1000
```

The function value at the global minimum is –6.551133. Within 50 iterations, i.e. 1000 function evaluations, the algorithm has found this point with accuracy of 10^{-6}.
If the accuracy of 10^{-1} were required, the algorithm would have stopped between 20 and 30 iterations.

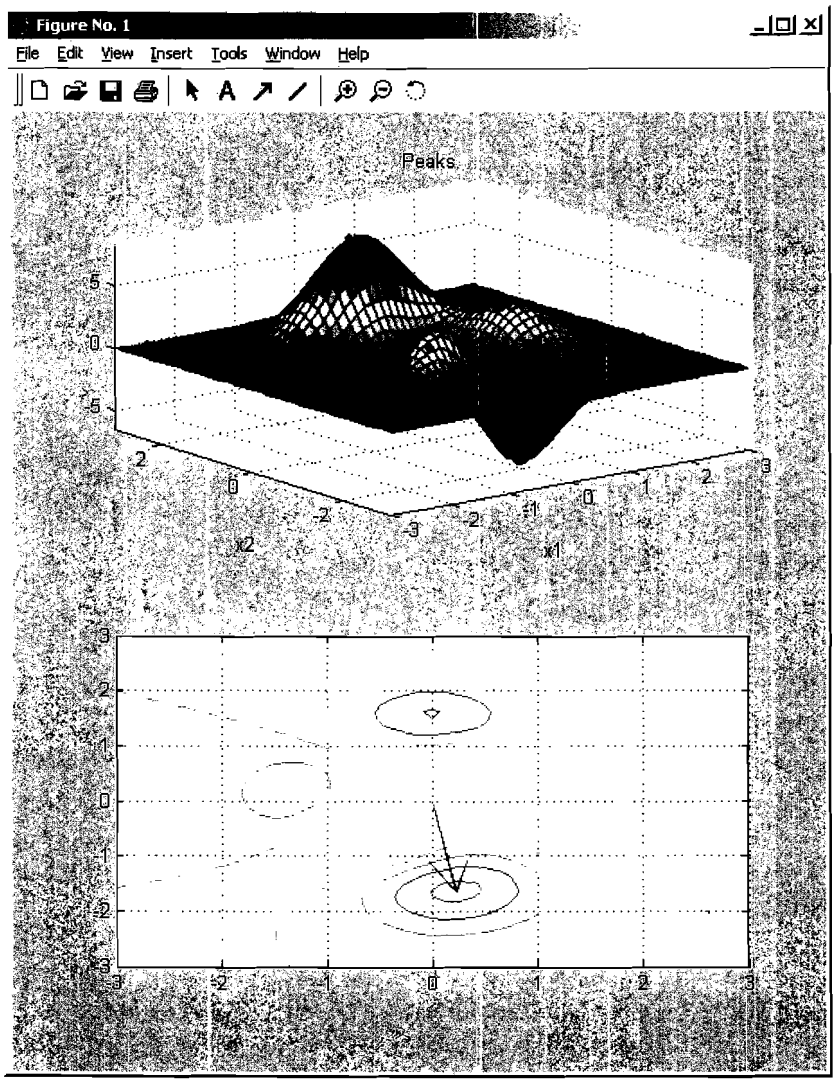


Figure 33: The whole population converges in the global minimum

Since stochastic processes rule DE, there is a difference between single runs. On the average, it takes around 800 function evaluations (i.e. 40 iterations) to find the minimum of function peaks with 10^{-6} precision, for this example.

10.2.1 Influence of the DE Control and Strategy Parameters

The DE has three control parameters that influence the speed and robustness of the convergence process: population size n_{pop}, weight factor F and the crossover probability C_r. Some rules of thumb for the choice of the control parameters, based on experience gathered during different tests performed with DE, are given below. How the choice of the strategy influences the convergence is discussed afterwards.

The reasonable choice for the population size is between $n_{pop} = 3 \cdot D$ and $8 \cdot D$, where D is the dimension of the parameter vector. It is important to have enough mutually different vectors. The larger the population, the higher the probability of finding the global minimum. However, this is not always necessary and a large population implies large number of function evaluations and longer calculation times.

The amplification factor F should not be smaller than a certain value to prevent premature convergence. This value depends on the cost function and on the other DE control parameters. A large F increases the chance of escaping the local minima. However, for $F > 1$ the convergence speed decreases [28]. A good initial value for the amplification factor is $F = 0.6$.

A large crossover constant, C_r, speeds up the convergence. However, from a certain value upwards the convergence speed may decrease or the population may converge prematurely. This value depends on the cost function and is located in the region $C_r = 0.9...1.0$.
A good choice for the initial crossover probability is a value between $C_r = 0.3$ and $C_r = 0.9$.

Two strategies have been evaluated: the 'Best member' and the 'Random' strategy. The Random strategy is more robust and is preferred above the Best member because it is more likely to lead to the global solution. The Best member, however, converges quicker and may be used in combination with the initial guess, if the region of the global minimum is roughly known.
With the Best member strategy and high crossover probability, the whole population tends to move quickly to a (local) minimum. With low value of F, the chance of escaping this, possibly local, minimum gets increasingly smaller.

10.2.2 Termination of the DE

In order to prevent a possible non-converging run to circle in an endless loop, the DE algorithm can also be terminated after a pre-specified number of iterations.

In contrary to GAs, a trial vector could never get far away from the population average. The largest distance is the maximum distance between two outermost vectors. As the population converges, this distance gets smaller.

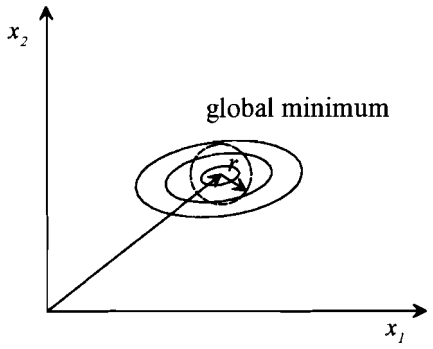


Figure 34: The termination radius

If the total population has converged within certain error area around the best member, given by radius *r* in Figure 34, the algorithm may be terminated. At that point, it is evident that there is no possibility to find any new, better points.

10.3 Discussion on GAs and DE

The experimentation with the GAs has been abandoned after it has shown that many tricks and work-arounds are needed to let the GA to run well. The basic idea, however, of the reproduction, crossover and mutation has been proven to work, as clearly is illustrated by Figure 26. If the mutation and crossover probabilities are well chosen, the population tends to evolve towards the highest fitness. Nevertheless, the non-ergodic way of sampling the search state [29] does not guarantee the global optimum.

GAs are concerned primitive, in comparison with the advanced search algorithms of today, but still may lend themselves well for tackling some non-differentiable or discrete problems. GAs are especially suited for use in self-learning systems and artificial intelligence.

The Differential Evolution algorithm showed better convergence properties than GAs, in this investigation, as well as in the literature [6]. Important advantage of DE compared to GAs is the termination criterion. The DE has been successfully used in some RF-engineering problems, as reported in [1], [30]. The DE is also available through an add-on for the renowned mathematical software package 'Mathematica'.

After the DE was made discrete, because a limited precision is needed and discrete data sets are available, the convergence velocity increased even further. Fast convergence, deterministic termination criterion and easy handling have lead to the discrete DE algorithm being selected as the technique to proceed with.

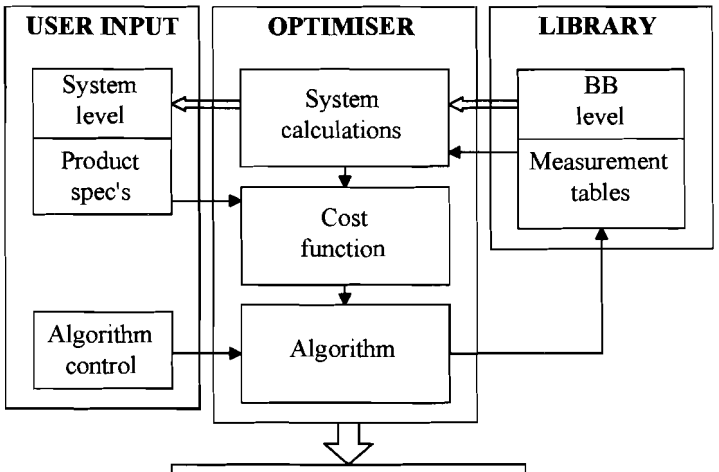
11
RF System Optimisation Tool

In previous chapters, the optimisation parameters, the system calculations, the cost function, the optimisation technique and the implementation tool have been defined. In this chapter, all these topics are combined into the optimisation tool. The performance of the method will be evaluated and handled through an example system.

11.1 Model of the Optimisation Tool

The optimiser with its environment is shown in Figure 35. There is user input at the one side and the library at the other. The coupling between the two is the optimiser. Its task is to find the optimum composition of the building blocks and the values of their adjustment inputs based on the product specifications provided by the user and the available building blocks in the library. The fourth block is the output block, where the results of the optimisation are presented.

Between the measurement tables, at the building block level, and the product specifications, at the system level, there are system calculations.



OUTPUT			
Building block		Parameter	
type	no.	x_1	x_2
LNA-mix	5	0.27	0.74
IF-baseband	2	1.15	1.82
etc.	etc.	etc.	etc.

Figure 35: Model of the optimisation tool

The method works as follows. The user fills in the product specifications that need to be optimised, sets the algorithm control parameters and starts the procedure. Then the algorithm generates the test vectors that are pointing to the specific operating points in the measurement tables. The building block parameter values at these locations are entered in the system equations where the system performance is calculated.

Subsequently, the cost function is built using the system performance values and the product specifications. The algorithm evaluates the cost function value and uses this information to determine the following test vectors, which closes the loop.
After the algorithm is terminated, the output is generated.

The implementation of the user input, the optimiser, the library and the output blocks form Figure 35 will be discussed in the following sections. Also the interfaces between these blocks will be discussed.

11.1.1 The User Input

The user input block from Figure 35 has been implemented in a graphical user interface. The algorithm is controlled through the *Algorithm control* frame and the product specifications are entered through the *Weighting factors* and the *Constraints* frames in Figure 36.

The *Algorithm control* frame contains the algorithm control parameter fields. These are, from top to bottom, number of population members, amplification factor, the crossover constant, convergence tolerance, maximum allowed number of iterations and the algorithm strategy.

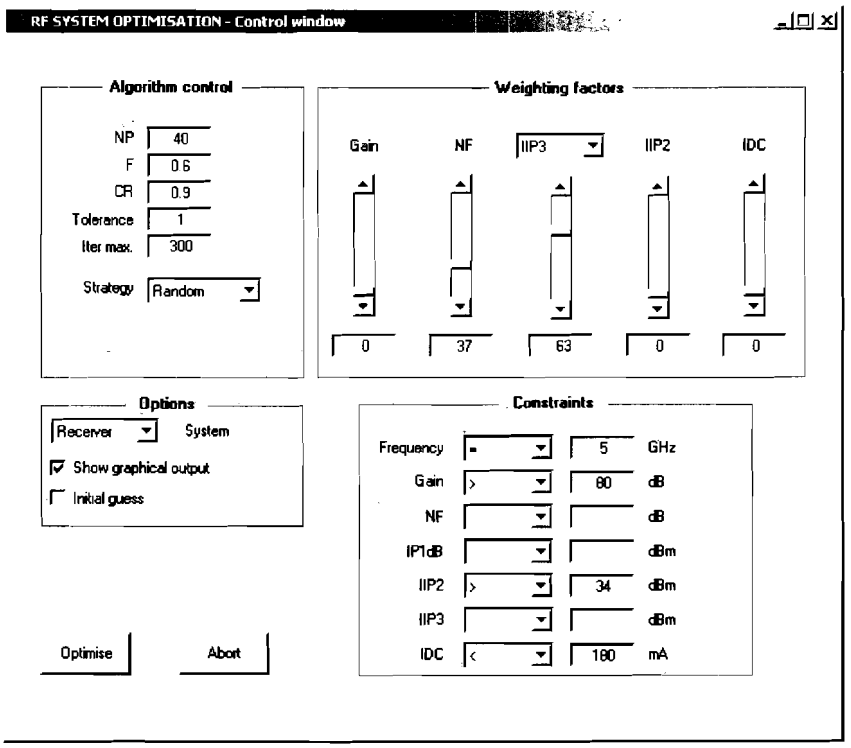


Figure 36: The RF System Optimisation control window

The parameters to be optimised are entered in the weighting factor fields. Using the slide bars, the relative importance of the parameters can be specified. Based on the position of the slide bars the weighting factors are automatically calculated to satisfy

$$\sum_{i=1}^{n} w_i = 1 \tag{49}$$

The parameters that are not optimised can be specified in the constraint fields, even though it is possible to optimise a certain parameter and at the same time to specify the constraint for it. A constraint can be specified as smaller than, <, or larger than, >. The frequency is specified as equal to, =, and is used to pre-select the building blocks that can operate in the specified frequency range.

Furthermore, three options can be chosen: *Receiver (or Transmitter) system, Show graphical output* and *Initial guess.* From the graphical output, the dependency of the optimality on the adjustment parameters and the locations of the possible local minima can be deduced. An example of graphical output is shown in Appendix D.
A push on the Optimise button reads the values entered in the GUI fields, passes them to the algorithm and starts the algorithm. The Abort button aborts the optimisation run.

11.1.2 The Optimiser

There are three sub-blocks within the Optimiser block: System calculations, Cost function and Algorithm. System calculations are defined in Chapter 4 and are summarised in Appendix C. The cost function is defined in equation (29). The core of the Algorithm is the Discrete Differential Evolution as described in section 9.1.1.

Besides the DDE, the algorithm also checks for the convergence. It evaluates the following MATLAB expression for each coordinate

```
max(pop) - min(pop) == Tolerance;
```

where `pop` is the population vector.
For `Tolerance = 1`, the algorithm will terminate when the whole population of the corresponding coordinate has converged within the range of one discrete step from each other.
Increasing the value of the `Tolerance` parameter relaxes the termination criterion.

The flowchart of the algorithm is given in Appendix B.

11.1.3 The Library

The gain and the noise figure of the system will be measured in units of dB and the non-linearity measures in units of dBm. However, the system performance and the cost function are calculated with the linear values instead of the dB's and dBm's. So the building block tables will have to be presented to the optimiser containing the linear values.
The DC current consumption of a building block will be measured in mA and does not need to be converted.

Working with linear values provides significant time saving by avoiding the logarithm calculations within the optimisation loop.

The building block measurement tables are defined as a 3-dimensional array. The first array index is used to address the adjustment input x_1, the second the adjustment input x_2 and the third, the building block parameter. The order of the parameters is defined in section 3.1.

The format of the 'building block matrix', BBMTX, is 10x10x6. So there are 6 ten-by-ten matrices for each building block. For instance, BBMTX(4,5,2) addresses the element $x_1 = 4$, $x_2 = 5$ of the table 2 (Noise Factor) containing the value, e.g. 1.8603. For more building blocks of the same type the matrix can be extended by a dimension. For instance, if there are three LNA-mixer building blocks, the BBMTX format would be 10x10x6x3.
If the pre-interpolation is used to enhance the precision, as discussed in section 9.1, the BBMTX may be of other format, e.g. 100x100x6.

11.1.4 The Output

A number of parameters can be monitored during an optimisation run in the output window. An example is shown in Figure 37. In the first row the iteration number and the value of the best member's cost function can be monitored. In the convergence field there are six positions, corresponding with the index of the adjustment parameter. When the corresponding parameter converges, a 1 is displayed.

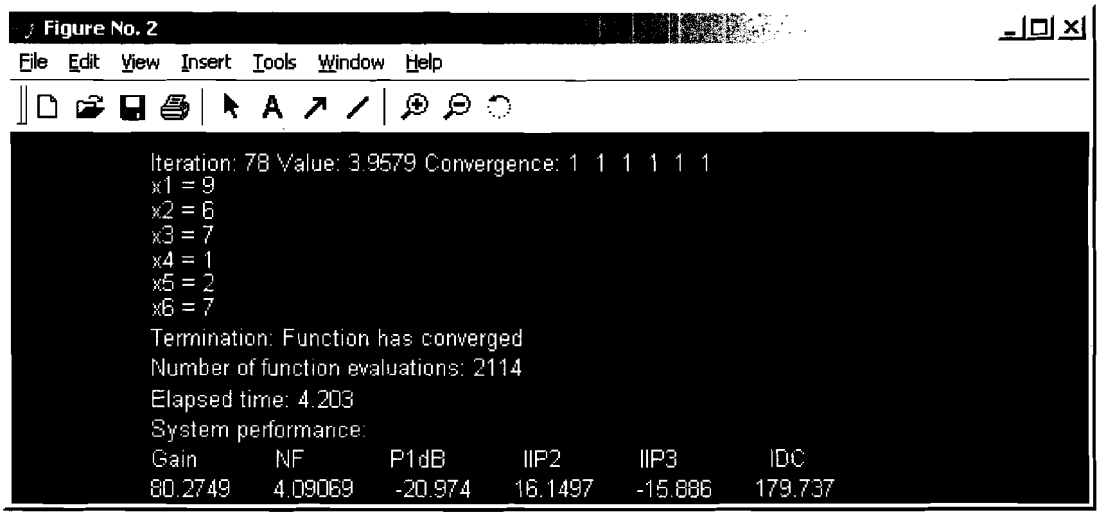


Figure 37: An example of the output

In the following six rows the values of the adjustment parameters are displayed. In the subsequent row the 'termination flag' is displayed. The possible flags are 'Function has converged', 'Maximum number of iterations exceeded' and 'Aborted'.
Next, the real number of function evaluations, which usually differs from 'number of iterations' multiplied by 'population count', is displayed. The elapsed time per optimisation run is in seconds. The last row displays the system performance. The gain and NF are in units of dB, the linearity measures in dBm and the current is in mA.

11.2 An example: IEEE 802.11a Receiver Optimisation

The usage and the specifics of the optimisation tool are best described through an example. For this, the demonstrator for the RF Platform project has been chosen. A system compliant to the IEEE 802.11a WLAN standard will be optimised.

The receiver and the transmitter are independent systems and their signal paths are separated by the baseband module. Therefore, they are optimised separately.

At present, the building blocks are not yet available, so neither are the measurement tables. A set of hypothetical, though realistic, measurement tables for the receiver chain will be used for the optimiser testing.
From the RF Platform study project the following receiver system specifications need to be met to comply with the IEEE 802.11a standard.

Table 8: IEEE 802.11a receiver specifications

Gain	> 80 dB
Noise Figure	< 7 dB
IP_{1dB}	> -23 dBm
IIP2	> 34 dBm
IIP3	> -17.5 dBm

Although it is possible to enter all the specifications in the optimiser at once, usually it is better to split this into a number of steps where each time a parameter is added.
First of all, it is useful to know what are the extremes of the design space. One can find this out by running an optimisation job each time on a single parameter. The results for this example are shown in Table 9.

Table 9: Best system performance per parameter

Gain	max. 86.80 dB
Noise Figure	min. 3.64 dB
IP_{1dB}	max. –16.38 dBm
IIP2	max. 42.34 dBm
IIP3	max. -12.66 dBm
I_{DC}	min. 108.37 mA

This means, in theory, that the specifications of Table 8 can be met, but the question is if they can be met simultaneously.
An optimisation could be started by setting the *NF* and *IIP3* weighting factors to 0.5. The resulting values of the adjustment inputs and the system performance are shown in Figure 38.

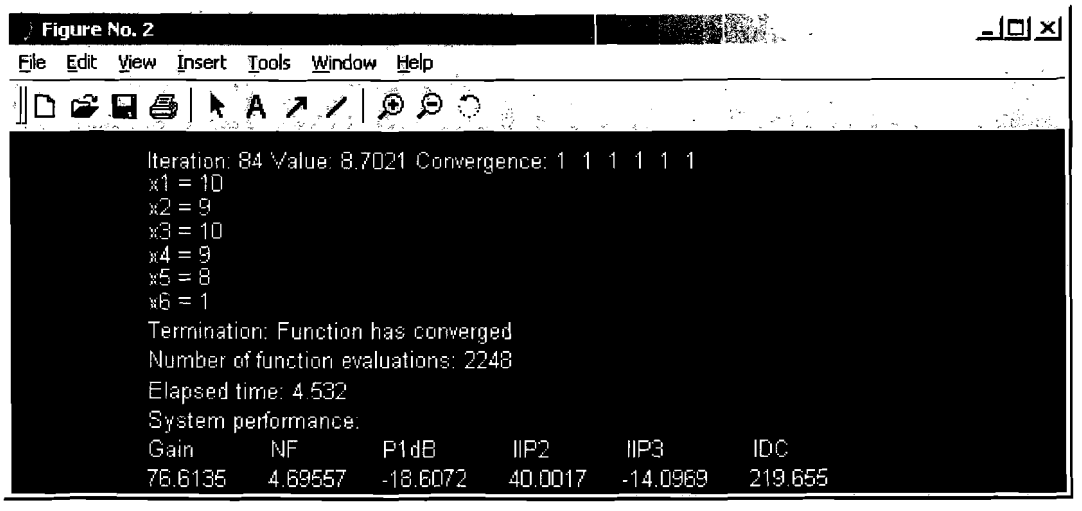


Figure 38: Output after first optimisation run

What can be seen in Figure 38 is that all the requirements are met, except for the gain. The gain is not a critical parameter and does not need to be optimised, so a following step could be to specify the gain as a constraint at > 80dB. The results of this run are shown in Table 10.

Table 10: The gain constraint at >80 dB

x_1	x_2	x_3	x_4	x_5	x_6
10	4	10	9	7	4
Gain	***NF***	IP_{1dB}	***IIP2***	***IIP3***	I_{DC}
80.0156	4.30631	-19.7494	39.3120	-14.7086	217.971

One of the system specifications could be that the DC current is not allowed to exceed the value of 200 mA. In that case, the following run could incorporate the DC current as a constraint at 200 mA, which is met after the optimisation run shown in Table 11.

Table 11: The DC current constraint at <200 mA

x_1	x_2	x_3	x_4	x_5	x_6
10	4	7	5	3	5
Gain	***NF***	IP_{1dB}	***IIP2***	***IIP3***	I_{DC}
80.0001	4.30253	-20.1106	39.0868	-14.8515	195.862

This example has shown the optimisation of a system in just a few runs. If the linearity is regarded more important than the noise performance, a further trade-off could be performed interactively in a few subsequent optimisation runs. The relative position of the slide bars, in Figure 36, could be used to specify the relative importance of these parameters.

11.3 Performance of the Method

The population of NP = 40 6-dimensional vectors converges within 100 iterations, for the *Tolerance* = 1. Typically, there are less than 0.25% of all the possible operating points evaluated.

The tool provides significant time saving in comparison with the Exhaustive search. For a system consisting of three building blocks, the method is typically 70 times faster. This is for the case that there is only one set of building blocks available per building block type. For instance, if there were two alternatives for the first block, two different systems would have to be calculated in case of the Exhaustive search. Then the method presented here would be slightly less than twice, around 120 times faster. This is caused by the fact that the number of points to be calculated is doubled in the case of the Exhaustive search whereas in the case of the RF-optimisation tool de number of points to be evaluated will increase only by a few percent. As larger the number of building blocks per type and as more measurement points, the larger the time improvement.

In addition to this, the method offers great flexibility when new building blocks need to be added to the library. For comparison, if a new block needs to be added for Exhaustive search, all possible combinations with other building blocks will have to be calculated. For instance, if a new LNA-mixer is developed, and there were already three VCO-PLLs and three IF-baseband blocks available, nine alternative systems would have to be calculated and stored in the library. The RF system optimisation tool avoids this.

12
Conclusions

The major goal of this Master's thesis was to design an optimisation method that will select the optimum composition of the building blocks and their optimal adjustment settings, based on the measurement tables and the product specifications. The selection of the optimisation parameters, system modelling and calculations, design of the cost function, implementation of two minimisation algorithms and the implementation and performance of the RF system optimisation tool have been discussed in previous chapters. This chapter presents the conclusions derived during the Master's thesis and recommendations for further work.

12.1 Conclusions

An important conclusion of this Master's thesis is that the optimisation aspect of the RF-platform project is feasible. It has been proven that it is possible to optimise a system of three building blocks, each having 100 different operating points, within a few seconds of calculation time. This allows for acceptable calculation times for iterative optimisation runs when the required performance is not met in a single run.
More than a factor 70 is gained in time in comparison with the Exhaustive search. The time gain gets almost doubled each time a building block per type is added, and increases even more when the number of measurement points is increased.

The definition of the measurement tables allows for a convenient extension of the optimisation parameters or the number of measurement points. A better precision can be achieved by interpolating the values from the measurement tables prior to feeding them to the optimiser. In this way, a lot of time is saved by avoiding the interpolation within the optimisation loop.

The choice for linear system calculations and cost function has significantly reduced the calculation time compared with a situation when the parameters are in units of dB and time-costly logarithm calculations need to be performed within the optimisation loop.

The Differential Evolution minimisation technique does not guarantee the global minimum, but will always lead to a good solution, if the algorithm control parameters are set well. If the adjustment parameter functions are simple, which has been assumed, the algorithm control parameters are already set well and do not have to be changed by the user.

The cost function definition in combination with the Differential Evolution algorithm leads to a feasible solution for each optimisation run. Even if unfeasible performance is requested, the method will provide the best feasible solution.

As a result of this Master's thesis, a model of the optimisation tool with its environment has been defined, as shown in Figure 35. If another implementation for any of the blocks shown in the figure may be chosen, the model can still offer a good starting point for this.

12.2 Recommendations

The described method offers a good starting point in the mobile and wireless transceiver design. A following step could be to use a circuit simulator, such as Advanced Design System (ADS), to include bond-wires, transmission lines, substrate and other effects. A commercially available simulator could also be used to replace the custom made simulator, the System calculations block in Figure 35.
In future, instead of using many measurement tables, a behavioural model of the building blocks, based on the measurement tables, could be stored in the library. This would provide a compact way of describing the building blocks.

Further improvement is possible on the system equations and the cost function definition. At this moment, all system calculations on the non-linearity parameters (P_{1dB}, $IP2$, $IP3$) are input-referred. This means that the gain of the latest stage in the cascade is not included in the equations. Therefore, it is also not included in the optimisation and can assume an arbitrary value. Further investigation is recommended to find a way to solve this.
The location of the optimum depends directly on the cost function. As a result of this work a cost function has been derived. Basically, the gain distribution within a receiver or transmitter chain is traded between the noise performance and the linearity performance. Further investigation could reveal if more trade-offs are possible and how to incorporate this into a mathematical description.

One of the advantages of the RF platform is the possibility to combine building blocks from different technologies, which could imply different supply voltages. Instead of using the DC current for the optimisation, power dissipation should be used in that case. No substantial changes need to be conducted in order to implement this. A simple multiplication of the building block table containing DC currents with the corresponding supply voltage will be adequate.

At present, the method is implemented to optimise the receiver and the transmitter chains separately. One of the arguments to optimise the transceiver as a whole could be the power dissipation trade-off between the receiver and the transmitter. The VCO-PLL building block is shared by the receiver and the transmitter, which could also plead for the simultaneous optimisation. In future, the optimisation method could be extended to incorporate this, however, at the cost of longer calculation times.

Appendix A
Derivation of cascaded 1-dB compression point, 2nd- and 3rd order intercept point

Input-referred cascaded 1-dB compression point

Consider a memoryless non-linear system,

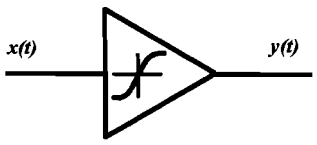


Figure 39: A non-linear system

which can be approximated with,

$$y(t) = \alpha_1 x(t) + \alpha_2 x^2(t) + \alpha_3 x^3(t) \tag{50}$$

If $x(t) = A\cos(\omega t)$, then

$$y(t) = \alpha_1 A\cos(\omega t) + \alpha_2 A^2 \cos^2(\omega t) + \alpha_3 A^3 \cos^3(\omega t)$$

using the relationships,

$$\cos^2(\gamma) = \frac{1}{2}[1 + \cos(2\gamma)]$$

$$\cos^3(\gamma) = \frac{3}{4}\cos(\gamma) + \frac{1}{4}\cos(3\gamma)$$

yields,

$$y(t) = \alpha_1 A\cos(\omega t) + \frac{1}{2}\alpha_2 A^2[1 + \cos(2\omega t)] + \frac{1}{4}\alpha_3 A^3[3\cos(\omega t) + \cos(3\omega t)]$$
$$= \frac{\alpha_2 A^2}{2} + \left(\alpha_1 A + \frac{3\alpha_3 A^3}{4}\right)\cos(\omega t) + \frac{\alpha_2 A^2}{2}\cos(2\omega t) + \frac{\alpha_3 A^3}{4}\cos(3\omega t)$$

At 1-dB compression point the voltage magnitude of the fundamental at the system output has dropped by 1 dB compared to the linear output[1]. The following equation is solved for the input voltage magnitude at which this occurs,

[1] occurs when α_3 is negative, otherwise there is expansion instead of compression.

$$20\log\left|\alpha_1 A_{1dB} + \frac{3}{4}\alpha_3 A_{1dB}^3\right| = 20\log|\alpha_1 A_{1dB}| - 1\ dB$$

$$\left|\alpha_1 + \frac{3}{4}\alpha_3 A_{1dB}^2\right| = |\alpha_1| \cdot 10^{-\frac{1}{20}}$$

$$A_{1dB}^2 = \left|\frac{4}{3} \cdot \left(10^{-\frac{1}{20}} - 1\right) \cdot \frac{\alpha_1}{\alpha_3}\right| \tag{51}$$

The voltage magnitude at the input of the system at which 1-dB compression occurs is,

$$A_{1dB} = \sqrt{0.145\left|\frac{\alpha_1}{\alpha_3}\right|}$$

Now, consider a cascade of two memoryless non-linear stages,

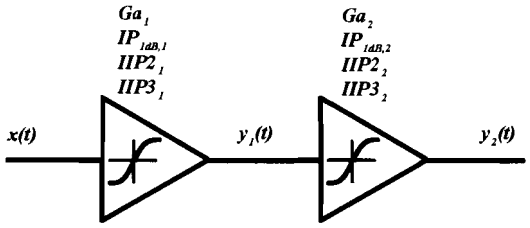


Figure 40: Cascaded non-linear stages

Let,

$$y_1(t) = \alpha_1 x(t) + \alpha_2 x^2(t) + \alpha_3 x^3(t)$$
$$y_2(t) = \beta_1 y_1(t) + \beta_2 y_1^2(t) + \beta_3 y_1^3(t)$$

then,

$$y_2(t) = \beta_1\left[\alpha_1 x(t) + \alpha_2 x^2(t) + \alpha_3 x^3(t)\right] + \beta_2\left[\alpha_1 x(t) + \alpha_2 x^2(t) + \alpha_3 x^3(t)\right]^2 + \beta_3\left[\alpha_1 x(t) + \alpha_2 x^2(t) + \alpha_3 x^3(t)\right]^3$$

In accordance with (50), for the equivalent system only the effect of up to the third-order terms is assumed while higher order terms are not considered, which yields,

$$y_2(t) = \alpha_1\beta_1 \cdot x(t) + \left(\alpha_2\beta_1 + \alpha_1^2\beta_2\right) \cdot x^2(t) + \left(\alpha_3\beta_1 + 2\alpha_1\alpha_2\beta_2 + \alpha_1^3\beta_3\right) \cdot x^3(t) \tag{52}$$

and if $x(t) = A\cos(\omega t)$ then,

$$y_2(t) = \alpha_1\beta_1 A\cos(\omega t) + \frac{3}{4}\alpha_3\beta_1 A^3\cos(\omega t) + \frac{3}{2}\alpha_1\alpha_2\beta_2 A^3\cos(\omega t) + \frac{3}{4}\alpha_1^3\beta_3 A^3\cos(\omega t)$$
$$+ \textit{constant term} + \textit{harmonics}$$

since we are interested only in the magnitude of the fundamental.

Again, the following equation is solved for the voltage magnitude at which 1-dB compression occurs,

$$20\log\left|\alpha_1\beta_1 A_{1dB} + \frac{3}{4}\alpha_3\beta_1 A_{1dB}^3 + \frac{3}{2}\alpha_1\alpha_2\beta_2 A_{1dB}^3 + \frac{3}{4}\alpha_1^3\beta_3 A_{1dB}^3\right| = 20\log\left|\alpha_1\beta_1 A_{1dB}\right| - 1dB$$

$$\left|\alpha_1\beta_1 A_{1dB} + \frac{3}{4}\alpha_3\beta_1 A_{1dB}^3 + \frac{3}{2}\alpha_1\alpha_2\beta_2 A_{1dB}^3 + \frac{3}{4}\alpha_1^3\beta_3 A_{1dB}^3\right| = \left|\alpha_1\beta_1 A_{1dB}\right|\cdot 10^{-\frac{1}{20}}$$

$$\left|\alpha_1\beta_1 + \frac{3}{4}\alpha_3\beta_1 A_{1dB}^2 + \frac{3}{2}\alpha_1\alpha_2\beta_2 A_{1dB}^2 + \frac{3}{4}\alpha_1^3\beta_3 A_{1dB}^2\right| = \left|\alpha_1\beta_1\right|\cdot 10^{-\frac{1}{20}}$$

$$A_{1dB}^2 = \frac{\left|10^{-\frac{1}{20}} - 1\right|\cdot\left|\alpha_1\beta_1\right|}{\left|\frac{3}{4}\alpha_3\beta_1 + \frac{3}{2}\alpha_1\alpha_2\beta_2 + \frac{3}{4}\alpha_1^3\beta_3\right|}$$

and using the rule $|a+b| \le |a| + |b|$ we obtain for the worst case,

$$\frac{1}{A_{1dB}^2} = \frac{\left|\frac{3}{4}\alpha_3\beta_1\right| + \left|\frac{3}{2}\alpha_1\alpha_2\beta_2\right| + \left|\frac{3}{4}\alpha_1^3\beta_3\right|}{\left|10^{-\frac{1}{20}} - 1\right|\cdot\left|\alpha_1\beta_1\right|}$$

$$= \frac{\left|\frac{3}{4}\alpha_3\beta_1\right|}{\left|10^{-\frac{1}{20}} - 1\right|\cdot\left|\alpha_1\beta_1\right|} + \frac{\left|\frac{3}{2}\alpha_1\alpha_2\beta_2\right|}{\left|10^{-\frac{1}{20}} - 1\right|\cdot\left|\alpha_1\beta_1\right|} + \frac{\left|\frac{3}{4}\alpha_1^3\beta_3\right|}{\left|10^{-\frac{1}{20}} - 1\right|\cdot\left|\alpha_1\beta_1\right|}$$

making the use of the results found for a single stage in equation (51), we may write,

$$\frac{1}{A_{1dB}^2} = \frac{1}{A_{1dB,1}^2} + \frac{\left|\frac{3}{2}\alpha_2\beta_2\right|}{\left|10^{-\frac{1}{20}} - 1\right|\cdot\left|\beta_1\right|} + \frac{\alpha_1^2}{A_{1dB,2}^2}$$

The linear gain is in general much higher than the high-order gain, we may assume $|\beta_1| >> |\alpha_2 \beta_2|$, which yields,

$$\frac{1}{A_{1dB}^2} = \frac{1}{A_{1dB,1}^2} + \frac{\alpha_1^2}{A_{1dB,2}^2}$$

Latter can be written in terms of *linear powers*, i.e. not in dB, and *linear gains*,

$$\frac{1}{ip_{1dB}} = \frac{1}{ip_{1dB,1}} + \frac{Ga_1}{ip_{1dB,2}}$$

By nesting the previous equation, the equivalent linear power level at the input of N stages at which the gain has degraded by 1 dB is given by,

$$\frac{1}{ip_{1dB}} = \frac{1}{ip_{1dB,1}} + \frac{Ga_1}{ip_{1dB,2}} + \frac{Ga_1 Ga_2}{ip_{1dB,3}} + \ldots + \frac{Ga_1 Ga_2 \ldots Ga_{N-1}}{ip_{1dB,N}}$$

Input-referred cascaded 2nd and 3rd order intercept points

Consider again a memoryless non-linear system, as shown in Figure 39, which can be approximated with equation (50).

If $x(t) = A_1 \cos(\omega_1 t) + A_2 \cos(\omega_2 t)$, then

$$y(t) = \alpha_1\left[A_1 \cos(\omega_1 t) + A_2 \cos(\omega_2 t)\right] + \alpha_2\left[A_1 \cos(\omega_1 t) + A_2 \cos(\omega_2 t)\right]^2 + \alpha_3\left[A_1 \cos(\omega_1 t) + A_2 \cos(\omega_2 t)\right]^3$$

using the relationships,

$$\cos^2(\gamma) = \frac{1}{2}\left[1 + \cos(2\gamma)\right]$$

$$\cos^3(\gamma) = \frac{3}{4}\cos(\gamma) + \frac{1}{4}\cos(3\gamma)$$

yields,

$$\begin{aligned} y(t) = {} & \frac{1}{2}\alpha_2\left(A_1^2 + A_2^2\right) + \left(\alpha_1 A_1 + \frac{3}{2}\alpha_3 A_1 A_2^2 + \frac{3}{4}\alpha_3 A_1^3\right)\cos(\omega_1 t) + \left(\alpha_1 A_2 + \frac{3}{2}\alpha_3 A_1^2 A_2 + \frac{3}{4}\alpha_3 A_2^3\right)\cos(\omega_2 t) + \\ & + \alpha_2 A_1 A_2 \cos\left((\omega_1 + \omega_2)t\right) + \alpha_2 A_1 A_2 \cos\left((\omega_1 - \omega_2)t\right) + \frac{1}{2}\alpha_2 A_1^2 \cos(2\omega_1 t) + \frac{1}{2}\alpha_2 A_2^2 \cos(2\omega_2 t) + \\ & + \frac{1}{4}\alpha_3 A_1^3 \cos(3\omega_1 t) + \frac{1}{4}\alpha_3 A_2^3 \cos(3\omega_2 t) + \frac{3}{4}\alpha_3 A_1^2 A_2 \cos\left((2\omega_1 + \omega_2)t\right) + \frac{3}{4}\alpha_3 A_1^2 A_2 \cos\left((2\omega_1 - \omega_2)t\right) + \\ & + \frac{3}{4}\alpha_3 A_1 A_2^2 \cos\left((2\omega_2 + \omega_1)t\right) + \frac{3}{4}\alpha_3 A_1 A_2^2 \cos\left((2\omega_2 - \omega_1)t\right) \end{aligned}$$

The spectral components of *y(t)* are shown in Figure 41, with $\omega = 2\pi f$.

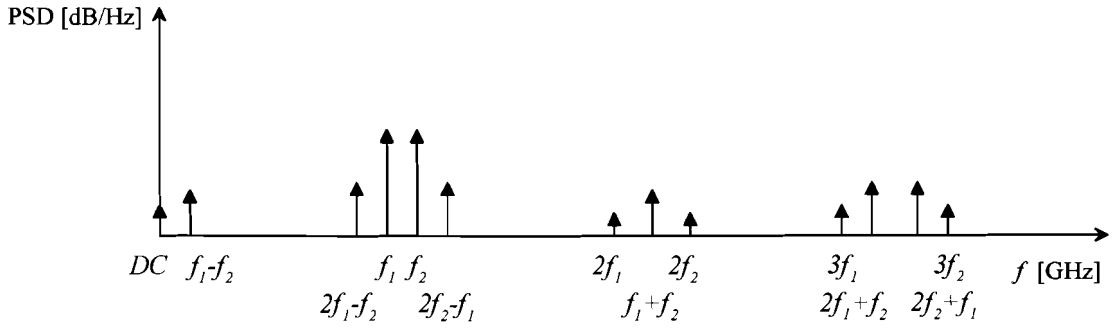


Figure 41: Spectral components of *y(t)* with two-tone input

Input-referred 2nd order intercept point

Let's assume that $A_1 = A_2 = A$. The 2nd order intercept point occurs (hypothetically) when the voltage magnitude of one of the fundamentals at the output of the non-linear stage is equal to the voltage magnitude of one of the 2nd order intermodulation products. For the voltage magnitude of the fundamental at the input of the non-linear stage at which this would occur holds

$$\left|\alpha_1 A + \frac{9}{4}\alpha_3 A^3\right| = \left|\alpha_2 A^2\right|$$

then let's assume that $\alpha_1 >> \frac{9}{4}\alpha_3 A^2$, which yields

$$A_{IIP2} = \frac{\alpha_1}{\alpha_2} \tag{53}$$

Now, consider again a cascade of two memoryless non-linear stages, as shown in Figure 40,

and if $x(t) = A\cos(\omega_1 t) + A\cos(\omega_2 t)$ then,

$$y_2(t) = \alpha_1\beta_1 A\cos(\omega_1 t) + \left(\alpha_2\beta_1 A^2 + \alpha_1^2\beta_2 A^2\right)\cos\left((\omega_1 - \omega_2)t\right)$$
$$+ \textit{constant term} + \textit{harmonics} + \textit{other intermodulation products}$$

since we are interested only in the magnitude of one of the fundamentals and one of the 2nd order intermodulation products.

Then,

$$\left|\alpha_1\beta_1 A\right| = \left|\alpha_2\beta_1 A^2 + \alpha_1^2\beta_2 A^2\right|$$

$$A_{IIP2} = \frac{\left|\alpha_1\beta_1\right|}{\left|\alpha_2\beta_1 + \alpha_1^2\beta_2\right|}$$

and using the rule $|a+b| \le |a| + |b|$ we obtain for the worst case,

$$\frac{1}{A_{IIP2}} = \frac{\left|\alpha_2\beta_1\right| + \left|\alpha_1^2\beta_2\right|}{\left|\alpha_1\beta_1\right|}$$

making the use of the results found for a single stage in equation (53), we may write,

$$\frac{1}{A_{IIP2}} = \frac{1}{A_{IIP2,1}} + \frac{\alpha_1}{A_{IIP2,2}}$$

Latter can be written in terms of *linear powers,* i.e. not in dB, and *linear gains,*

$$\frac{1}{\sqrt{iip2}} = \sqrt{\frac{1}{iip2_1}} + \sqrt{\frac{Ga_1}{iip2_2}}$$

By nesting the previous equation, the equivalent input-referred 2nd order intercept point of N stages is given by

$$\frac{1}{\sqrt{iip2}} = \sqrt{\frac{1}{iip2_1}} + \sqrt{\frac{Ga_1}{iip2_2}} + \sqrt{\frac{Ga_1 Ga_2}{iip2_3}} + \ldots + \sqrt{\frac{Ga_1 Ga_2 .. Ga_{N-1}}{iip2_N}}$$

Input-referred 3rd order intercept point

Let's assume that $A_1 = A_2 = A$. The 3rd order intercept point occurs (hypothetically) when the voltage magnitude of one of the fundamentals at the output of the non-linear stage is equal to the voltage magnitude of one of the 3rd order intermodulation products. For the voltage magnitude of the fundamental at the input of the non-linear stage at which this would occur holds

$$\left|\alpha_1 A + \frac{9}{4}\alpha_3 A^3\right| = \left|\frac{3}{4}\alpha_3 A^3\right|$$

then let's assume that $\alpha_1 >> \frac{9}{4}\alpha_3 A^2$, which yields

$$A_{IIP3} = \sqrt{\frac{4}{3}\left|\frac{\alpha_1}{\alpha_3}\right|} \tag{54}$$

Now, consider a cascade of two memoryless non-linear stages, as shown in Figure 1.

Using the equivalent system equation (52) and the results for one stage found in equation (54) we substitute for $\alpha_1 \to \alpha_1\beta_1$, and for $\alpha_3 \to \alpha_3\beta_1 + 2\alpha_1\alpha_2\beta_2 + \alpha_1^3\beta_3$, which yields

$$A_{IIP3} = \sqrt{\frac{4}{3}\left|\frac{\alpha_1\beta_1}{\alpha_3\beta_1 + 2\alpha_1\alpha_2\beta_2 + \alpha_1^3\beta_3}\right|}$$

and again using the rule $|a+b| \le |a| + |b|$ we obtain for the worst case,

$$\frac{1}{A_{IIP3}^2} = \frac{3}{4}\frac{|\alpha_3\beta_1| + |2\alpha_1\alpha_2\beta_2| + |\alpha_1^3\beta_3|}{|\alpha_1\beta_1|}$$

making the use of the results found for a single stage in equation (54), we may write,

$$\frac{1}{A_{IIP3}^2} = \frac{1}{A_{IIP3,1}^2} + \frac{3\alpha_2\beta_2}{2\beta_1} + \frac{\alpha_1^2}{A_{IIP3,2}^2}$$

The linear gain is in general much higher than the high-order gain, we may assume $|\beta_1| >> |\alpha_2\beta_2|$, which yields,

$$\frac{1}{A_{IIP3}^2} = \frac{1}{A_{IIP3,1}^2} + \frac{\alpha_1^2}{A_{IIP3,2}^2}$$

Latter can be written in terms of *linear powers*, i.e. not in dB, and *linear gains*,

$$\frac{1}{iip3} = \frac{1}{iip3_1} + \frac{Ga_1}{iip3_2}$$

Finally, by nesting the previous equation, the equivalent input-referred 3rd order intercept point of N stages is given by

$$\frac{1}{iip3} = \frac{1}{iip3_1} + \frac{Ga_1}{iip3_2} + \frac{Ga_1Ga_2}{iip3_3} + \ldots + \frac{Ga_1Ga_2..Ga_{N-1}}{iip3_N}$$

Appendix B
Flowcharts of Genetic Algorithm and Discrete Differential Evolution Algorithm

Genetic Algorithm flowchart

1. Set k:=0; form initial population P(0)
2. Evaluate P(k)
3. If stopping criterion satisfied, then stop
4. Select M(k) from P(k)
5. Evolve M(k) to form P(k+1)
6. Set k:=k+1, go to step 2

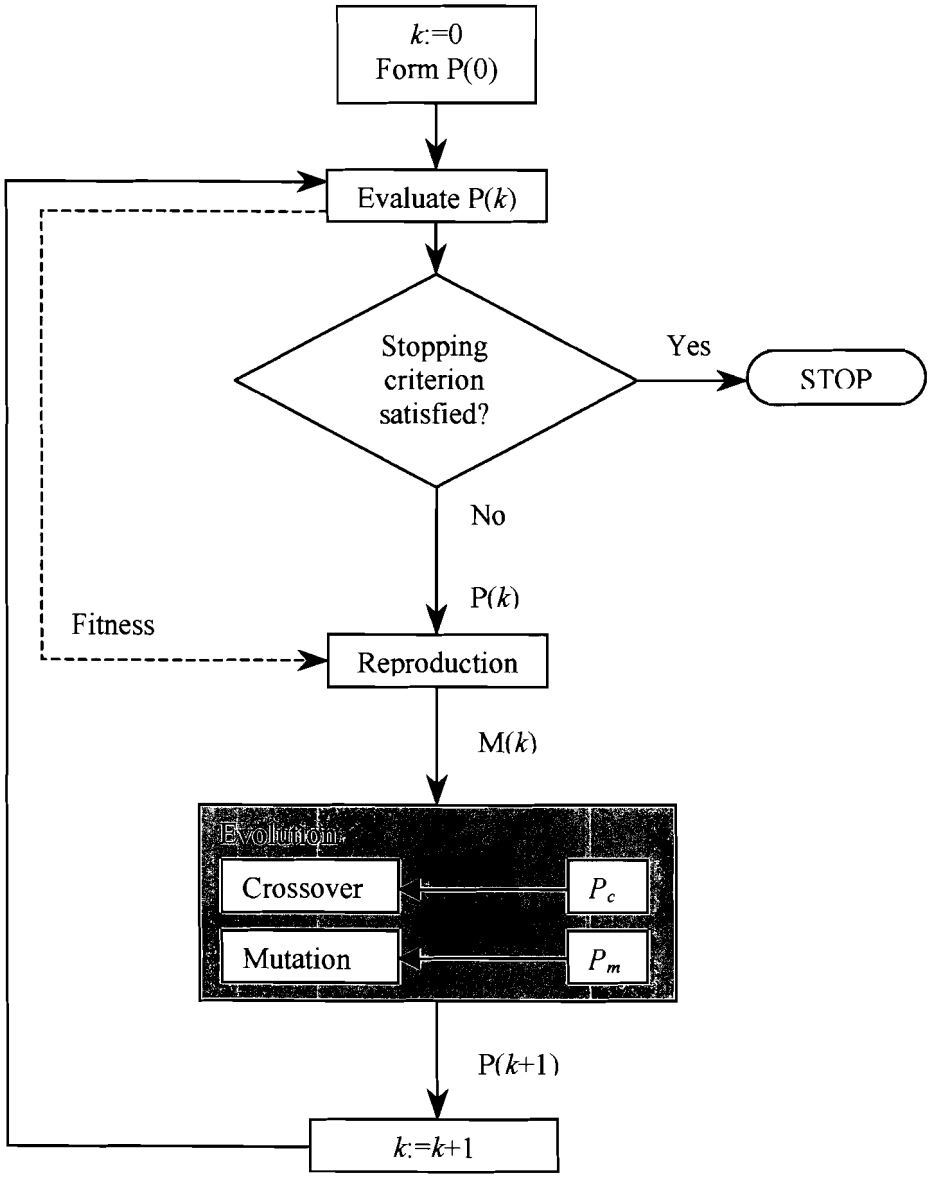


Figure 42: Genetic Algorithm flowchart

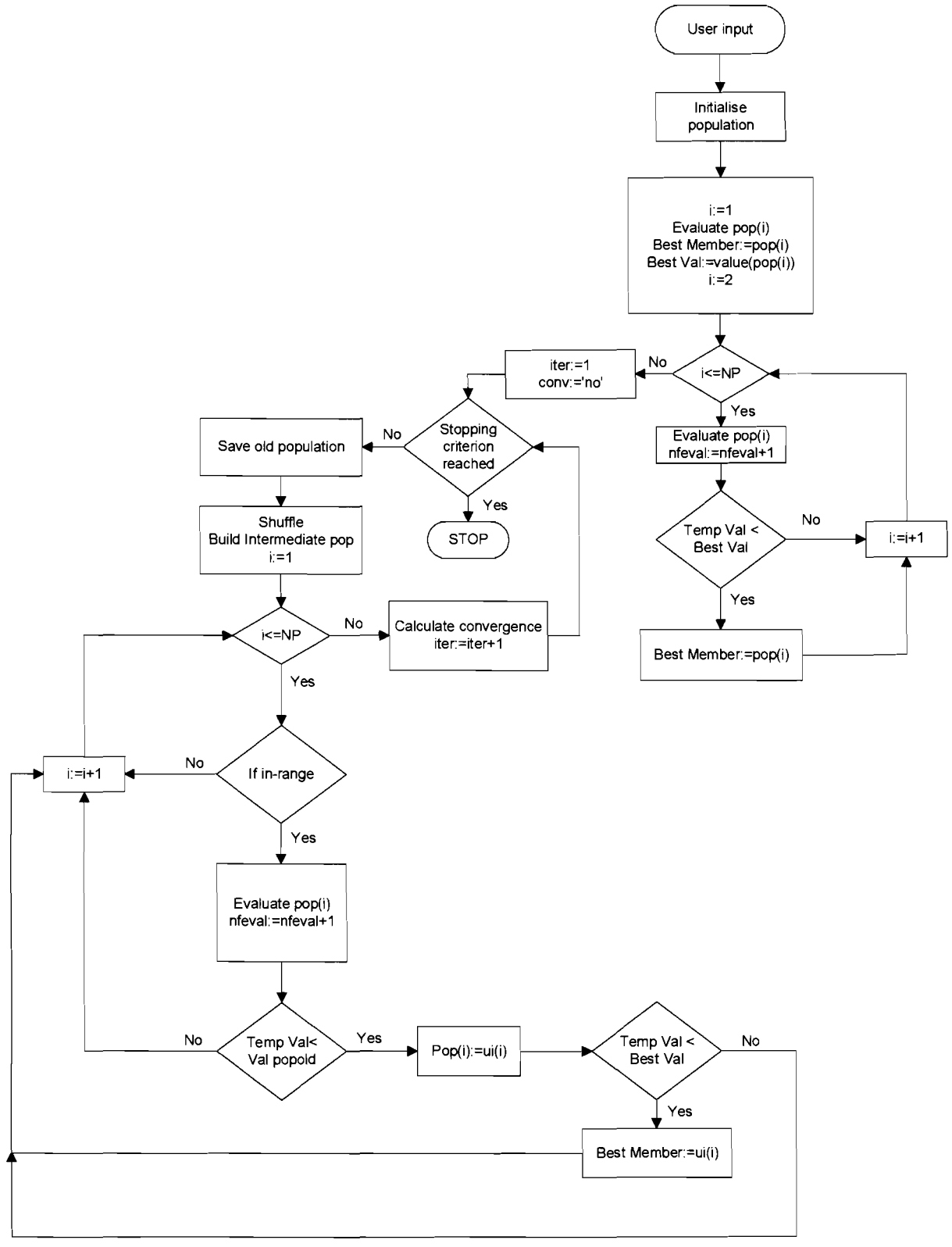


Figure 43: Discrete Differential Evolution algorithm flowchart

Appendix C
System calculations

Linear factor system calculations

$$Ga = Ga_1 \cdot Ga_2 \cdot Ga_3 \cdot \ldots \cdot Ga_N \qquad \textit{dimensionless}$$

$$F = F_1 + \frac{F_2 - 1}{Ga_1} + \frac{F_3 - 1}{Ga_1 Ga_2} + \ldots + \frac{F_N - 1}{Ga_1 Ga_2 .. Ga_{N-1}} \qquad \textit{dimensionless}$$

$$\frac{1}{ip_{1dB}} = \frac{1}{ip_{1dB,1}} + \frac{Ga_1}{ip_{1dB,2}} + \frac{Ga_1 Ga_2}{ip_{1dB,3}} + \ldots + \frac{Ga_1 Ga_2 .. Ga_{N-1}}{ip_{1dB,N}} \qquad \left[\frac{1}{mW}\right]$$

$$\frac{1}{\sqrt{iip2}} = \sqrt{\frac{1}{iip2_1}} + \sqrt{\frac{Ga_1}{iip2_2}} + \sqrt{\frac{Ga_1 Ga_2}{iip2_3}} + \ldots + \sqrt{\frac{Ga_1 Ga_2 .. Ga_{N-1}}{iip2_N}} \qquad \left[\frac{1}{\sqrt{mW}}\right]$$

$$\frac{1}{iip3} = \frac{1}{iip3_1} + \frac{Ga_1}{iip3_2} + \frac{Ga_1 Ga_2}{iip3_3} + \ldots + \frac{Ga_1 Ga_2 .. Ga_{N-1}}{iip3_N} \qquad \left[\frac{1}{mW}\right]$$

$$I_{DC} = I_{DC,1} + I_{DC,2} + I_{DC,3} + \ldots + I_{DC,N} \qquad [mA]$$

dB to linear and vice versa

$G = 10 \cdot \log_{10}(Ga)$	$[dB]$	$\Leftrightarrow$	$Ga = 10^{\frac{G}{10}}$	*dimensionless*
$NF = 10 \cdot \log_{10}(F)$	$[dB]$	$\Leftrightarrow$	$F = 10^{\frac{NF}{10}}$	*dimensionless*
$IIP3 = 10 \cdot \log_{10}(iip3)$	$[dBm]$	$\Leftrightarrow$	$iip3 = 10^{\frac{IIP3}{10}}$	$[mW]$
$IIP2 = 10 \cdot \log_{10}(iip2)$	$[dBm]$	$\Leftrightarrow$	$iip2 = 10^{\frac{IIP2}{10}}$	$[mW]$
$IP_{1dB} = 10 \cdot \log_{10}(ip_{1dB})$	$[dBm]$	$\Leftrightarrow$	$ip_{1dB} = 10^{\frac{IP_{1dB}}{10}}$	$[mW]$

Appendix D
Graphical Output

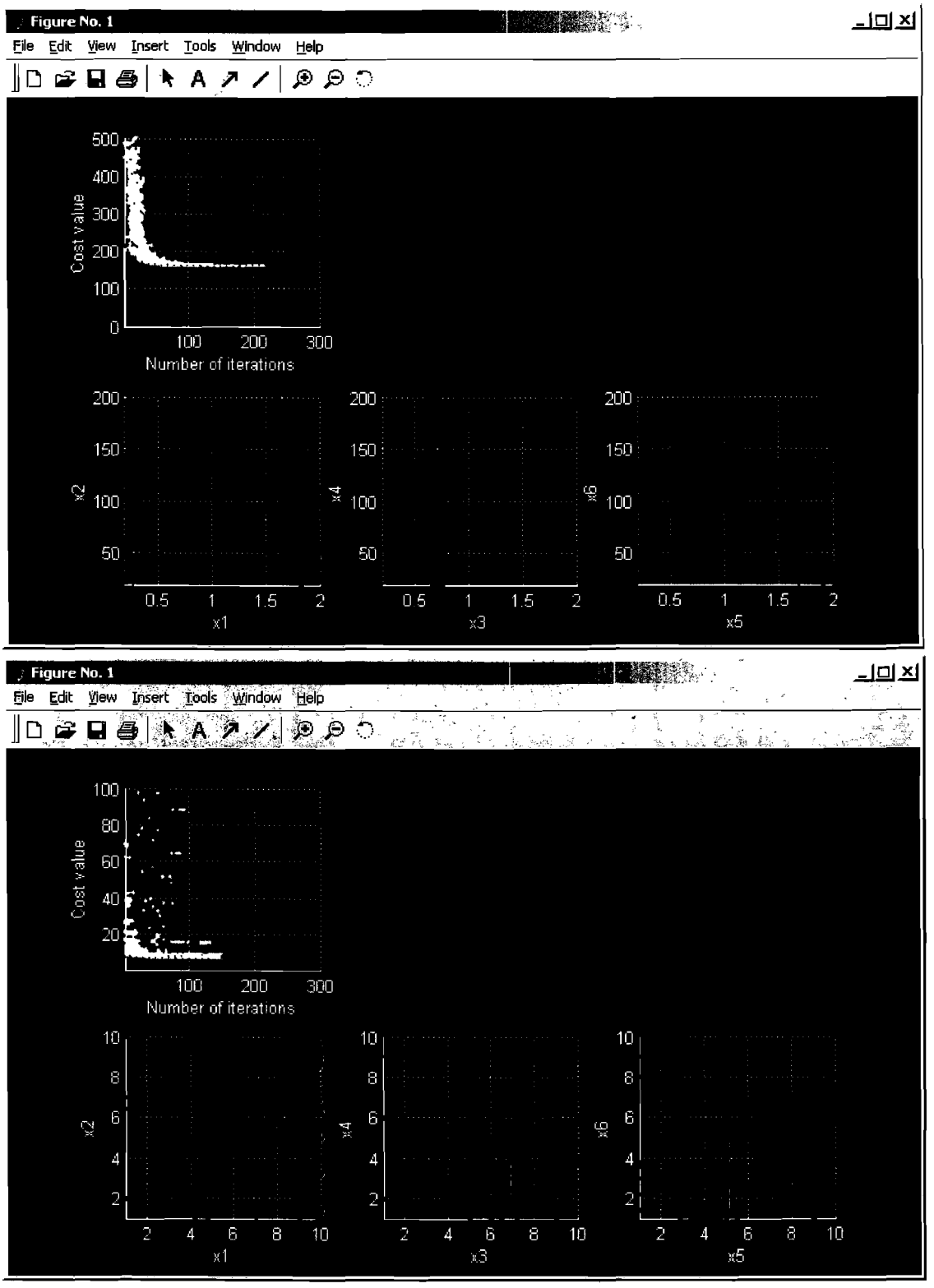


Figure 44: Two examples of graphical output of the RF system optimisation tool

Appendix E
References

[1] Vancorenland, P. and C. de Ranter, M. Steyaert, G. Gielen
Optimal RF design using smart evolutionary algorithms, IEEE Proceedings 2000: Design Automation Conference, 2000, p.7-10.

[2] Dermentzoglou, L. and A. Arapoyanni, A. Pneumatikakis
A direct conversion receiver analysis for multistandard wireless applications, Proceedings on 10th Mediterranean Electrotechnical Conference, Information Technology and Electrotechnology for the Mediterranean Countries, IEEE, 1998, Vol.1, p.318-21.

[3] Crols, J. and S. Donnay, M. Steyaert, G. Gielen
A high-level design and optimization tool for analog RF receiver front-ends, 1995 IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, IEEE Comput. Soc. Press, 1995, p.550-3.

[4] Veselinovic, P. and D. Leenaerts, W. van Bokhoven, F. Leyn, F. Proesmans, G. Gielen, W. Sansen
A flexible topology selection program as part of an analog synthesis system, Proceedings on the European Design and Test Conference, IEEE Comput. Soc. Press, 1995, p.119-23.

[5] Droinet, Y.
An innovative N-ZIF RFIC architecture for multimode cellular handsets, Microwave Engineering, Oct. 2001, p.33-7.

[6] Storn, R. and K. Price
Differential evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces, International Computer Science Institute, 1995, TR-95-012.

[7] Pennock, S.R. and M.H. Burchett, M.A. Redfern, X. Zhang
A building block approach to the design of prototype RF systems, IEE Colloquium on The RF Design Scene, 1996, p.7/1-5.

[8] Fang Wang and V.K. Devabhaktuni, Xi. Changgeng, Qi Jun Zhang
Neural network structures and training algorithms for RF and microwave applications, International Journal of RF and Microwave Computer Aided Engineering, May 1999, Vol.9, no.3, p.216-40.

[9] Alotto, P. and A. Caiti, G. Molinari, M. Repetto
A multiquadrics-based algorithm for the acceleration of simulated annealing optimization procedures, IEEE Transactions on Magnetics, May 1996, Vol.32, no.3, p.1198-201.

[10] Kundert, K.
Simulation of nonlinear circuits in the frequency domain, IEEE Transactions on Computer-Aided Design, 1986, Vol. 5, no. 4, p. 512-35.

[11] De Smedt, B. and G. Gielen
Models for systematic design and verification of frequency synthesizers, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Oct. 1999, Vol.46, no.10, p.1301-8.

[12] Gielen, G and R.Rutenbar
Computer-Aided-Design of Analog and Mixed-Signal Integrated-Circuits, Proceedings of the IEEE, 2000, Vol.88, no.12, pp 1825-1852..

[13] Kruiskamp, M.
Analog design automation using genetic algorithms and polytopes, Doctoral dissertation, Eindhoven University of Technology, The Netherlands, 1996.

[14] Gielen, G. and J. Huijsing, R. van der Plassche and W. Sansen, Ed.
Top-Down Design of Mixed-Mode Systems: Challenges and Solutions, Kluwer, 1998.

[15] Holland, J.H.
Adaptation in natural and artificial systems, University of Michigan Press, 1975.

[16] Goldberg, D.E.
Genetic algorithms in search, optimization and machine learning, Addison-Weslay, 1989.

[17] Barnasconi, M.
Architecture comparison for RF platforms, Internal laboratory report, ERA/IR02016.01.

[18] Neelen, A.H.
Partition of the functional building blocks for the RF Platform, Internal laboratory report, ERA/IR02019.1.

[19] Côme, B. et al
Impact of front-end non-idealities on Bit Error Rate performances of WLAN-OFDM transceivers, Microwave Journal, pp.126-140, February 2001.

[20] Soorapanth, T. and T.H. Lee
RF Linearity of Short-Channel MOSFETs, Stanford University, http://www-smirc.stanford.edu/papers/cancun97p-chet.pdf

[21] Chong, E.K.P. and S.H. Zak
An Introduction to Optimisation, Wiley-Interscience Publication, New York, 1996.

[22] Nelder, J.A. and R. Mead
A simplex method for function minimisation, Computer Jurnal, Vol.7, Jan 1965, p308-313.

[23] Metropolis, N. et al
Equation of State Calculations by Fast Computing Machines, J. Chem. Phys., 21, 6, 1087-1092, 1953.

[24] Matteij, R.M.M.
Inleiding Numerieke Methoden (2N210), Course syllabus, Technische Universiteit Eindhoven.

[25] Bliek, C. et al
Algorithms for solving Nonlinear Constrained an Optimisation Problems: The State of The Art, June 8 2001, http://www.mat.univie.ac.at/~neum/glopt/coconut/StArt.html.

[26] Keane, A.J.
A brief comparison of some evolutionary optimisation methods, pp. 255-272 in Modern Heuristic Search Methods, ed. V. Rayward-Smith, I. Osman, C. Reeves and G. D. Smith,J. Wiley, ISBN 0 471 96280 5, 1996.

[27] Storn, R.
Differential Evolution Homepage, http://http.icsi.berkeley.edu/~storn/code.html.

[28] Gämperle, R. et al
A Parameter Study for Differential Evolution, Institute of Comp. Sciences, Switzerland, http://www.icos.ethz.ch/research/wseas02.pdf.

[29] Ingber, L. and B. Rosen
Genetic Algorithms and Very Fast Reannealing: A comparison, Mathematical and Computer Modelling, 16(11) 1992, 87-100.

[30] Storn, R.
System Design by Constrain Adaptation and Differential Evolution, International Computer Science Institute, TR-96-039, November 1996.