TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Entwurfsautomatisierung

# Discrete Sizing of Analog Integrated Circuits

**Christian Michael Pehl**

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

**Doktor-Ingenieurs**

genehmigten Dissertation.

| | | |
|---|---|---|
| Vorsitzender: | | Univ.-Prof. Paolo Lugli, Ph.D. |
| Prüfer der Dissertation: | 1. | Priv.-Doz. Dr.-Ing. Helmut Gräb |
| | 2. | Univ.-Prof. Dr. Sc. (ETH) Samarjit Chakraborty |

Die Dissertation wurde am 16.04.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 12.11.2012 angenommen.

"Everything begins with an idea"
*Earl Nightingale*

## Acknowledgments

As almost everything, also this work started with an idea. But an idea is nothing without the scope for creative thinking or without enough time and discussions which let the idea grow and come to maturity. Therefore, I initially want to thank Prof. Ulf Schlichtmann for the opportunity to carry out this thesis at the Institute for Electronic Design Automation. I also want to thank PD Dr. Helmut Gräb for many fruitful discussions and for his guidance during this work. Without his feedback some of the ideas I hit on in this thesis would never have been born. Also Prof. Samarjit Chakraborty and Prof. Paolo Lugli, I want to give thanks for their interest and the time they spend for reading and for analyzing my thesis.

To all my colleagues from the Institute for Electronic Design Automation – especially the members of the working group on analog design automation – many thanks not only for scientific discussions but also for more than five years of good memories. Not to forget: Many thanks to all students who worked under my supervision and supported me in teaching and research.

I also wish to express my gratitude to my family, especially to my wife Yvonne, who encouraged and motivated me also in difficult times, and to my mother Ursula for all her help. I want to mention my father Josef, as well, who died during this thesis. Without him I would not have had the opportunity to do my studies to the extent I did.

Finally, I want to thank my friends for the rests from the scientific ideas, for listening to strange mathematical and technical stuff, and for each kind of support during the last years. Especially, I want to thank Evi Söding for her kind support in linguistic concerns.

Even if almost everything starts with an idea: only an idea does not change anything if it is not realized. Therefore, thanks once again to everybody who contributed to the realization of my ideas in any way.

# Contents

# Chapter 1

# Introduction

The design of integrated circuits (ICs) is a core challenge in today's electronics industry. Especially in the field of automotive and communication, a high innovation rate is required. The limiting factor in today's design flow is the designers' productivity, which is much lower than technology capabilities. This discrepancy between technological capabilities and designer productivity is called the hardware design gap [ITR09] and is visualized in Figure 1.1. To exploit technology capabilities, which follow the expansion rate known as Moore's law [Moo65], designers' productivity must be increased by a factor of 50 [ITR09]. However, it can be seen that the productivity gap is growing. One method to counteract the increasing design gap is an advance in circuit design automation.

In contrast to the well advanced automation in the design of digital circuits, the design of analog circuits, like amplifiers or A/D- and D/A-converters, is still predominantly manual work in practice [ITR10]. One reason for the lack of automation is affiliated to the problem that many important issues are not covered satisfactorily in today's

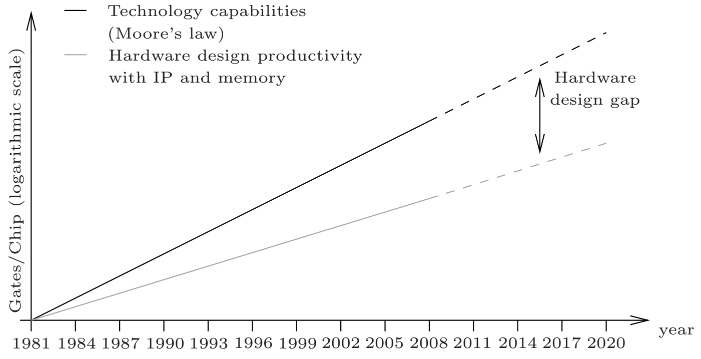


Figure 1.1: Technology capabilities and hardware design productivity over time following [ITR09] (technology capabilities are doubled after 3 years)

commercial analog design tools. Among other things, [ITR10] mentions that discrete parameters, and yield and reliability requirements are not or insufficiently considered by interactive design solutions.

Along with the low level of automation, the design of an analog circuit is a highly complex task, although analog components typically consist of only 10 to 100 devices [RGR07]. The high complexity is predominantly caused by the consideration of many circuit performances behaving competitively and measured in a continuous domain. As a consequence, the design of analog components requires a high percentage of the overall design effort, is error prone, and causes approximately half of the redesigns [Rut]. Additionally, the growth rate of non-memory ICs containing analog and RF components ($\approx 66\%$ in 2008 [Rut]) is higher than the growth rate of non-mixed-signal ICs. Consequently, analog design strongly contributes to the increasing design gap.

To reduce the design effort for analog design and to help designers to avoid errors which cause redesigns, the automation of analog circuit design must be enhanced. This can only be achieved if the problems described in [ITR10] are solved. This thesis contributes to that goal by presenting an algorithm which considers discrete design parameters, different operating conditions of a circuit, and variations in the manufacturing process during the step of analog sizing.

## 1.1 Analog Design Flow and Its Automation

Developing a new system on chip, one of the first project decisions is which parts of the system should be realized in hardware and software, respectively. The hardware part is further divided into several functional blocks, and it must be decided which digital and analog components are required. For each component, the behavior of the circuit must be specified. For analog circuits, this typically includes minimum and maximum values for performances, e.g., maximum power consumptions or minimum gain for an amplifier, and environmental conditions, e.g., the temperature range or the required range for the supply voltage the circuit should be designed for. Additionally, the technology used for the system is defined at this point.

Afterward, a designer must decide for analog components if a known schematic can be reused or if a new schematic must be built for the required functionality. This is done in the topology generation step. To support the designer taking this decision, several approaches have been presented to estimate the capabilities of a circuit, e.g., by computing the Pareto front (e.g., [MGG10, MG09, TTR06]). However, despite several attempts to automate the generation of a circuit schematic (e.g., [KH06,SMF02,SKK03]), no general approach has been found as yet, which can be used in practice.

To decide if a circuit schematic is sufficient for the specified functionality, an initial sizing is required, i.e., initial values must be assigned to transistor lengths and widths, resistances, etc. However, the initial sizing typically does not fulfill all specifications.

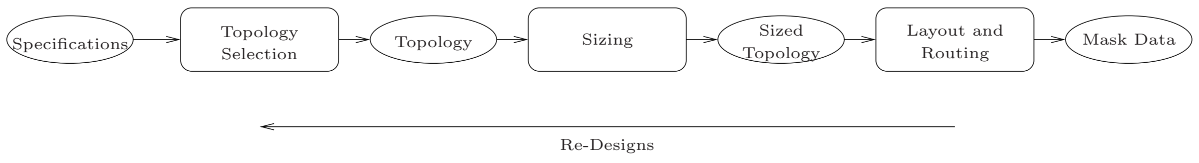


Figure 1.2: Visualization of the analog design flow

Therefore, after defining the structure which should be used for a new circuit, the parameters of the circuit must be sized. The resulting sizing should not only fulfill specifications but should also ensure that the circuit is robust against variations in the manufacturing process [GR00]. For this highly complex task, several commercial and non-commercial approaches have been developed which support the designer to compute such a sizing (cf. Chapter 1.3). However, these approaches are sparsely used in practice due to several open problems [ITR10].

After computing the sizing of an analog circuit, the layout and routing can be done. In this stage of the analog design flow, transistor geometries and positions are finalized and mask data are extracted which can be used to manufacture the circuit. For this task, a small number of approaches can be found to support the designer (e.g., [Str11, YD09, ZJBS08]).

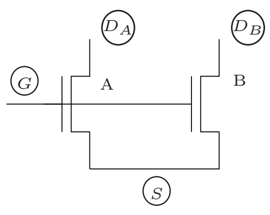
The complete analog design flow is shown in Figure 1.2. In each step of the flow, the circuit is evaluated and its functionality is verified using simulations. If the specifications of a circuit can not be fulfilled in a certain design step, a modification of the circuit in one of the previous steps is required. The cost for such a redesign typically increases if the problem is detected in one of the last steps. Therefore, it is necessary to consider as much information as possible in the first design steps, e.g., to consider information about the layout during the sizing step to develop a layout-friendly sizing, or to consider process variations during the sizing step to make the circuit robust.

## 1.2 Problem Description

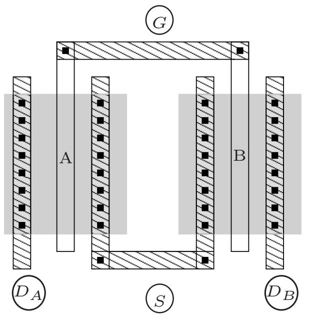
### 1.2.1 Technical Problem

Automating the task of analog sizing has been an important field of research in electronic design automation for many years. In automatic approaches, the parameter values of the devices in a circuit, e.g., transistor widths and lengths, should be derived under nominal operating and nominal process conditions as well as under consideration of different operating conditions and process variations. This thesis especially focuses on the enhancement of these approaches by considering discrete parameters.

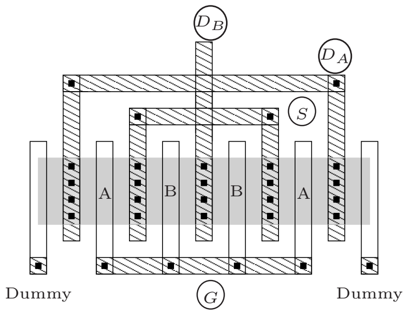
Discrete design parameters can be classified with respect to their origin. In the analog sizing task, discrete parameters appear due to:



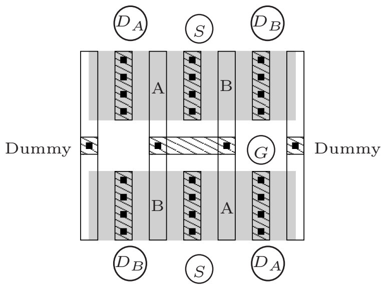
(a) Schematic for transistors A and B connected at gate and source



(b) Layout of transistors A and B as two single transistors



(c) Layout of transistors A and B enlarged by dummy transistors using multiple fingers in parallel [Has01]



(d) Layout of transistors A and B enlarged by dummy transistors as cross-coupled pair [Has01]

Figure 1.3: Schematic and layout for two transistors with equal lengths and total widths $w_{total}$

- Layout: In the traditional sizing step, parameter values, e.g., lengths and widths of transistors, are assigned. However, in the subsequent layout step the single elements from the sizing step are frequently split into multiple elements. Typical examples are the layout of transistors as multi-finger structures or as common centroid structures (Figure 1.3) to receive a more compact design and to increase matching. Transistors are also realized as multi-finger transistors to reduce the source bulk and the drain bulk capacitance $C_{SB}$ and $C_{DB}$, respectively. However, if a transistor with a total width $w_{total}$ is split into $m$ equal transistors in parallel with a finger width of $\frac{w_{total}}{m}$ each, the resulting device behaves differently from the actually designed transistor. Thus, the number of transistors connected in parallel should be considered in the analog sizing and not only in the layout step. The dependency of the transistor on the realization is visualized in Figure 1.4. The figure shows the drain current of a CMOS-Transistor in a 180$nm$ technology with a total width $w_{total}$ but different values for $m$, i.e., different values for the number of transistors connected in parallel. Further examples which show the effect of splitting transistors during the layout can be found in [KKC$^+$08] and [YKC$^+$05]. To model the number of transistors connected in parallel, e.g., BSIM models [BSI11]

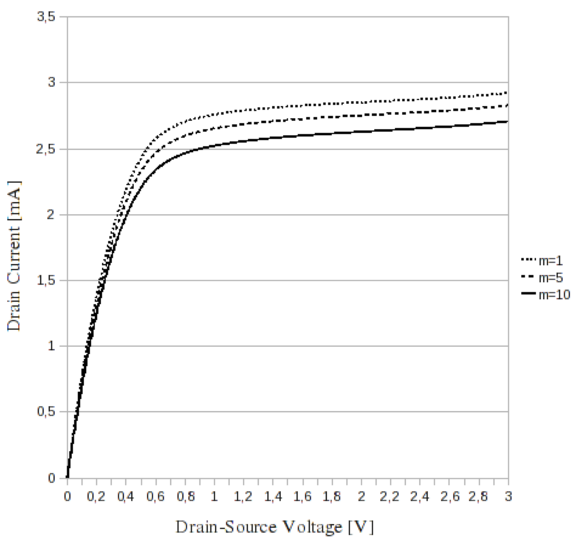


Figure 1.4: Simulation of the drain current over drain-source voltage for a NMOS-Transistor in a $180nm$ technology with a total width $w_{total} = 10\mu m$, gate source voltage $V_{GS} = 1.5V$, and finger numbers $m \in \{1, 5, 10\}$ using a BSIM3 model

provide a variable referred to as multiplier. The number of transistors connected in parallel is obviously integer, i.e., discrete.

Additionally, the edges of the devices of a circuit, e.g., the edges of the transistor gate, must lie on a manufacturing grid and the parameters of the devices, e.g., transistor lengths and widths, are discrete in some processes. This grid is typically in the range of several nanometers and the effect of snapping the parameters to the grid was negligible for previous technologies. However, with shrinking transistor sizes, the impact must be considered in the sizing step.

A further layout-specific discrete parameter appears if integrated inductors have to be realized. The number of winding turns in such devices can typically only be changed in discrete, e.g., eighth, quarter, or whole windings. This can be considered in the sizing step if appropriate models are provided.

The consideration of multipliers and parameter grid is also mentioned in [ITR10] as an open point for automated sizing of analog circuits.

- New technologies: In some new technologies, the parameters of the devices can only be scaled discretely. Considering, e.g., FinFETs (e.g., [LAG+03]), the width of the gate area controlled by a single fin can be computed as $2 \cdot h_{fin}$, where $h_{fin}$ is the height of the fin [SH06]. This is illustrated in Figure 1.5. However, the fin height should be equal throughout the device and, consequently, the gate width for a multi-gate FinFET can be determined by the (integral) number of fins. This

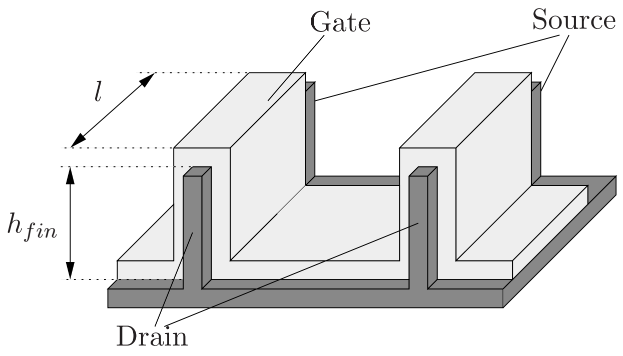


Figure 1.5: Illustration of a FinFET with two fins, fin height $h_{fin}$ and length $l$

integral number can only be considered as a discrete parameter in the circuit sizing step [NAL+04].

- Modeling: One important basis to compute the sizing of circuits is the existence of sufficiently accurate device models. However, in some cases the models are accurate only for certain discrete points or are piecewise continuously defined in practice. To consider this problem in an automatized design flow, the sizing task must be formulated as a discrete problem.

It can be seen that in practical circuit sizing approaches many discrete parameters may appear. However, today the automation of analog sizing considering discrete parameters has not yet been solved in a practicable way (cf. Chapter 1.3).

## 1.2.2 Mathematical Problem

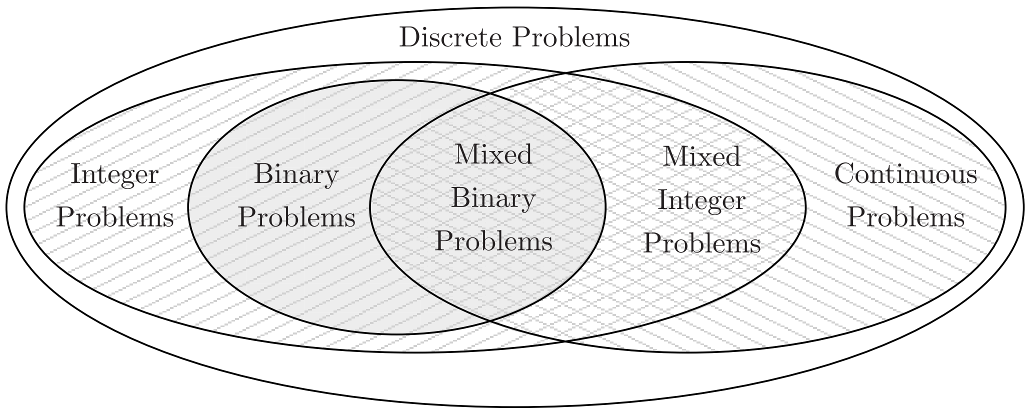


Figure 1.6: Optimization classes

The task of sizing an analog circuit is typically formulated as an optimization problem (cf. Chapter 2.5). These optimization problems can be subdivided into different classes, depending on the parameters used (Figure 1.6). The terminology for the different problem classes is not always unique. Within this thesis, the terms are used as follows:

- Continuous Problems are optimization problems, which must be solved on an continuous parameter domain, i.e., parameters can take each value between a lower bound and an upper bound. Such formulations were used, e.g., in many previous analog sizing tasks (cf. Chapter 1.3) where it was assumed that all circuit parameters could be scaled continuously.
- Integer Problems consider only integer values as variables in the optimization task. Compared to discrete problems the parameter values are always ordered and equidistant.
- Mixed-Integer Problems consider integer parameters, as well as continuous parameters. In circuit sizing, such problems arise if – besides continuous parameters – only parameters on an equidistant parameter grid or multipliers are used.
- Binary Problems or 0-1 problems are problems where parameters can only take the values *true* and *false*. Such problems are often used to model, e.g., knapsack or assignment problems [NW88]. It can be found that each optimization problem considering exclusively discrete or integer values can be converted into a binary problem.
- Mixed-Binary Problems consider binary variables and continuous parameters. Such problems arise, e.g., if piecewise linear functions are used in the optimization task or if disjunctive constraints arise in an optimization task.
- Discrete Problems are problems which can include continuous parameters, integer parameters, and additionally arbitrary discrete parameters. In contrast to integer parameters, discrete parameters are not necessarily equidistant. During analog sizing, such problems arise, e.g., if the parameter grid is not constant over the parameter space or if the model of a transistor is only valid for certain non-equidistant parameter points. Also non-numerical parameters to model switching between different technologies might appear and can be modeled as discrete parameters, but are not explicitly considered in this work. It is assumed in this thesis that the discrete values of each discrete parameter can be ordered by size and that this order has a physical meaning (cf. Chapter 2.1.1).

Due to the definition of the terms above, the analog sizing task considering parameters as described in Chapter 1.2.1 must be formulated as a discrete optimization problem. One difficulty that results from the formulation of the sizing task as a discrete optimization problem is that there is no optimality criterion like the *Karush-Kuhn-Tucker* conditions to check whether the solution found is optimum [LS06]. As a consequence, algorithms to solve the discrete sizing task typically compare different solution candidates, until no further improvement can be seen.

Additionally, it can be expected – although it has not been proved yet – that the discrete optimization task is an *NP-hard* problem and no algorithm can be found to solve the problem in polynomial-time complexity. The assumption is derived from the fact that *NP-hardness* has been proved for certain binary and integer problems [LS06] which are typically easier to solve than a general discrete optimization problem.

Despite the required discreteness of the solution, continuous optimization can be frequently performed on the analog sizing task to compute a continuous solution. It might be assumed that in these cases continuous optimization with subsequent rounding is an approach to solve the task. However, it can be shown that the solution of the discrete problem may be far away from the solution of the continuous optimization problem [LS06]. An example of this problem is shown in Appendix A.1. But also in case the discrete solution is one of the points next to the continuous solution, the solution can be one out of $2^{N_d}$ points, where $N_d$ is the number of discrete parameters. Therefore, a more sophisticated approach must be developed and is presented in this thesis, which solves the discrete sizing task within a practicable runtime.

## 1.3 State of the Art

### 1.3.1 Approaches for Automated Sizing

Unlike previous knowledge-based approaches, modern approaches formulate the task of sizing an analog circuit as an optimization problem (cf. Chapter 2.5) [GR00]. Iterative methods are used to solve this optimization task. The methods require algorithm parameters and – in some cases – initial solutions as inputs and compute one or multiple intermediate solutions in each iteration. These intermediate solutions must be evaluated using a description of the circuit. The results of the evaluation are used to improve the intermediate solutions until the sizing algorithm terminates and, e.g., a sizing has been found which fulfills specifications and constraints. This proceeding is presented in Figure 1.7. In this figure, the algorithm used to solve the optimization task is implemented in the *Optimization Engine*, the evaluation of the intermediate solutions takes place in the *Circuit Evaluation Engine*.

#### Symbolic vs Simulation-Based Approaches

Concerning the description used and the method to evaluate the description of an analog circuit during the circuit sizing process, approaches for analog sizing can be roughly subdivided into approaches which use a symbolic or formal description (called symbolic approaches in the following) and approaches which evaluate the circuit using simulations (referred to as simulation-based approaches).

Symbolic approaches require an explicit formulation of the circuit performances depending on the design parameters of the circuit. However, the explicit description of the performance models must be set up either manually or using automated model generation approaches (e.g., [DGS03, WFG$^+$95, YS96, TCF96]). In both cases, the description of the circuit properties is simplified and the resulting formulation may be inaccurate.

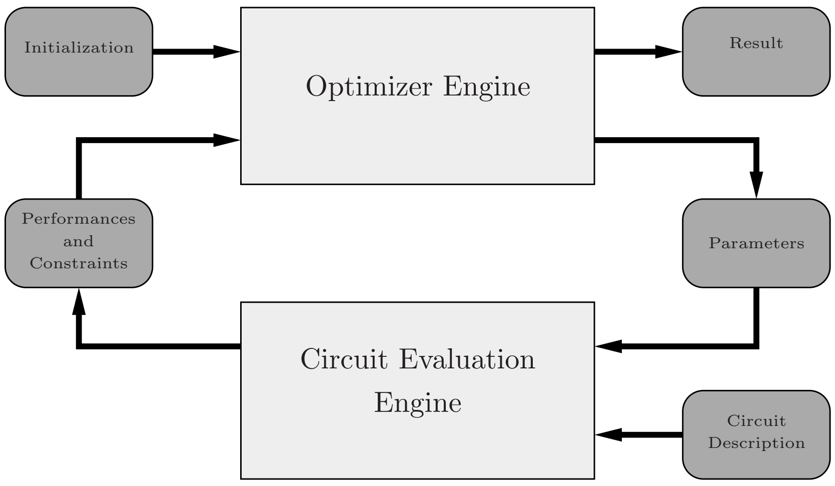


Figure 1.7: General structure of an approach for automated analog sizing

Therefore, a good symbolic description can typically only be given for near linear circuit properties and, e.g., not for transient characteristics [GR00]. Additionally, deriving a model is time-consuming and the resulting model is valid only for one circuit and for the parameter range considered during model construction. However, after generating the model, circuit behavior can be approximated very fast. Additionally, the sizing task can be formulated as a geometrical program (e.g., [BKVH07, MV01, dMH04]) if the model is formed by posynomial functions. In case of a formulation as a geometrical program, the problem has a unique solution and the global optimum of the problem can be found efficiently. However, due to the simplification in the model, the optimum computed on the model does not necessarily solve the original sizing task.

In contrast, simulation-based approaches evaluate the given circuit using SPICE-like circuit simulators [NKZB94]. Using such simulators, time consumption is relatively high for each circuit evaluation. However, the result is accurate and the sizing approach does not require the designer to provide a model for each new circuit. In this thesis a simulation-based approach is realized due to its higher accuracy and generality.

### Stochastic vs Deterministic Approaches

Depending on the algorithm implemented in the optimization engine, current sizing approaches can be subdivided into stochastic and deterministic approaches. Stochastic approaches use random numbers to determine intermediate solutions for an optimization problem. The most popular approaches in this field are the Simulated-Annealing approaches [KGV83], which imitate the cooling of fluids to solve an optimization problem, and evolutionary algorithms which try to imitate the Darwinian evolution. In contrast, deterministic approaches try to improve the intermediate solution based on explicit knowledge regarding the mathematical problem. Popular approaches in this field, e.g., Sequential Quadratic Programming (SQP) and the Conjugate Gradient method, use gradients and explicitly computed performance and constraint values to solve the problem iteratively.

Comparing stochastic and deterministic approaches, stochastic approaches typically have the potential to find a global optimal point, whereas deterministic approaches (without enhancement) often get stuck at the next local optimum. However, in stochastic approaches, a trade-off must be made between globality and convergence, and the number of function evaluations is typically much higher than the number of function evaluations required for deterministic approaches.

For most analog sizing tasks, a good initial solution can be provided by the designer. In these cases, it is sufficient to provide an algorithm which converges to the next local optimum solution. Additionally, the time required for the evaluation of the circuit properties strongly contributes to the runtime of the algorithm if a simulation-based approach is used, as it was decided for this thesis. Considering the typically much higher number of function evaluations for stochastic approaches, which often leads to an impracticable high runtime of the sizing algorithm, a deterministic approach is established in this work to solve the sizing task.

### 1.3.2 Discrete Sizing of Analog Circuits

Analog circuit sizing is currently mostly considered to be a continuous sizing approach. However, the following selection shows that stochastic approaches often have the capability to solve discrete problems.

In [GWS90] an approach is presented which solves the task of analog sizing without considering the operating region and process variations. The approach uses Simulated Annealing to solve the optimization task and symbolic analysis to evaluate the circuit. To run Simulated Annealing efficiently, the design variables are forced to lie on a grid, i.e., all design parameters are discrete. Consequently the result is always on an initially chosen grid and can not be a continuous solution in between. The discrete values in this work are assumed to be equidistant, although Simulated Annealing has the capability to solve problems with parameters on a non-uniform grid. The runtime of the algorithm in this work strongly depends on the parameter range and the user is required to limit this range properly. Due to the use of Simulated Annealing, the approach allows to find a near global optimum. However, the result is inaccurate due to the symbolic analysis used, which requires simplifying the formulation of the circuit properties.

[ORC96] also suggests a Simulated Annealing approach. The circuit properties in this approach are evaluated using accurate circuit models (referred to as encapsulated device evaluators). As a consequence, evaluation time is shorter and the result less accurate than in traditional simulation-based approaches. At the same time, evaluation time is higher and the result more accurate than in traditional symbolic approaches. The evaluation method used limits the approach to essentially linear performances. Due to the use of Simulated Annealing, the approach can be used for discrete problems, although they are not explicitly considered. Additionally, although focusing on computing a nominal design for the circuit, the authors mention that, in theory, the approach is able to to consider operation regions and process variations.

In contrast to the methods above, [PKR$^+$00] accepts the much higher runtime of a simulation-based evaluation of the circuit to gain a more accurate result. The algorithm combines an evolutionary approach with a pattern search strategy to find the global optimum. It is claimed that the algorithm can find the global solution and can consider process variations and operating conditions by constraints (cf. Chapter 2.2), which ensure that the transistors work in the envisaged operation region. Additionally, the approach is said to be able to consider discrete parameters, such as multipliers. But one of the key mechanisms in the algorithm is a coordinate search with randomized direction and step size. Since a coordinate search does not necessarily find the optimum or a near optimal point of a discrete problem, this might affect the convergence of the algorithm in a discrete sizing task.

In [ABD03] an evolutionary approach for the task of circuit sizing is presented which uses a Simulated Annealing modification in the selection step of the algorithm. The approach considers discrete design parameters as well as process variations. To consider process variations, neural-fuzzy performance models are used. However, the approach considers only DC- and AC-performances of the circuits which are simulated by a fast in-house simulator.

A large number of further approaches, such as [SCP07, DCR05, VSBM04, HC04, YWLF95] can be found, which use stochastic approaches partly in combination with deterministic approaches (e.g., [GDL$^+$95]). One advantage of most of these approaches is their capability to find a global optimum point. Additionally, these approaches typically have the theoretical capability to solve a discrete sizing task. However, stochastic approaches require a large number of function evaluations. Therefore, they are often too slow in practical applications if the circuit is evaluated by simulations. To overcome this problem, in many statistical approaches more abstract models are used to approximate the properties of the circuit. However, as the effort to generate a sufficiently good abstract model is high, methods using such abstract models are uncommon in practice.

Another idea to overcome the problem of the slow convergence of stochastic approaches for a discrete sizing task has been presented in [PMG08] and [PG09]. In these works, a simulation-based approach is presented which determines step direction and step size based on gradient information on the circuit performances and based on random numbers. The approach can guarantee to find intermediate solutions which are discrete. However, due to the use of a gradient direction, the approach might get stuck at suboptimal discrete points where the circuit specifications are not fulfilled, and Tabu-Search [GL97] is required to guarantee convergence. As a consequence, the convergence of the algorithm is slow in many cases.

Due to the high runtime of evolutionary approaches, continuous sizing approaches (e.g., [Sch02, SSGA00, CCH$^+$96]) are often used in practice and the result is rounded to a discrete solution. However, as discussed in Chapter 1.2.2, this often results in suboptimal discrete points and the specifications might not be fulfilled for the solution point.

In some cases mixed-integer approaches have already been used for problems related to discrete analog sizing:

In [SHA$^+$05] a standard Mixed-Integer Nonlinear Programming (MINLP) approach is used to consider multipliers for the gate sizing of digital circuits using FinFETs. In this work, only the power consumption of the circuit is optimized and the complexity of the approach is therefore much lower than in typical analog sizing tasks. The problem formulation in this work was given as a geometrical program [BKVH07], i.e., the formulation of performance and constraints was explicitly given using symbolic equations. However, the runtime of the MINLP-solver is stated to be very high and advanced rounding is used in later works to solve the task, accepting an inaccuracy of the result [SH06].

For analog synthesis, the approaches in [MCR92] and [MCR95] use integer nonlinear programming and MINLP, respectively. Discrete parameters are used in these approaches to model different structures which can be selected. Design parameters (cf. Chapter 2.1.1) are considered to be continuous. Hence, the cardinality of the discrete domain, i.e., the number of allowed values for each discrete design parameter, is low in these works. The problem is formulated as a geometrical problem, which requires simplified descriptions of the performances as functions of the design parameters.

Concluding, it can be found that the problems related to analog sizing and solved by MINLP are less complex than the task considered in this thesis. Evolutionary approaches are often slow in practice if simulations are used for circuit evaluation and too inaccurate if not. Current deterministic approaches which solve the continuous sizing problem and add subsequent rounding are fast but might end up with a solution that does not solve the sizing task even though a solution exists.

In commercial tools, deterministic and statistic approaches are realized (e.g., [WIC11, CAD11]). However, if discrete parameters are considered in these tools, the deterministic approaches use rounding or advanced rounding to discretize the final solution or each intermediate solution.

### 1.3.3 Tolerance Design

During the sizing of an analog circuit, the parameters of the circuit should be assigned so as to fulfill the performances for the complete operation region. Additionally, the performances should be robust against variations in the subsequent manufacturing process. Current approaches mostly solve this task using yield optimization (e.g., [Gra07,Sch04]). The approaches can be subdivided into different groups with respect to the used approximation method for the yield [LFG11].

The most common approaches to approximate the yield are Monte-Carlo analysis and yield analysis based on worst-case approximation [Gra07]. Although Monte-Carlo analysis is the most general and – using a sufficiently high number of samples – most accurate method, it is usually computationally too expensive to be used within an iterative algorithm for yield optimization. Therefore, in practice mostly methods based on worst-case approximation are used [AG, AEG$^{+}$00, DK95]. Other approaches use capability indexes [AS94] or linearized performance penalties [KD95] to approximate the yield.

In practice, it is sufficient in many cases to obtain a predefined required yield. Obviously, such a solution can be found by stopping a yield optimization as soon as the yield requirements are met. An alternative method is presented in [SSPG02] and used in [MG09]. In this approach, the required yield is initially specified and a successively refined approximation of the worst cases is used to ensure that the predefined yield requirements are finally fulfilled. The approach uses a gradient method combined with a trust-region approach for sizing.

[LFG11] also allows the direct consideration of a required yield in an optimization problem without optimizing over the yield. This approach uses a Monte Carlo analysis to approximate the yield and differential evolution to compute the sizing for the circuit.

## 1.4 Contribution of this Work

In previous works, the task of analog sizing is considered a discrete task only if a statistical approach has been used or if the circuit performances were formulated as symbolic functions. However, stochastic approaches are inefficient if a good initial solution can be provided, and the symbolic formulation of circuit performances requires simplifications which lead to inaccuracies. Additionally, symbolic formulations of the circuit performances are only valid for a single circuit and the construction of the symbolic description is time-consuming.

In contrast, this work solves the task of sizing an analog circuit with discrete parameters by two new deterministic approaches. Due to the higher generality and accuracy, the circuit properties are evaluated by simulations. The approaches are related to MINLP approaches. No previous approach uses MINLP or related approaches for analog sizing with simulation-based evaluation of the circuit performances. However, it can be expected that the runtime is much higher than for continuous sizing using deterministic approaches. To avoid an impracticable high runtime, two new, highly efficient approaches were developed in this thesis. Both new approaches use carefully adapted and enhanced Sequential Quadratic Programming and Branch-and-Bound algorithms.

For the first approach – referred to as Branch-and-Bound for analog sizing – it is assumed that simulation is possible for each continuous value in the design space. This is a common case in many practical problems, e.g., if the multiplier of a transistor which is described by a BSIM model [BSI11] should be considered as a design parameter. The assumption allows the computation of a continuous intermediate solution although the final result should be discrete. To find a discrete solution, an outer Branch-and-Bound approach is used to guarantee (local) convergence, and a new method is introduced to predict the discrete solution quickly. The outer Branch-and-Bound approach computes in each recursion a solution for the continuous sizing task using a simulation-based Sequential Quadratic Programming (SQP) approach. The size of the search space for the continuous solution is recursively reduced by rounding constraints. As a consequence, the continuous optimum in the search space is a discrete point after a finite number of recursions. For the prediction of a discrete solution, a linearly constrained quadratic model of the sizing task is used. Such a model is computed during the SQP approach. Solving the model over the discrete parameter domain, a discrete solution for the sizing task can be predicted quickly with a low number of additional simulations. The Branch-and-Bound approach is modified for the consideration of variations in process and operating conditions, which requires predominantly a modification of the SQP approach.

For the second approach – referred to as model-based analog sizing – it is assumed that the circuit cannot be evaluated for parameter points which are not in a predefined discrete set. This task is harder to solve because it requires the computation of a discrete solution in each step. To tackle this problem, a quadratic model for each performance and for each constraint is built up iteratively based on gradients. Afterward, a new discrete solution which fulfills the constraints and improves the circuit performances is computed on the model. Sequential Quadratic Programming and Branch-and-Bound are used to compute the discrete solution on the model.

To guarantee that each intermediate point in the optimization on the continuous domain is feasible, a Feasible Sequential Quadratic Programming [LT01] implementation is used as SQP approach. The approach is enhanced in this thesis by a gradient computation of the performances and the constraints and by a step size computation, both highly parallelized with respect to the required simulations.

Branch-and-Bound is used in the new Branch-and-Bound approach for analog sizing to compute a discrete sizing based on simulations and on a quadratic model of the sizing task to predict the discrete solution of the task efficiently. For the new model-based analog sizing approach, Branch-and-Bound is used to find improved intermediate solutions on a nonlinear model. The Branch-and-Bound approach on a simulation-based level must be focused predominantly on the computation of a solution for the sizing task, using a low number of computationally expensive simulations. In contrast, the Branch-and-Bound approach used to solve the quadratic model in the Branch-and-Bound approach for analog sizing and used to solve the nonlinear model in the model-based analog sizing approach should compute a close to optimal solution of the model efficiently but can use a higher number of function evaluations to achieve this goal. To consider the different requirements for the applications, two new Branch-and-Bound algorithms are presented which differ in the used branching heuristics and pruning conditions.

The computational complexity of a Branch-and-Bound algorithm is exponential in the worst case [TDG09]. Therefore, the typical assumption that the runtime of the algorithm is dominated by the time required for simulations might not be true anymore if the quadratic and the nonlinear model in the new approaches are solved by Branch-and-

Bound. A trade-off between the number of simulations and the time required by other numerical operations is presented in this thesis to consider the potentially high runtime of Branch-and-Bound as well as the simulation time.

For the two new algorithms, different formulations with respect to the objective functions are useful. The selection of these functions is motivated and explained in this work.

The analog sizing task in this thesis is solved for nominal operating and nominal process conditions and considering different operating conditions and process variations. The consideration of different operating conditions and process variations allows the design of a robust circuit but requires more simulations and a higher runtime than the design under nominal conditions. Thus, the design under nominal conditions is used to compute a good initial solution for determining a robust design. For the computation of a robust design an approach similar to [SSPG02] is used in this thesis. However, the approach was adapted to Feasible Sequential Quadratic Programming and Branch-and-Bound, and was enhanced for the application in the new model-based analog sizing approach.

The two new approaches presented in this thesis are verified and evaluated by simulation-based experiments on four different circuits. In the experiments a nominal sizing as well as a tolerance sizing is computed for each of the circuits. The quality of the tolerance sizing was evaluated by Monte Carlo analyses.

## 1.5 Previous Publications

One of the algorithms presented in this thesis deals with an analog sizing problem under the assumption that the circuit can be evaluated at each continuous point. A previous version of the method presented in this thesis, which was developed for sizing under nominal operating and nominal process conditions, has been published in [PZG10] and [PG11]. The enlarged approach for sizing under consideration of tolerances is published in [PZG11].

Parts of the problem formulation in this thesis date back to an earlier approach and have been published in [PMG08] and [PG09].

## 1.6 Structure of this Thesis

In Chapter 2, the task of analog sizing in consideration of discrete parameters and variations in operation and process conditions is formalized and defined as an optimization task. The optimization methods which are the basis for the new approaches in this thesis are introduced and discussed in Chapter 3 and new enhancements are shown which are required for analog sizing. The two new approaches for analog sizing are introduced in Chapters 4 and 5 and verified and evaluated by simulation-based experiments in Chapter 6. Chapter 7 concludes.

# Chapter 2

# Analog Sizing Task

Starting from the definition of design parameters, operating parameters, and process parameters (Section 2.1), this chapter gives a mathematical formulation of the analog sizing task as an optimization problem. For this purpose, sizing rules [Eck98, Ziz01, MGS08, Mas10] for an analog circuit are formulated as constraints for the optimization task (Section 2.2). Additionally, the performances of a circuit are defined and a *performance-to-specification gap* is introduced as the normalized distance from a performance value to the corresponding performance specification bound (Section 2.3).

In a typical analog sizing task, many counteracting circuit properties must be considered at the same time. As a result, the task of sizing an analog circuit is a multi-objective task in general. However, most optimization algorithms require a scalar formulation of the optimization problem. For this purpose, a mapping from the multi-objective sizing task to a scalar is shown in Section 2.4. This mapping is used in Section 2.5 to formulate the design task for different design objectives.

## 2.1 Circuit Parameters

The circuit parameters described in this section are subdivided into design parameters $\mathbf{d}$, which can be influenced by the designer, operating parameters $\mathbf{o}$, which are part of the circuit specification, and process parameters $\mathbf{s}$, which are used to model process variations. The three types of parameters are collected in an $N_p$-dimensional vector $\mathbf{p}$

$$\mathbf{p} = \begin{bmatrix} \mathbf{d} \\ \mathbf{o} \\ \mathbf{s} \end{bmatrix} \tag{2.1}$$

## 2.1.1 Design parameters

During the sizing step in the analog design flow, the parameter values of the devices, e.g., transistor widths and lengths, must be assigned so as to fulfill certain circuit properties. The parameters which can be influenced by the designer are referred to as *design parameters* or *tuning parameters* and labeled by $\mathbf{d}$. The $N$ design parameters of a circuit can be subdivided into $N_c$ continuous parameters, e.g., transistor lengths and widths without considering a manufacturing grid, and $N_d$ discrete parameters (cf. Chapter 1.2.1). Each $N$-dimensional parameter point consisting of discrete design parameters $\mathbf{d}_d$ and continuous design parameters $\mathbf{d}_c$ can be written as

$$\mathbf{d} = \begin{bmatrix} \mathbf{d}_d \\ \mathbf{d}_c \end{bmatrix} \in \mathbb{D}^N = \mathbb{D}_d^{N_d} \times \mathbb{D}_c^{N_c} \tag{2.2}$$

$\mathbb{D}^N$ is referred to as the *design space*. $\mathbb{D}_d^{N_d}$ and $\mathbb{D}_c^{N_c}$ are the design space for the discrete and for the continuous parameters, respectively. Without loss of generality, it can be assumed that the first $N_d$ parameters are discrete.

The design space for continuous parameters $\mathbf{d}_c \in \mathbb{D}_c^{N_c}$ can be defined by lower bounds $\mathbf{d}_{c,l}$ and upper bounds $\mathbf{d}_{c,u}$:

$$\mathbf{d}_c \in \mathbb{D}_c^{N_c} = \left\{ \mathbf{d} \in \mathbb{R}^{N_c} \,|\mathbf{d}_{c,l} \le \mathbf{d} \le \mathbf{d}_{c,u} \right\} \tag{2.3}$$

In contrast, to define the design space for discrete parameters $\mathbf{d}_d \in \mathbb{D}_d^{N_d}$, it must be considered that each discrete parameter $d_{d,i}$ is an element of a discrete domain $\mathbb{D}_{d,i}$ with cardinality $\left|\mathbb{D}_{d,i}\right| = n_i$:

$$d_{d,i} \in \mathbb{D}_{d,i} = \{d_1, ..., d_{n_i}\} \tag{2.4}$$

The design space for $N_d$ discrete parameters can be defined as

$$\mathbf{d}_d \in \mathbb{D}_d^{N_d} = \bigtimes_{i=1}^{N_d} \mathbb{D}_{d,i} \tag{2.5}$$

According to Chapter 1.2.1, the discrete design parameters can be ordered with respect to their physical meaning. Thus, the design space $\mathbb{D}_{d,i}$ for each discrete parameter $d_{d,i}$ is a strictly totally ordered set

$$\mathbb{D}_{d,i} := (\mathbb{D}_{d,i}, <) \tag{2.6}$$

where the relation $<$ is (cf. [PD00])

1. irreflexive: $d_{d,i} \in \mathbb{D}_{d,i} \Rightarrow \neg\left(d_{d,i} < d_{d,i}\right)$
2. asymmetric: $d_{d,i} < d_{d,j} \Rightarrow \neg\left(d_{d,j} < d_{d,i}\right)$
3. transitive: $\left(d_{d,i} < d_{d,j} \wedge d_{d,j} < d_{d,k}\right) \Rightarrow \left(d_{d,i} < d_{d,k}\right)$
4. connex: $d_{d,i} \neq d_{d,j} \Rightarrow \left(d_{d,i} < d_{d,j} \vee d_{d,j} < d_{d,i}\right)$

As a consequence, for each discrete parameter set $\mathbb{D}_{d,i}$ in (2.4) an index set $\mathbb{I}_i$

$$\mathbb{I}_i = \{1, ..., n_i\} \tag{2.7}$$

can be defined such that

$$\underset{j,k \in \mathbb{I}_i}{\forall} \quad d_{d,j} \in \mathbb{D}_{d,i} \wedge d_{d,k} \in \mathbb{D}_{d,i} \wedge (j < k \Leftrightarrow d_{d,j} < d_{d,k}) \tag{2.8}$$

In addition, a lower bound $d_{i,l}$ and an upper bound $d_{i,u}$ can be defined for each discrete parameter:

$$d_{i,l} = d_1 = \min \mathbb{D}_{d,i} \tag{2.9}$$

$$d_{i,u} = d_{n_i} = \max \mathbb{D}_{d,i} \tag{2.10}$$

The lower and upper bounds for continuous and discrete parameters can be collected in the vectors $\mathbf{d}_L$ and $\mathbf{d}_U$, respectively, i.e.,

$$\mathbf{d}_L = \begin{bmatrix} \mathbf{d}_{d,l} \\ \mathbf{d}_{c,l} \end{bmatrix}; \text{ with } \mathbf{d}_{d,l} = [d_{1,l}, ..., d_{N_d,l}]^T \tag{2.11}$$

$$\mathbf{d}_U = \begin{bmatrix} \mathbf{d}_{d,u} \\ \mathbf{d}_{c,u} \end{bmatrix}; \text{ with } \mathbf{d}_{d,u} = [d_{1,u}, ..., d_{N_d,u}]^T \tag{2.12}$$

As finite upper bounds $d_{i,u}$ and lower bounds $d_{i,l}$ for the parameters are given in an analog sizing task, e.g., by physical limitations or by the range where a device model is valid, each parameter can be normalized by

$$\widetilde{d_i} := \frac{d_i - d_{i,l}}{d_{i,u} - d_{i,l}} \tag{2.13}$$

This normalization is essential for the comparability of the parameters and for the numerical robustness of the optimization algorithms. Hence, it is assumed for the remainder of this thesis that all parameters are normalized although their normalization is not explicitly labeled for the sake of a simple description.

For many discrete optimization algorithms, a relaxation of the discrete design space is required, i.e., for discrete parameters also continuous intermediate values must be allowed (see Figure 2.1 for an example with one continuous and one discrete design parameter). The design space is a discrete parameter domain $\mathbb{D}^N$ (2.2) which is composed of continuous and discrete dimensions, and which is bounded according to (2.11), (2.12). The relaxation $\mathbb{D}^N_{\text{rel}}$ of this discrete parameter domain $\mathbb{D}^N$ can be defined as

$$\mathbb{D}^N_{\text{rel}} = \{\mathbf{d} \,|\mathbf{d}_L \leq \mathbf{d} \leq \mathbf{d}_U\} \tag{2.14}$$

I.e., the domain for each continuous parameter is left unchanged while for discrete parameters the parameter domain is enlarged to continuous values between the lower

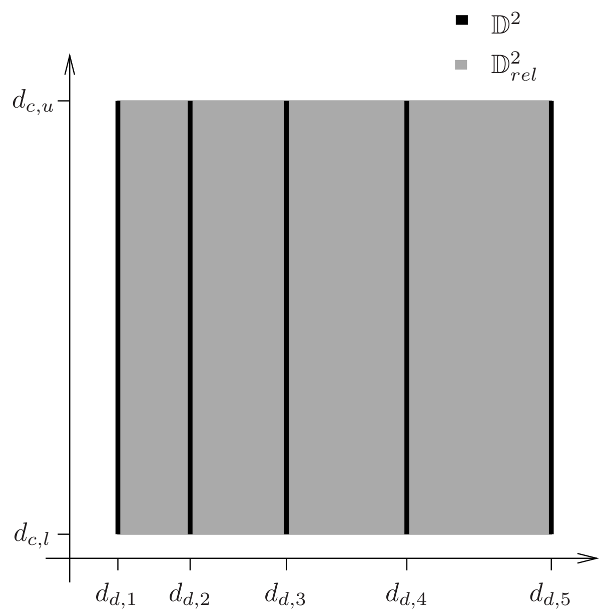


Figure 2.1: Design space $\mathbb{D}^2$ and its relaxation $\mathbb{D}^2_{rel}$ for one continuous and one discrete parameter.

and the upper bound for the parameter. As a consequence, the discrete parameter domain is always a subset of its relaxation and rounding operators can be used to get from a point in the relaxed domain back to a discrete parameter point. To round a single parameter value $d_i$ in the relaxed domain with $i = 1, ..., N_d$ to a value in the non-uniform discrete domain, the operators $\lceil \bullet \rceil$ and $\lfloor \bullet \rfloor$ are used:

$$\lceil d_i \rceil = \min_{d_k \in \mathbb{D}_{d,i}} d_k \text{ s.t. } d_k \geq d_i; \quad \lfloor d_i \rfloor = \max_{d_k \in \mathbb{D}_{d,i}} d_k \text{ s.t. } d_k \leq d_i \tag{2.15}$$

Accordingly, to round a single component of a parameter vector

$$\mathbf{d} = [d_1, ..., d_i, ..., d_N]^T \in \mathbb{D}^N_{\text{rel}} \tag{2.16}$$

in the relaxed domain to a value in the discrete domain, the operators $\lceil \bullet \rceil_i$ and $\lfloor \bullet \rfloor_i$ with $i = 1, ..., N_d$ are defined

$$\lceil \mathbf{d} \rceil_i = [d_1, ..., d_{i-1}, \lceil d_i \rceil, d_{i+1}, ..., d_N]^T \tag{2.17}$$

$$\lfloor \mathbf{d} \rfloor_i = [d_1, ..., d_{i-1}, \lfloor d_i \rfloor, d_{i+1}, ..., d_N]^T \tag{2.18}$$

The operator to round a parameter in the relaxed domain to the next point in the discrete domain is defined by

$$\lceil \mathbf{d} \rfloor = [\lceil d_1 \rfloor, ..., \lceil d_{N_d} \rfloor, d_{N_d+1}, ..., d_{N_d+N_c}]^T \tag{2.19}$$

with

$$\lceil d_i \rfloor = \arg \min_{d_k \in \mathbb{D}_{d,i}} |d_k - d_i| ; \, i \in \{1, ..., N_d\} \tag{2.20}$$

The different rounding operations are visualized in Figure 2.2.

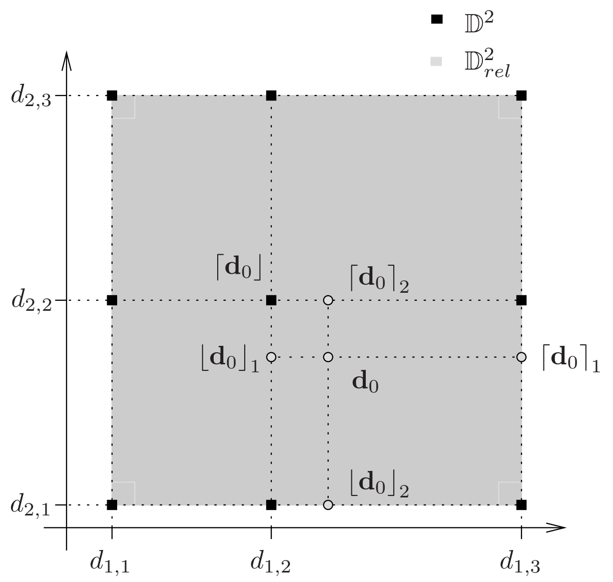

■ $\mathbb{D}^2$
■ $\mathbb{D}^2_{rel}$
$d_{2,3}$
$d_{2,2}$
$d_{2,1}$
$d_{1,1}$
$d_{1,2}$
$d_{1,3}$
$\lceil \mathbf{d}_0 \rfloor$
$\lceil \mathbf{d}_0 \rceil_2$
$\lfloor \mathbf{d}_0 \rfloor_1$
$\lceil \mathbf{d}_0 \rceil_1$
$\mathbf{d}_0$
$\lfloor \mathbf{d}_0 \rfloor_2$

Figure 2.2: Rounding operators applied to a parameter point $\mathbf{d}_0$ in the relaxed domain $\mathbb{D}^2_{rel}$ for a design space $\mathbb{D}^2$ with two discrete parameters.

### 2.1.2 Operating Parameters

The functionality of a circuit must be guaranteed for different operating conditions, e.g., for temperatures from $-55\ °C$ to $125\ °C$ or for variations in the supply voltage. During analog sizing, *operating* or *range parameters* $\mathbf{o}$ are used to model the dependency of a circuit from operating conditions.

The operating region where the functionality of the circuit must be ensured is part of the specification and given by lower bounds $\mathbf{o}_l$ and upper bounds $\mathbf{o}_u$ for each parameter. With these bounds, the operating region can be defined as the $N_o$ dimensional domain

$$\mathbf{o} \in \mathbb{T}_o^{N_o} = \{\mathbf{o}\,|\mathbf{o}_l \leq \mathbf{o} \leq \mathbf{o}_u\} \tag{2.21}$$

The operating parameters can be normalized according to (2.13).

Operating conditions are typically defined in a continuous space and consequently modeled by continuous variables in this thesis. However, if a discrete or piecewise continuous specification of an operating condition is given, it can be relaxed (cf. Section 2.1.1). As the discrete domain is a subset of its relaxation, it is sufficient to guarantee the functionality of the circuit in the relaxed domain. Only if the circuit cannot be simulated for each operating condition in a relaxed domain, the sizing task must be considered to be discrete with respect to the operating parameters. However, it can be seen in Chapters 4.4 and 5.4 that – under mild assumptions – the algorithm in Chapter 4.4 may be applied to problems with discrete operating parameters without further modifications and that the approach in 5.4 requires only slight modifications to consider discrete operating parameters.

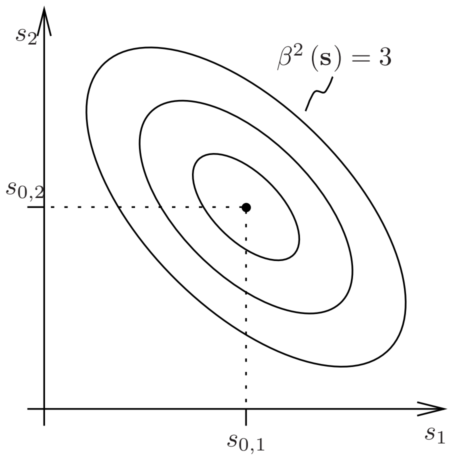


Figure 2.3: Contour lines of a Gaussian normal distribution with mean $[s_{0,1}, s_{0,2}]^T$ for $\beta^2(\mathbf{s}) \in \{1, 2, 3\}$ (as an example the value of $\beta^2(\mathbf{s}) = 3$ is labeled)

### 2.1.3 Process Parameters

The influence of variations in the manufacturing process of a circuit increases with shrinking technology size. Variations can be considered in the sizing step by $N_s$ *process* or *statistical parameters* $\mathbf{s}$:

$$\mathbf{s} \in \mathbb{R}^{N_s} \tag{2.22}$$

Process variations can be modeled by random variables [PDML94] which can be arbitrarily distributed. However, it is assumed in this thesis that the random variables $\mathbf{s}$ are Gaussian normally distributed $\mathbf{s} \sim \mathcal{N}(\mathbf{s}_0, \mathbf{C})$ with mean value $\mathbf{s}_0$ and variance-covariance matrix $\mathbf{C}$. I.e., the probability density function for the process parameters is defined as

$$\mathrm{pdf}(\mathbf{s}) = \frac{1}{\sqrt{2\pi}^{N_s}\sqrt{\det(\mathbf{C})}} \cdot \exp\left(-\frac{1}{2}\beta^2(\mathbf{s})\right) \tag{2.23}$$

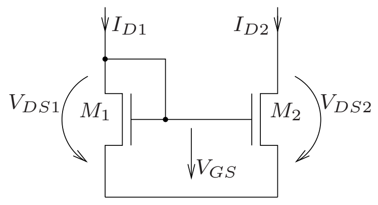
with

$$\beta^2(\mathbf{s}) = (\mathbf{s} - \mathbf{s}_0)^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{s} - \mathbf{s}_0) \tag{2.24}$$

The assumption of a Gaussian normal distribution of $\mathbf{s}$ is valid because the considered process parameters are either Gaussian or can be transformed into a Gaussian normal distribution [Esh92].

The value of $\beta^2(\mathbf{s})$ in (2.23) and (2.24) corresponds to the contour lines of the Gaussian distribution in the two-dimensional case (see Figure 2.3) and to the surface of a hyper-ellipsoid in higher dimensional cases. It can be noted that the value of $\beta^2(\mathbf{s})$ also corresponds to a certain yield (cf. Section 2.5.4).

Within this thesis, the process parameter vector $\mathbf{s}$ is assumed to be element of an $N_s$-dimensional continuous real valued domain. This assumption is valid if the device models can be simulated for each continuous process parameter point (cf. Section 2.1.2).



| | Geometric Sizing Rules | Electric Sizing Rules |
|---|---|---|
| Sizing Rules for Functionality | $l_1 = l_2$ | $\lvert V_{DS1} - V_{DS2}\rvert \leq \Delta V_{DS,max}$<br>$V_{DS1/2} - (V_{GS1/2} - V_{th}) \geq V_{sat,min}$<br>$V_{DS1/2} \geq 0$<br>$(V_{GS1/2} - V_{th}) \geq 0$ |
| Sizing Rules for Robustness | $w \cdot l \geq A_{min}$<br>$w \geq W_{min}$<br>$l \geq L_{min}$ | $(V_{GS1/2} - V_{th}) \geq V_{GS,min}$ |

Figure 2.4: Sizing rules for a simple current mirror [Mas10] with minimum area $A_{min}$, minimum width $w_{min}$, and minimum length $l_{min}$, threshold voltage $V_{th}$, minimum saturation voltage $V_{sat,min}$ and maximum drain source voltage offset $V_{DS,max}$

However, discrete models, e.g., device corner models [CM10], are used in some design scenarios to model the process conditions. To consider such discrete models, the process conditions must be treated like operating conditions (cf. Section 2.1.2). Thus, the performance specifications must be fulfilled for each recombination of the discrete process parameters and no minimum yield requirements (cf. Section 2.5.4) are supported.

## 2.2 Sizing Rules and Constraints

Given the schematic of an analog circuit, a set of rules can be defined which must be fulfilled to ensure the robustness of a circuit and to avoid its degeneration. These *sizing rules* [Eck98,Ziz01,MGS08,Mas10] describe relations between the geometries or between the electrical properties of devices or groups of devices in a circuit. As an example, the set of sizing rules for a simple current mirror is shown in Figure 2.4.

Sizing rules contribute to the efficiency of an automatic design tool in two ways:

1. Parameter reduction: The set of sizing rules often contains equalities which force one design parameter to be equal to a fixed multiple of another parameter. In this case, one of the parameters is redundant and can be substituted by the other. Therefore, the number of independent parameters and, as a consequence, the complexity of the sizing task is reduced.

2. Design space reduction: Other sizing rules can be formulated as inequality constraints $\mathbf{c}(\mathbf{p})$:

$$\mathbf{c}(\mathbf{p}) \geq \mathbf{0} \tag{2.25}$$

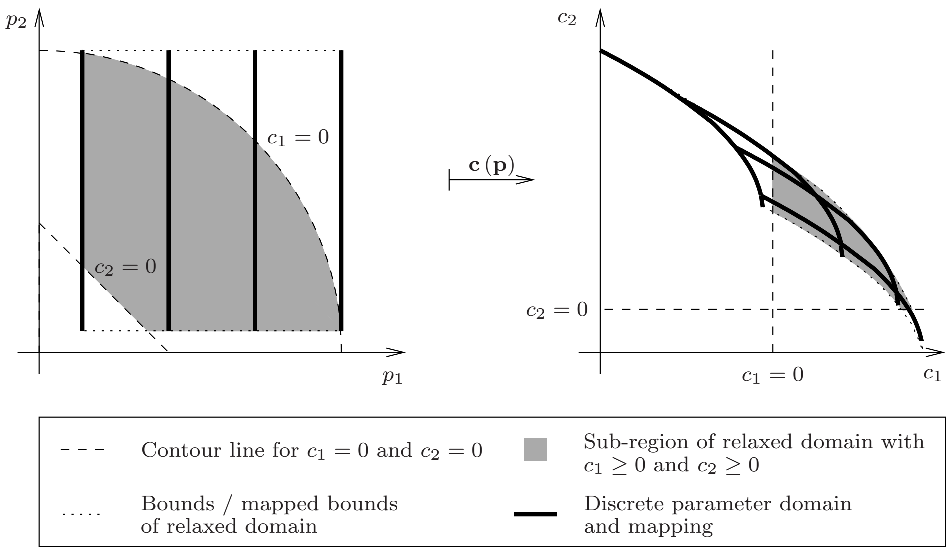

$p_2$
$c_1 = 0$
$c_2 = 0$
$p_1$
$\mathbf{c}(\mathbf{p})$
$c_2$
$c_2 = 0$
$c_1 = 0$
$c_1$
Contour line for $c_1 = 0$ and $c_2 = 0$
Sub-region of relaxed domain with $c_1 \geq 0$ and $c_2 \geq 0$
Bounds / mapped bounds of relaxed domain
Discrete parameter domain and mapping

Figure 2.5: Constraints, discrete design space, and relaxed design space in the parameter domain (left) and in the constraint domain (right)

$\mathbf{c}(\mathbf{p})$ is a surjective mapping from the parameter domain to the constraint domain (see Figure 2.5):

$$\mathbf{c} : \mathbf{p} \mapsto \mathbf{c}(\mathbf{p}) \tag{2.26}$$

As device geometries are typically defined as design parameters (cf. Section 2.1.1), the mapping for geometrical properties can be derived explicitly. In contrast, the mapping for electrical constraints – in a simulation-based approach – must be computed using simulations.

Considering inequality constraints in an automatic sizing approach can prevent the approach from entering unrewarding regions in the design space. Hence, fulfilling the constraints during the whole sizing process can be interpreted as cutting off non-promising parts from the design space.

Considering the requirements for the layout in the discrete sizing problem, the sizing rules presented in [Mas10] can be enlarged with respect to the transistor geometries [Has01]: Different lengths and widths of transistors strongly influence the matching of the transistors. As a consequence, each transistor of a matched transistor pair should have the same length and width. Additionally, all widths and lengths in a multi-finger structure must be equal.

This claim can be fulfilled by setting the lengths and widths of matched transistors equal during analog sizing. Consequently, the relation between the devices can only be influenced by multipliers (cf. Chapter 1.2.1). Considering this rule leads to a reduction of free parameters in the discrete sizing task.

Besides sizing rules, other circuit properties, e.g., a maximum area or a maximum power consumption, can be formulated as constraints for the sizing task. Within this thesis, such circuit properties are considered as performances.

## 2.3 Circuit Performances

Circuit performances $\mathbf{f}(\mathbf{p})$ refer to circuit properties which specify the required application-specific characteristics of the circuit. Typical examples are the gain, phase margin, or slew rate of an amplifier or the power consumption and area of a circuit. In this thesis, circuit performances are evaluated by simulations which perform a surjective mapping from the parameter domain into the performance domain analogous to the mapping presented for the constraints (cf. Figure 2.5):

$$\mathbf{f} : \mathbf{p} \mapsto \mathbf{f}(\mathbf{p}) \tag{2.27}$$

### 2.3.1 Performance Specifications

The requirements for each performance $f_i(\mathbf{p})$ of a circuit can be specified by upper and lower performance specification bounds $f_{u,i}$ and $f_{l,i}$, respectively. In the following it is assumed that for each performance $f_i(\mathbf{p})$ exactly one upper bound $f_i(\mathbf{p}) \leq f_{u,i}$ is defined. This assumption is valid because

1. a performance which is specified by upper and lower bounds can be split into two performances with one upper or lower bound each:

$$f_{l,i} \leq f_i(\mathbf{p}) \leq f_{u,i} \Leftrightarrow (f_{l,i} \leq f_i(\mathbf{p})) \wedge (f_i(\mathbf{p}) \leq f_{u,i}) \tag{2.28}$$

   I.e., the performance $f_i(\mathbf{p})$ appears twice in the performance vector $\mathbf{f}(\mathbf{p})$ if this transformation is applied: once as performance specified by the lower bound $f_{l,i}$ and once as performance specified by the upper bound $f_{u,i}$.

2. lower bounds can be transformed into equivalent upper bounds

$$f_{l,i} \leq f_i(\mathbf{p}) \Leftrightarrow \hat{f}_{u,i} := -f_{l,i} \geq -f_i(\mathbf{p}) =: \hat{f}_i \tag{2.29}$$

### 2.3.2 Sensitivity of Performances

For gradient-based optimization approaches, the sensitivities of the performances with respect to changes in the parameters are required. Some simulators for analog circuits can compute the sensitivities of performances directly if sufficient device models are provided. However, to be independent of the simulator and of the model, the central form of a finite differences approach is used within this thesis. The approach is enlarged below to capture discrete parameters. It is assumed that design and operating parameters are normalized according to (2.13).

The central form of the finite differences approach is used in this thesis to compute a highly accurate approximation for the gradient. However, the number of simulations required for the central form of a finite differences approach is twice the number of simulations required for the less accurate forward or backward form.

Using the central form of finite differences, the gradient or sensitivity $j_{k,i}$ of a performance $f_k$ with respect to a parameter $p_i$ is approximated by:

$$j_{k,i} = \frac{\partial f_k}{\partial p_i} \approx \frac{f_k(p_i + \Delta p_i) - f_k(p_i - \Delta p_i)}{2 \cdot \Delta p_i} \tag{2.30}$$

$\Delta p_i$ refers to a finite positive deflection for design parameter $p_i$. $f_k(p_i + \Delta p_i)$, and $f_k(p_i - \Delta p_i)$ refers to the change of performance $f_k$ if parameter $p_i$ is deflected and all other parameters are kept constant. The finite difference for $\Delta p_i \to 0$ is by definition the partial derivative. However, if $f_k(p_i + \Delta p_i)$ and $f_k(p_i - \Delta p_i)$ are computed by simulations, the value of $\Delta p_i$ must be large enough to reduce the effects of numerical inaccuracies in the simulation. In addition, $\Delta p_i$ must be small enough to ensure that the local approximation of sensitivity is good enough. Within this thesis, a fixed value of 0.1% of the parameter range has been found experimentally as a good trade-off and is used for $\Delta p_i$. However, this value can be directly influenced by the user of the developed tool.

It is assumed in this thesis that discrete parameters are ordered with respect to their physical meaning. Thus, it can be supposed that for the relaxation of the discrete domain (2.14) the performances $\mathbf{f}\left(\mathbf{p}\right)$ are differentiable. It can be followed that also for each point in the discrete domain a gradient can be defined. However, the approximation of the gradient must consider that in a discrete problem the points where an evaluation of the performances is possible can lie on a non-uniform grid, and – in some cases – no intermediate points can be evaluated.

To overcome this problem, each performance $f_k$ at a point $\mathbf{p}$ is approximated in direction of parameter $p_i$ by a quadratic function

$$f_k(p_i + \Delta p_i) \approx a_k \cdot \Delta p_i^2 + b_k \cdot \Delta p_i + c_k \tag{2.31}$$

The values of $a_k$, $b_k$, and $c_k$ can be calculated if the performance is evaluated for three values of $\Delta p_i$ (cf. Appendix B.1). In this thesis the values are chosen as $\Delta p_{i,0} = 0$, $\Delta p_{i,1} > 0$, and $\Delta p_{i,2} < 0$, where $\Delta p_{i,0} = 0$ corresponds to an evaluation at the point $\mathbf{p}$ where the gradient should be computed. With this assumption, the sensitivity $j_{k,i}$ of performance $k$ against variations in parameter $p_i$ can be approximated by

$$\begin{aligned} j_{k,i} = \tfrac{\partial f_k}{\partial p_i} \approx b_k = \\ = \tfrac{\Delta p_{i,2}}{\Delta p_{i,2} - \Delta p_{i,1}} \cdot \tfrac{f_k(p_i + \Delta p_{i,1}) - f_k(p_i)}{\Delta p_{i,1}} + \tfrac{\Delta p_{i,1}}{\Delta p_{i,1} - \Delta p_{i,2}} \cdot \tfrac{f_k(p_i + \Delta p_{i,2}) - f_k(p_i)}{\Delta p_{i,2}} \end{aligned} \tag{2.32}$$

It can be seen that this formula to compute the gradient approximation is equal to the central form for finite differences if $\Delta p_{i,1} = -\Delta p_{i,2}$, i.e., if the deflection of the parameter is symmetric around $\Delta p_{i,0} = 0$. However, for discrete parameters which can only be evaluated on a non-uniform grid, $\Delta p_{i,1}$ must be chosen as $\Delta p_{i,1} \neq -\Delta p_{i,2}$ to ensure that the performances can be computed.

In addition to the approximated sensitivity given by $b_k$, the value of $a_k$ – and thus an approximation of the curvature of the performance function in direction of parameter $p_i$ – can be computed. This is used in Chapter 5.

Collecting the sensitivities of a performance $f_k$ in a sensitivity vector

$$\mathbf{j}_k = \left[j_{k,1}, ..., j_{k,N_p}\right]^T \tag{2.33}$$

the *Jacobian matrix* $\mathbf{J}_f$ for $N_f$ performances can be defined:

$$\mathbf{J}_f = \left[\mathbf{j}_1, ..., \mathbf{j}_{N_f}\right]^T \tag{2.34}$$

Analogously, a Jacobian matrix $\mathbf{J}_c$ for the constraints in Chapter 2.2 can be defined.

## 2.3.3 Performance Normalization

To support tools for automatic sizing and to avoid numerical difficulties, the values of the performances must be comparable. This requires a normalization of the performances. For an ideal normalization, the minimum and maximum possible performance values $f_{max,i}$ and $f_{min,i}$ are used and the normalized performance can be computed by:

$$\tilde{f}\left(\mathbf{p}\right) = \frac{f_i\left(\mathbf{p}\right) - f_{min,i}}{f_{max,i} - f_{min,i}} \tag{2.35}$$

This type of normalization is used for the parameters in (2.13). However, the maximum and minimum values are not available for performances in general.

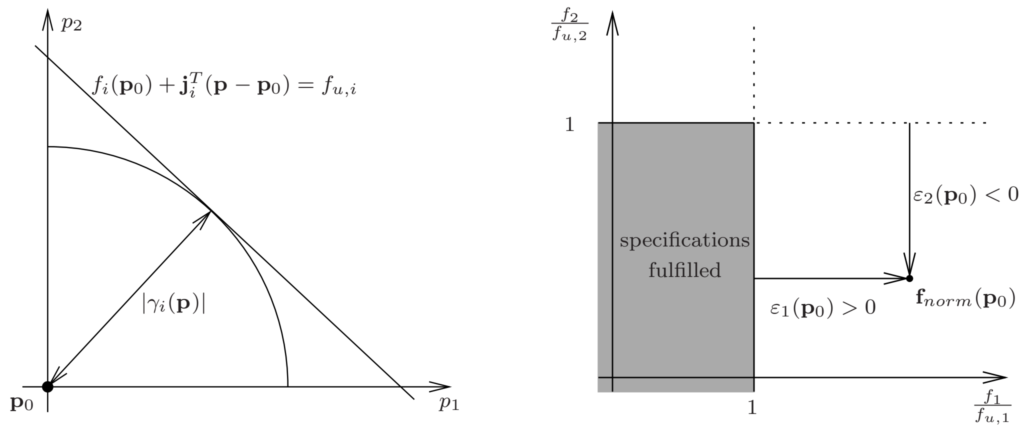
In other analog sizing approaches a measure referred to as *parameter distance* $\gamma_i(\mathbf{p})$ is suggested (e.g., [Eck98]). Considering only upper bounds, using the sensitivity vectors in (2.33), and following [Eck98], the parameter distance for performance $f_i\left(\mathbf{p}\right)$ can be given as:

$$\gamma_i(\mathbf{p}) = \frac{f_i\left(\mathbf{p}\right) - f_{u,i}}{\sqrt{\mathbf{j}_i^T\mathbf{j}_i}} \tag{2.36}$$

The parameter distance can be interpreted as the distance from a point in the parameter domain to the next point where the performance specification is exactly fulfilled using a linear performance model (see Figure 2.6a). As the sensitivity of the performances is computed in each step of an iterative algorithm, the norm for a performance changes in each iteration step. As a consequence, the quality of the quadratic model used in Chapter 5 and in the Sequential Quadratic Programming approach in Chapters 3 and 4 may be affected if the parameter distance is used.

In this thesis, a normalization with respect to the parameter bounds is used. I.e., assuming only upper bounds with $f_{u,i} \neq 0$:

$$\varepsilon_i(\mathbf{p}) = \frac{f_i\left(\mathbf{p}\right) - f_{u,i}}{\left|f_{u,i}\right|} \tag{2.37}$$


$p_2$
$f_i(\mathbf{p}_0) + \mathbf{j}_i^T(\mathbf{p} - \mathbf{p}_0) = f_{u,i}$
$|\gamma_i(\mathbf{p})|$
$\mathbf{p}_0$
$p_1$
$\frac{f_2}{f_{u,2}}$
1
specifications fulfilled
$\varepsilon_2(\mathbf{p}_0) < 0$
$\varepsilon_1(\mathbf{p}_0) > 0$
$\mathbf{f}_{norm}(\mathbf{p}_0)$
1
$\frac{f_1}{f_{u,1}}$

(a) Parameter distance for performance $f_i(\mathbf{p})$ in the parameter domain

(b) Performance-to-specification gap for two performances in the domain of normalized performances

Figure 2.6: Graphical interpretation of parameter distance and performance-to-specification gap

The value of $\varepsilon_i(\mathbf{p})$ is referred to as the size of the *performance-to-specification gap* or *performance gap* and is a measure for the distance from the performance value to the performance specification bound (see Figure 2.6b). It can be seen that the value for the gap size is positive if the performance violates the bound $f_{u,i}$ and negative if a specification is fulfilled. The performance-to-specification gaps are collected in an $N_\varepsilon$-dimensional vector

$$\boldsymbol{\varepsilon}(\mathbf{p}) = [\varepsilon_1(\mathbf{p}), ..., \varepsilon_{N_\varepsilon}(\mathbf{p})]^T \tag{2.38}$$

Assuming that the number of performances is equal to the number of performance-to-specification gaps, the Jacobian matrix for the performance gaps $\mathbf{J}_\varepsilon$ can be computed directly from (2.33) by

$$\mathbf{J}_\varepsilon = \left[\frac{1}{|f_{u,1}|}\mathbf{j}_1, ..., \frac{1}{|f_{u,N_\varepsilon}|}\mathbf{j}_{N_\varepsilon}\right]^T \tag{2.39}$$

## 2.4 Objective Functions

Using the definition of a performance-to-specification gap in Section 2.3.3, the task of analog sizing can be considered as a multi-objective minimization problem. I.e., all performance-to-specification gaps should be minimized at the same time. However, the optimization algorithms in Chapters 3, 4, and 5 require a formulation of the sizing task as a scalar optimization problem. For this mapping from a multi-objective optimization problem to a scalar one, an *objective function* $\varphi(\boldsymbol{\varepsilon}(\mathbf{p})) := \varphi(\mathbf{p})$ is used[1]:

$$\varphi : \boldsymbol{\varepsilon}(\mathbf{p}) \mapsto \varphi(\mathbf{p}) \tag{2.40}$$

[1]In this thesis, the constraints are not considered as part of the objective function.

The usability of an objective function strongly depends on the optimization task considered and on the optimization algorithm used. The properties of the objective function, which are claimed for the formulation of an analog sizing task as a minimization problem of the form

$$\min_{\mathbf{p}} \varphi(\mathbf{p}) \text{ s.t. } \mathbf{c}(\mathbf{p}) \geq \mathbf{0} \tag{2.41}$$

are described in the next section.

## 2.4.1 Claims for Scalar Objective Functions in Analog Sizing

The following claims are formulated for the objective functions in this thesis:

1. *The objective function should not add discontinuities or points of non-differentiability to the problem.*

   Assuming that the design objectives, i.e., the performances, are continuous and differentiable, gradient-based methods can be used. This property should not be affected by the objective function.

2. *The objective function should not penalize the over-fulfillment of specifications.*

   A point which fulfills all specifications can have a worse objective function value than a point which violates some specification if over-fulfillment is penalized. However, in some design scenarios one specification can only be fulfilled if another specification is over-fulfilled. In such a case, an optimization of an objective function which penalizes over-fulfillment cannot be used to reliably find a solution for the sizing task (cf. Appendix A.2).

   Penalizing the over-fulfillment of performances can also be critical if the objective function value for each point which fulfills the specifications is smaller than for each point which violates a specification (cf. claim 4)[2]: In this case, the penalization can cause a barrier which must be overcome to enter the domain where the specifications are fulfilled. I.e., local minima can be added where a gradient-based method may get stuck.

3. *Given two objectives, an alteration of the worse objective should have a higher influence on the objective function value than an equally large alteration of the better objective.*

   For an analog sizing task, a trade-off solution for different design objectives must be computed. If not all specifications can be fulfilled, it is preferable to have all performances close to the specification bounds rather than getting many very good performances at the cost of a few, strongly violated specifications. This goal can typically be achieved if the worst performance has the highest influence on the objective function value.

[2]Objective functions which fulfill claim 4 but do not fulfill claim 2 can be constructed, e.g., if weights are adapted dynamically to the size of each performance-to-specification gap during an optimization.

4. *The objective function value at a point which fulfills the specifications should be better than the objective function value at any point where a specification is violated.*

   Comparing two points by an objective function which is constructed according to the third claim, the alteration of a worse objective has a greater influence compared to the same alteration of a better one. However, some worse objectives might be deteriorated while improving the objective function if the improvement of a better objective is large enough or if more than one better objectives are improved.

   This characteristic of an objective function is acceptable as long as no point where specifications are violated has a lower objective function value than a point which solves the sizing task. In other cases, the minimization of the objective function starting at a point where all specifications are fulfilled might end up at a point where some specification is violated. As a consequence, no minimization – neither local nor global – can be used to solve the sizing task reliably and without additional effort if this fourth claim is violated (cf. Appendix A.2).

## 2.4.2 Objective Function Types

To convert a multi-objective problem into a scalar objective function, min-norm approaches can be used [Lin05]. A general form of min-norm objective functions for $N_\varepsilon$ performance-to-specification gaps (2.37) can be given by:

$$\varphi = \left( \sum_{i=1}^{N_\varepsilon} \left( \eta_i \left| \varepsilon_i(\mathbf{p}) \right| \right)^\kappa \right)^{\frac{1}{\kappa}} \tag{2.42}$$

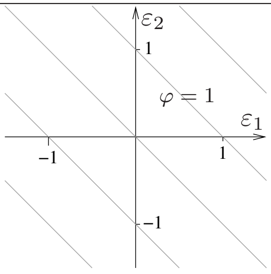
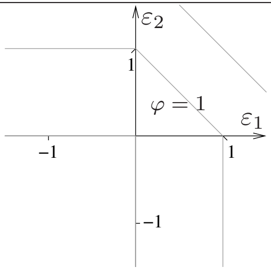
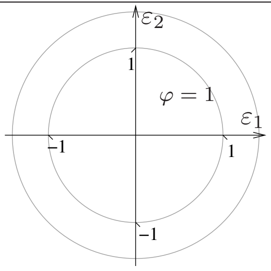
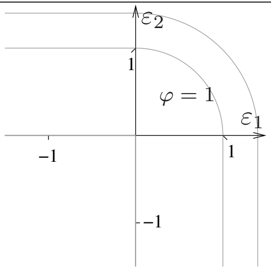
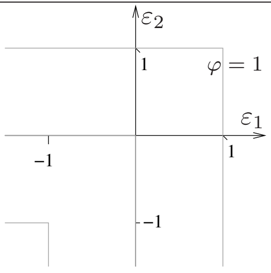
This class of norms is referred to as *weighted Hölder norms.* $\eta_i \geq 0$ represents a weight which can be used to increase the influence of a single term. $\kappa$ is a positive integer. Within this thesis the weight $\eta_i = 1$ is used for all performances. The norms for $\kappa = 1$, $\kappa = 2$, and $\kappa \to \infty$ are outlined and discussed in this section. Table 2.1 shows an overview of the discussed functions in order of appearance in this section.

$\underline{\kappa = 1}$

For $\kappa = 1$ and $\eta_i = 1$, the objective function is equal to a sum over the absolute values of the performance-to-specification gaps. The use of the absolute value causes a negative value of a gap – i.e., a performance which fulfills its specification – to increase the value of the objective function. As a consequence, the weighted sum of absolute errors violates claim 2 in Section 2.4.1, i.e., penalizes an over-fulfillment of the performances. In addition, this objective function is not differentiable at the axes $\varepsilon_i(\mathbf{p}) = 0$ (claim 1).

An objective function which does not penalize the over-fulfillment of the performances can be derived by replacing the absolute value by the non-absolute value $\varepsilon_i(\mathbf{p})$ or by

Table 2.1: Overview of important objective functions, and corresponding contour lines for two performance-to-specification gaps $\varepsilon_1$, $\varepsilon_2$ with $\eta_i = 1$, $\gamma = 0$.

| Objective Function | Formal Description | Contour Lines |
|---|---|---|
| Weighted Sum of Errors (2.43) | $\sum_{i=1}^{N_\varepsilon} \eta_i \cdot \varepsilon_i(\mathbf{p})$ |  |
| Truncated Weighted Sum of Errors (2.44) | $\sum_{i=1}^{N_\varepsilon} \eta_i \cdot \max(0, \varepsilon_i(\mathbf{p}))$ |  |
| Least-Squares (2.45) | $\sum_{i=1}^{N_\varepsilon} \eta_i \cdot \varepsilon_i(\mathbf{p})^2$ |  |
| Truncated Least-Squares (2.47) | $\sum_{i=1}^{N_\varepsilon} (\eta_i \cdot \max(0, \varepsilon_i(\mathbf{p}) + \gamma))^2$ |  |
| Max-Norm (2.49) | $\max_{i=1,\ldots,N_\varepsilon} (\varepsilon_i(\mathbf{p}))$ |  |

a truncated value $\max(0, \varepsilon_i(\mathbf{p}))$ for each performance gap. The resulting objective functions are referred to as the *weighted sum of errors*

$$\varphi = \sum_{i=1}^{N_\varepsilon} \eta_i \cdot \varepsilon_i(\mathbf{p}) \tag{2.43}$$

and the *truncated weighted sum of errors*

$$\varphi = \sum_{i=1}^{N_\varepsilon} \eta_i \cdot \max(0, \varepsilon_i(\mathbf{p})) \tag{2.44}$$

In both cases, an alteration in the size of the performance-to-specification gap with $\varepsilon_i(\mathbf{p}) > 0$ contributes linearly to the objective function value and is independent of the value of $\varepsilon_i(\mathbf{p})$. I.e., claim 3 is violated in both cases. As a consequence, a trade-off solution computed by these objective functions may over-fulfill some specifications at the cost of a few strongly violated performances if no solution for the sizing task exists where all specifications and constraints are fulfilled.

In addition, claim 4 is violated for (2.43). This can be observed, e.g., if two performance-to-specification gaps are assumed which can take values in the range $-10 < \varepsilon_1(\mathbf{p}) < 1$ and $-1 < \varepsilon_2(\mathbf{p}) < 1$. In such an example, a solution $\mathbf{p}_1$ with $\varepsilon_1(\mathbf{p}_1) = -10$ and $\varepsilon_2(\mathbf{p}_1) = 1$ obviously has a lower (better) objective function value for (2.43) than a solution $\mathbf{p}_2$ with $\varepsilon_1(\mathbf{p}_2) = -1$ and $\varepsilon_2(\mathbf{p}_2) = -1$. However, a negative value of the performance gap implies that the corresponding specification is fulfilled (cf. Section 2.3.3) such that only $\mathbf{p}_2$ is a point which fulfills all specifications. It can be concluded that an optimization over (2.43) might not find a solution of the sizing task where all specifications are fulfilled (cf. Appendix A.2).

To overcome this problem, negative values of the performance gaps are set to zero in the truncated weighted sum of errors (2.44). As a consequence, this objective function is always zero if all specifications are fulfilled and greater zero if any specification is violated. E.g., in the example above, the objective function value of (2.44) is zero for $\mathbf{p}_2$ but one for $\mathbf{p}_1$. It follows that claim 4 is fulfilled for (2.44). However, Table 2.1 shows that (2.44) is not differentiable at points along the axes $\varepsilon_i(\mathbf{p}) = 0$, i.e., claim 1 in Section 2.4.1 is violated.

It can be concluded that – concerning the claims in Section 2.4.1 – the min-norm objective function for $\kappa = 1$ and the derived objective functions (2.43) and (2.44) should not be selected for the analog sizing task.

$\underline{\kappa = 2}$

Setting $\kappa$ in (2.42) to $\kappa = 2$, the *Euclidean norm* of the performance gaps is used. The resulting objective function is referred to as *Least-Squares approach.* The square root and the absolute value in (2.42) can be omitted such that this objective function can be written as:

$$\varphi = \sum_{i=1}^{N_\varepsilon} \eta_i \cdot \varepsilon_i(\mathbf{p})^2 \tag{2.45}$$

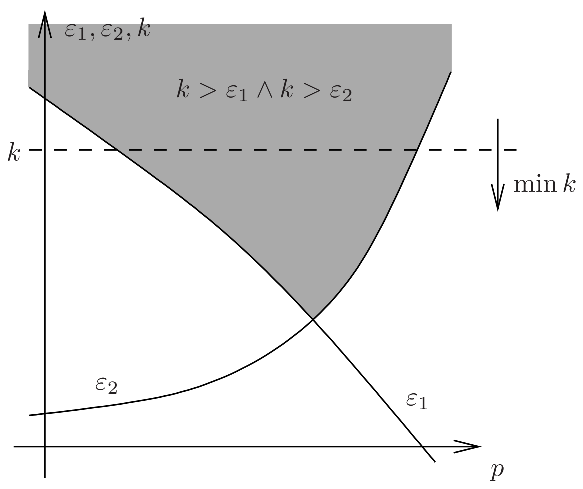

$\varepsilon_1, \varepsilon_2, k$
$k > \varepsilon_1 \wedge k > \varepsilon_2$
$k$
$\min k$
$\varepsilon_2$
$\varepsilon_1$
$p$

Figure 2.7: Interpretation of the min-max formulation

The Least-Squares objective function penalizes the over-fulfillment of the specifications (cf. Appendix A.2) because a negative value of a performance gap contributes to the objective function by a positive value. Analogously to (2.44) this problem can be solved by replacing each value of a performance gap $\varepsilon_i(\mathbf{p})$ in (2.45) by $\max(0, \varepsilon_i(\mathbf{p}))$:

$$\varphi = \sum_{i=1}^{N_\varepsilon} (\eta_i \cdot \max(0, \varepsilon_i(\mathbf{p})))^2 \tag{2.46}$$

This modified objective function is differentiable at each point if the performances are differentiable (claim 1). It does not penalize the over-fulfillment of specifications (claim 2) and puts – due to the squared form – greater weight on higher values of the performance gaps (claim 3). Due to the truncation it guarantees that the (global) minimum of the objective functions is a point which fulfills the specifications if such a point exists (claim 4). A slightly modified version of the objective function is used within this thesis for the new approach in Chapter 4 and is referred to as *truncated least squares objective function*:

$$\varphi_{lsq}(\mathbf{p}) = \sum_{i=1}^{N_\varepsilon} (\eta_i \cdot \max(0, \varepsilon_i(\mathbf{p}) + \gamma))^2 \tag{2.47}$$

$\gamma$ can be set to a small positive value and can be interpreted as a forced over-fulfillment of the performances. In the experiments of this thesis this value is set to $\gamma = 0.005$, i.e., the specifications are forced to be over-fulfilled by 0.5% of the specified value.

$\underline{\kappa \to \infty}$

Setting in (2.42) $\kappa \to \infty$ results in the *Chebyshev norm* of the performance-to-specification gap sizes. The absolute value in (2.42) causes the function not to support the over-fulfillment of specifications. However, the function can be modified according to (2.43), i.e., the value of each performance gap is used directly instead of the absolute value. The resulting objective function

$$\varphi = \lim_{\kappa \to \infty} \left( \sum_{i=1}^{N_\varepsilon} \left( \eta_i \varepsilon_i(\mathbf{p}) \right)^\kappa \right)^{\frac{1}{\kappa}} \tag{2.48}$$

can be interpreted for $\eta_i = 1$ as

$$\varphi_{max}\left(\mathbf{p}\right) = \max \boldsymbol{\varepsilon}(\mathbf{p}) := \max_{i=1,\ldots,N_\varepsilon} \left( \varepsilon_i(\mathbf{p}) \right) \tag{2.49}$$

and is referred to as *max-norm* in the following. It can be seen in Table 2.1 that $\varphi_{max}\left(\mathbf{p}\right)$ might be non-differentiable if the greatest two or more values of performance gaps are equal. To overcome this problem, an optimization problem of the form (2.41), i.e., for $\varphi\left(\mathbf{p}\right) = \varphi_{max}\left(\mathbf{p}\right)$

$$\min_{\mathbf{p}} \max \boldsymbol{\varepsilon}(\mathbf{p}) \text{ s.t. } \mathbf{c}\left(\mathbf{p}\right) \geq \mathbf{0} \tag{2.50}$$

can be rewritten (e.g., [EG02]) as a goal attainment optimization problem:

$$\min_{k,\mathbf{p}} k \text{ s.t.:} \quad \begin{aligned} &\mathbf{c}\left(\mathbf{p}\right) \geq \mathbf{0} \\ &k - \varepsilon_i(\mathbf{p}) \geq 0;\ i = 1, ..., N_\varepsilon \end{aligned} \tag{2.51}$$

A graphical interpretation of this optimization is given in Figure 2.7. The rewritten formulation – referred to as *min-max* formulation – fulfills all claims in Section 2.4.1 and is used for the new approach in Chapter 5. The pro of the min-max formulation is that – assuming reasonable normalized values of $\varepsilon_i(\mathbf{p})$ – it guarantees a uniform minimization of the values of all performance-to-specification gaps. However, due to the formulation in (2.51) additional constraints are added and the minimization over the max-norm is harder to solve in many cases than a minimization of the least squares objective function.

Exponentially Weighted Sum

Besides the weighted Hölder norms other objective functions can be found in literature. One function which was frequently used in former approaches for analog sizing is an exponentially weighted sum over the values of the performance gaps (e.g. [AEG+00]). However, it can be found that claim 4 in Section 2.4.1 is violated for this objective function (cf. Appendix A.2). In addition, the exponential function may result in objective function values of high magnitude and may cause numerical problems if the normalization should be kept unchanged over more than one iteration step. Therefore, it is not used in this thesis.

## 2.5 Design Objectives

Using the formal descriptions mentioned above, a formulation of the design task as a minimization problem can be given. This formulation depends on the design scenario considered.

### 2.5.1 Optimality of a Design

The design of analog circuits is a multi-objective task. In general, not every performance can be at its individual optimum at the same time and a trade-off solution is required. To distinguish between different options for this trade-off, the following definitions can be made:

**Definition 2.1:**
A discrete design point $\mathbf{d}_0 \in \mathbb{D}^N$ is a *c-feasible point* if all constraints of the design task (cf. Section 2.2) are fulfilled at this point.

**Definition 2.2:**
A discrete design point $\mathbf{d}_0 \in \mathbb{D}^N$ is an *s-feasible point* if all specifications and constraints of the design task (cf. Sections 2.3 and 2.2) are fulfilled at this point.

**Definition 2.3:**
A c-feasible discrete design point $\mathbf{d}_0 \in \mathbb{D}^N$ is a *Pareto-optimal point* if at this point no performance can be improved without worsening any other performance (cf. [MG09]).

**Definition 2.4:**
A c-feasible discrete design point $\mathbf{d}_0 \in \mathbb{D}^N$ is a *local Pareto-optimal point* if in a neighborhood $\mathbb{D}^N_{neighbor} = \{\mathbf{d}\,|\mathbf{d}_1 < \mathbf{d} < \mathbf{d}_2\}$, with $\mathbf{d}_1, \mathbf{d}_2 \in \mathbb{D}^N$, no performance can be improved without worsening any other performance (cf. [Wan75]). A Pareto-optimal solution point is always local Pareto-optimal.

**Definition 2.5:**
A *local Pareto-optimal design* is a design at a local Pareto-optimal point.

**Definition 2.6:**
An *optimal design* is a design at an s-feasible and local Pareto-optimal point.

**Definition 2.7:**
A *suboptimal design* is a design at any s-feasible point.

The defined demands for an optimal design are more restrictive than for a suboptimal design and require the objective function to support the improvement of performances which fulfill the specifications. Therefore, the truncated least squares objective function (2.47) – wherein each term is zero if the corresponding performance fulfills its specification – cannot be used without further modifications to compute an optimal design. In contrast, a local optimum of the max-norm objective function is always an optimal design if it is s-feasible. For the task of finding a suboptimal design, the truncated least squares objective function can be used as well as the max-norm formulation.

The quality of an optimal design is obviously better than the quality of a suboptimal design. However, the computational cost of finding an optimal design typically is much higher and it is sufficient for most design scenarios to find any s-feasible design. This thesis focuses on the computation of a suboptimal design. This task is equivalent to the task of finding a c-feasible point $\mathbf{p}^*$ which fulfills:

$$\underset{i=1,...,N_\varepsilon}{\forall} \quad \varepsilon_i(\mathbf{p}^*) \leq 0 \tag{2.52}$$

(2.52) can be used as a stop criterion in the optimization algorithms described in Chapters 3, 4 and 5.

### 2.5.2 Design for Feasibility

The term *design for feasibility* refers to the computation of a c-feasible sizing, i.e., to a sizing which fulfills the constraints defined in Section 2.2. The constraints should be typically fulfilled for the nominal point of the operating parameters $\mathbf{o} = \mathbf{o}_0$ and the mean value of the process parameters $\mathbf{s} = \mathbf{s}_0$. I.e., any design parameter point $\mathbf{d}^*$ should be found such that

$$\mathbf{d}^* \in \mathbb{A}_c = \left\{ \mathbf{d} \in \mathbb{D}^N \middle| \, \mathbf{c}(\mathbf{d}, \mathbf{o}_0, \mathbf{s}_0) \geq \mathbf{0} \right\} \tag{2.53}$$

$\mathbb{A}_c$ is referred to as the *acceptance region with respect to the constraints.* This task must be solved, e.g., to find an initial solution for the computation of the nominal design described below. The minimization problem can be formulated as

$$\min_{\mathbf{d} \in \mathbb{D}^N} \varphi(\mathbf{d}, \mathbf{o}_0, \mathbf{s}_0) \tag{2.54}$$

For $\varphi(\mathbf{d}, \mathbf{o}_0, \mathbf{s}_0)$ the max-norm formulation as well as a truncated least squares formulation can be used if the performance-to-specification gaps are replaced by the gap sizes regarding the constraints.

### 2.5.3 Nominal Design

A *nominal design* is s-feasible – i.e., fulfills the constraints defined in Section 2.2 as well as the specifications for the performances defined in Section 2.3 – for nominal operating conditions $\mathbf{o} = \mathbf{o}_0$ and mean values of the process parameters $\mathbf{s} = \mathbf{s}_0$. Using (2.52) and (2.53), a design parameter point $\mathbf{d}^*$ should be found which fulfills:

$$\mathbf{d}^* \in \mathbb{A}_{nom} = \mathbb{A}_c \cap \left\{ \mathbf{d} \in \mathbb{D}^N \middle| \underset{i=1,...,N_\varepsilon}{\forall} \quad \varepsilon_i(\mathbf{d}, \mathbf{o}_0, \mathbf{s}_0) \leq 0 \right\} \tag{2.55}$$

$\mathbb{A}_{nom}$ is referred to as the *acceptance region with respect to the constraints and performances.* To find such a point the minimization problem

$$\min_{\mathbf{d} \in \mathbb{D}^N} \varphi(\mathbf{d}, \mathbf{o}_0, \mathbf{s}_0) \text{ s.t. } \mathbf{c}(\mathbf{d}, \mathbf{o}_0, \mathbf{s}_0) \geq \mathbf{0} \tag{2.56}$$

is used.

$\varphi = \varphi_{lsq}$ (2.47) or $\varphi = \varphi_{max}$ (2.48) together with stop criterion (2.52) can be used to find any nominal design. A nominal design is required, e.g., as initial solution for a tolerance design task or if no operating and process parameters should be considered.

In some scenarios also an optimal design is of interest, e.g., if power and area of a circuit should be optimized. For this task all performance specifications which should not be optimized can be considered as constraints in (2.56) and the remaining performances can be optimized, e.g., using $\varphi = \varphi_{max}$. With slight modifications, the algorithms in Chapters 4 and 5 can also be used to find an optimal nominal design.

### 2.5.4 Tolerance Design

For a typical analog sizing task, operating conditions are defined where the performances of a circuit must fulfill the specifications. In addition, process variations should be considered to make the circuit robust and to achieve a high yield during manufacturing. The design considering operating conditions and predefined yield requirements is referred to as *tolerance design*.

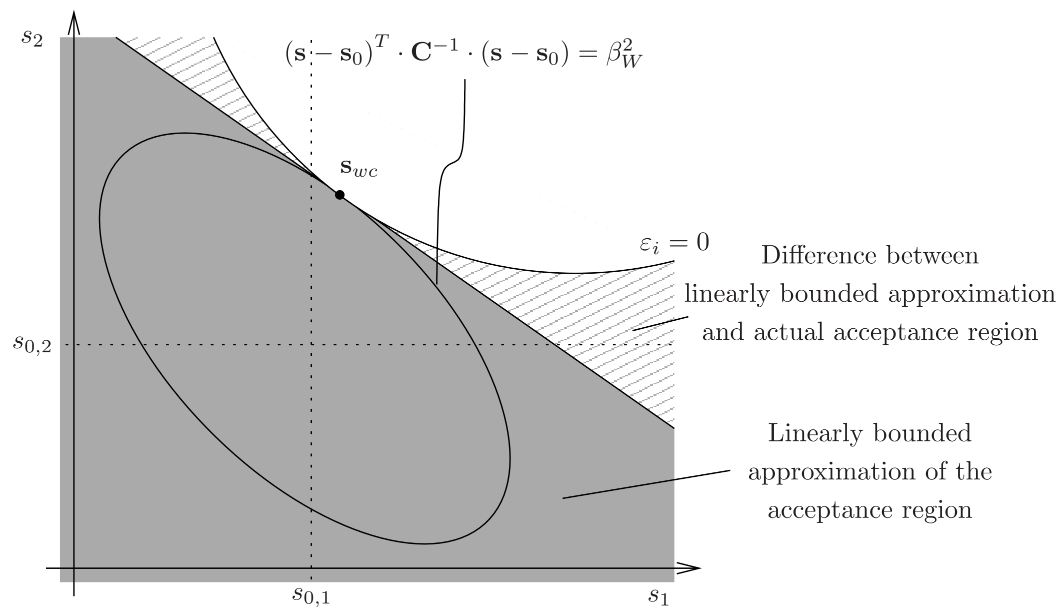
Considering a single performance gap, the parametric yield $Y_i$ of an analog circuit with respect to the corresponding performance specification, i.e., the probability that the value of the corresponding performance-to-specification gap $\varepsilon_i(\mathbf{p})$ is smaller than 0, can be computed by:

$$Y_i = \int_{-\infty}^{0} \text{pdf}\left(\varepsilon_i\right) d\varepsilon_i \tag{2.57}$$

This integral cannot be solved directly because the probability density function $\text{pdf}\left(\varepsilon_i\right)$ is not explicitly known. However, using (2.24) for a fixed value $\beta^2\left(\mathbf{s}\right) = \beta_W^2$, the yield can be approximated by

$$Y_i \approx \bar{Y}_i = \int_{-\infty}^{\beta_W} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\beta^2\right) d\beta \tag{2.58}$$

if the specifications are exactly fulfilled at one point of the hull of the ellipsoid defined by (2.24) and $\beta_W$. For the approximation, the acceptance region in the domain of the process parameters – i.e., the subdomain of the process parameter domain where the specifications are fulfilled – is assumed to be linearly bounded. This is visualized in Figure 2.8 where the approximated yield corresponds to the integral over the probability density function of $\mathbf{s}$ in the linearly approximated area where the specification is fulfilled. A derivation following [Gra07] is provided in Appendix B.2. It can be seen from Figure 2.8 that – due to the assumed linearity of the acceptance region bound – an error occurs in the yield approximation (integral over pdf($\mathbf{s}$) in hatched area). However,



$s_2$
$(\mathbf{s}-\mathbf{s}_0)^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{s}-\mathbf{s}_0) = \beta_W^2$
$\mathbf{s}_{wc}$
$\varepsilon_i = 0$
Difference between linearly bounded approximation and actual acceptance region
$s_{0,2}$
Linearly bounded approximation of the acceptance region
$s_{0,1}$
$s_1$

Figure 2.8: Acceptance region (gray and hatched area) and its linearly bounded approximation (gray area) in the process parameter domain (cf. (2.63)). The linear bound is computed at the boundary point $\mathbf{s}_{wc}$ between a contour line of pdf($\mathbf{s}$) and the actual performance specification bound $\varepsilon_i = 0$.

Table 2.2: Relation of $\beta_W$ and approximated yield $\bar{Y}_i$

| $\beta_W$ | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| $\bar{Y}_i$ [%] | 50.00 | 84.13 | 97.72 | 99.87 | 99.997 | >99.999 |

the integral over the probability density function in this area is small and the error is negligible in practical approaches.

Using (2.58) a predefined yield can be expressed in terms of $\beta_W$ (see Table 2.2). I.e., using (2.24), the requirement of a certain yield can be interpreted as the claim to fulfill the specifications of the performances for all process parameter points in a tolerance domain

$$\mathbb{T}_s^{N_s} = \left\{ \mathbf{s} | \left(\mathbf{s}-\mathbf{s}_0\right)^T \cdot \mathbf{C}^{-1} \cdot \left(\mathbf{s}-\mathbf{s}_0\right) \leq \beta_W^2 \right\} \tag{2.59}$$

I.e., for fixed design parameters $\mathbf{d} = \hat{\mathbf{d}}$ and operating parameters $\mathbf{o} = \hat{\mathbf{o}}$, the performance specifications must be fulfilled and the size of the performance-to-specification gap $\varepsilon_i(\mathbf{p})$ must be negative for each performance $i$ at its *worst case process parameter point* $\mathbf{s}_{wci}$

$$\mathbf{s}_{wci} = \arg \max_{\mathbf{s} \in \mathbb{T}_s^{N_s}} \varepsilon_i \left( \hat{\mathbf{d}}, \hat{\mathbf{o}}, \mathbf{s} \right) \tag{2.60}$$

In practical cases $\mathbf{s}_{wci}$ typically lies on the hull of the ellipsoid defined by $\beta_W^2$. Therefore, $\beta_W$ is referred to as the *worst-case distance.*

Satisfying the claim that $\mathbf{s}_{wci}$ fulfills the specifications and constraints can only guarantee that the yield for an analog circuit with respect to a single variable is at least

$$Y_i \geq \underset{\mathbf{s}\in\mathbb{T}_s^{N_s}}{\int \cdots \int} \mathrm{pdf}(\mathbf{s})d\mathbf{s} \tag{2.61}$$

wherein pdf($\mathbf{s}$) is defined according to (2.23). Although the yield which can be guaranteed due to (2.61) is typically lower than the predefined yield requirements (an approximation is provided in [Gra07]), the approximation in (2.58) is sufficiently accurate in practice.

Analogous to (2.60), the requirement to fulfill the performance specifications for all operating conditions can be formulated as the claim to fulfill the performances at the *worst case operating points* $\mathbf{o}_{wci}$. For fixed design parameters $\mathbf{d} = \hat{\mathbf{d}}$ and process parameters $\mathbf{s} = \hat{\mathbf{s}}$ the worst case operating point for performance $i$ is defined with (2.21) as

$$\mathbf{o}_{wci} = \arg \max_{\mathbf{o}\in\mathbb{T}_o^{N_o}} \varepsilon_i \left(\hat{\mathbf{d}}, \mathbf{o}, \hat{\mathbf{s}}\right) \tag{2.62}$$

For a tolerance design the performance specifications should be fulfilled considering operating and process parameters. This can be formulated as the claim that for a design parameter point $\mathbf{d} = \mathbf{d}^*$ the specifications must be fulfilled at the worst case points with respect to operating parameters and process parameters. Using (2.37), (2.60), (2.62) the objective of finding a tolerance design can be formulated as the task of computing a point in the acceptance region for a tolerance design

$$\mathbf{d}^* \in \mathbb{A}_{tol} = \mathbb{A}_c \cap \left\{ \mathbf{d} \in \mathbb{D}^N \,\middle|\, \underset{i=1,...,N_\varepsilon}{\forall} \varepsilon_i(\mathbf{d}, \mathbf{o}_{wci}, \mathbf{s}_{wci}) \leq 0 \right\} \tag{2.63}$$

Thus, the optimization task can be formulated as

$$\min_{\mathbf{d}\in\mathbb{D}^N} \varphi(\mathbf{d}, \mathbf{o}_{wc1}, ..., \mathbf{o}_{wcN_\varepsilon}, \mathbf{s}_{wc1}, ..., \mathbf{s}_{wcN_\varepsilon}) \text{ s.t. } \mathbf{c}(\mathbf{d}, \mathbf{o}_0, \mathbf{s}_0) \tag{2.64}$$

Herein $\mathbf{o}_{wc1}, ..., \mathbf{o}_{wcN_\varepsilon}$ and $\mathbf{s}_{wc1}, ..., \mathbf{s}_{wcN_\varepsilon}$ depend on the design parameter point, i.e., must be re-computed in each step of an iterative optimization algorithm.

Analogous to the nominal design, $\varphi$ can be the truncated least squares objective function $\varphi_{lsq}$ or the max-norm $\varphi_{max}$ (2.52). Also, the computation of an optimal tolerance design is possible as discussed for a nominal design in Section 2.5.3.

### 2.5.5 Yield Optimization

If the yield is supposed to be as high as possible and no predefined yield is given, the task of analog sizing can be formulated as a maximization of the distances to the worst case parameter points. This task is not considered in this thesis. However, using the formulation of a geometric-yield optimization [Gra07], the presented algorithms can be applied to this task with only slight modifications.

# Chapter 3

# Optimization Algorithms

The analog sizing task can be formulated as an optimization problem. In this chapter the basic algorithms implemented within this thesis are introduced and new enhancements are presented which are required to apply these algorithms to the analog sizing task. The presented algorithms are used in Chapters 4 and 5 for the new methods to solve the discrete sizing task.

In Section 3.1 the optimality conditions for continuous optimization are introduced which are the basis for the continuous optimization algorithms in Sections 3.2, 3.3, and 3.4. In Sections 3.2, 3.3, and 3.4 a linear, a quadratic, and a general nonlinear optimization problem on a continuous domain are formulated and appropriate solution methods are presented. Continuous problems appear in a discrete analog sizing task, e.g., if an optimization problem is to be solved in the relaxed domain of the design parameters (cf. (2.14) in Chapter 2.1.1).

In Section 3.5 the discrete nonlinear optimization problem is characterized and the Branch-and-Bound algorithm is introduced to solve it.

## 3.1 First-Order and Second-Order Optimality Conditions

For continuous optimization problems the first-order necessary optimality conditions and the second-order necessary and sufficient optimality conditions are used to qualify the local optimality of a point.

The first-order necessary conditions are also referred to as *Karush-Kuhn-Tucker conditions* or *KKT conditions*. To formulate these conditions, a constrained optimization problem of the form

$$\min_{\mathbf{d}} \varphi(\mathbf{d}) \quad \text{s.t.} \quad \begin{aligned} c_l(\mathbf{d}) &\geq 0; \text{ for } l \in \mathcal{I} \\ c_l(\mathbf{d}) &= 0; \text{ for } l \in \mathcal{E} \end{aligned} \tag{3.1}$$

with the Lagrangian function

$$\mathcal{L}(\mathbf{d}, \boldsymbol{\lambda}) = \varphi(\mathbf{d}) - \sum_{l \in (\mathcal{E} \cup \mathcal{I})} \lambda_l c_l(\mathbf{d}) \tag{3.2}$$

is assumed. $\lambda_l$ is the Lagrangian multiplier which corresponds to constraint $c_l(\mathbf{d})$. $\mathcal{E}$ and $\mathcal{I}$ are the index sets for the equality and inequality constraints, respectively. The objective function and the constraints are assumed to be differentiable. It can be proved (e.g., [NW99]) that – if a point $\mathbf{d}^*$ is optimal – the first-order necessary conditions

$$\nabla_{\mathbf{d}} \mathcal{L}(\mathbf{d}^*, \boldsymbol{\lambda}^*) = \mathbf{0} \tag{3.3}$$

$$c_l(\mathbf{d}^*) \geq 0 \qquad l \in \mathcal{I} \tag{3.4}$$

$$c_l(\mathbf{d}^*) = 0 \qquad l \in \mathcal{E} \tag{3.5}$$

$$\lambda_l \geq 0 \qquad l \in \mathcal{I} \tag{3.6}$$

$$\lambda_l c_l(\mathbf{d}^*) = 0 \qquad l \in (\mathcal{E} \cup \mathcal{I}) \tag{3.7}$$

must be fulfilled. For further discussion, the following definition is used:

**Definition 3.1:**
A constraint $c_l(\mathbf{d})$ is *active* at $\mathbf{d}^*$ if $c_l(\mathbf{d}^*) = 0$. The index set of active constraints at $\mathbf{d}^*$

$$\mathcal{A} := \mathcal{A}(\mathbf{d}^*) = \mathcal{E} \cup \{l \in \mathcal{I} \,|\, c_l(\mathbf{d}^*) = 0\} \tag{3.8}$$

is referred to as *active set*.

The KKT conditions are necessary conditions for local optimality. Thus, only the locally limiting constraints, i.e., equality constraints and active inequality constraints, must be considered. For this purpose, the Lagrangian multipliers are defined to be $\lambda_k = 0$ for all non-active constraints. As each active constraint fulfills $c_l(\mathbf{d}^*) = 0$ and each $\lambda_l = 0$ for inactive constraints, (3.7) must be fulfilled. Furthermore, (3.4) and (3.5) must be fulfilled if $\mathbf{d}^*$ is a feasible solution. The remaining equations (3.3) and (3.6) can be derived, using that - if $\mathbf{d}^*$ defines a minimum - no feasible descent direction can exist at $\mathbf{d}^*$ (see, e.g., [NW99]). A negative Lagrangian multiplier for an inequality constraint – i.e., a violation of (3.6) – implies that the solution can be further improved if the corresponding constraint is forced to be inactive.

The KKT conditions might be fulfilled although $\mathbf{d}^*$ is no local solution for (3.1) if $\nabla_{\mathbf{d}} \varphi(\mathbf{d}^*)^T \Delta\mathbf{d} = 0$ is fulfilled for some direction $\Delta\mathbf{d} \neq \mathbf{0}$. Thus, the second derivative of the Lagrangian function (3.2) is used to identify the optimality of a point.

Let the KKT conditions be fulfilled at $\mathbf{d}^*$ and let direction $\Delta\mathbf{d}$ lead to a feasible point different from $\mathbf{d}^*$. If such a direction $\Delta\mathbf{d}$ exists and if $\Delta\mathbf{d}$ leads to an improvement of the objective function, it must fulfill (e.g., [NW99]):

$$\Delta\mathbf{d} \in \mathcal{F} = \left\{ \Delta\mathbf{d} \neq \mathbf{0} \;\middle|\; \begin{array}{ll} \nabla_{\mathbf{d}} c_l\left(\mathbf{d}^*\right)^T \Delta\mathbf{d} = 0 & ; \text{if } l \in \mathcal{E} \cup \{l \in \mathcal{I} \cap \mathcal{A} \,|\, \lambda_l^* > 0\} \\ \nabla_{\mathbf{d}} c_l\left(\mathbf{d}^*\right)^T \Delta\mathbf{d} \geq 0 & ; \text{if } l \in \{l \in \mathcal{I} \cap \mathcal{A} \,|\, \lambda_l^* = 0\} \end{array} \right\} \tag{3.9}$$

With (3.9), the *Taylor approximation* of (3.2) cut after the second term and determined at $\mathbf{d}^*$ can be given as

$$\begin{aligned} \mathcal{L}(\mathbf{d}^* + \Delta\mathbf{d}, \boldsymbol{\lambda}^*) &\approx \mathcal{L}(\mathbf{d}^*, \boldsymbol{\lambda}^*) + \nabla_{\mathbf{d}}\mathcal{L}(\mathbf{d}^*, \boldsymbol{\lambda}^*)^T \Delta\mathbf{d} + \tfrac{1}{2}\Delta\mathbf{d}^T \nabla^2_{\mathbf{dd}}\mathcal{L}(\mathbf{d}^*, \boldsymbol{\lambda}^*)\Delta\mathbf{d} \\ &\overset{(3.3),(3.7),(3.9)}{=} \varphi(\mathbf{d}^*) + \tfrac{1}{2}\Delta\mathbf{d}^T \nabla^2_{\mathbf{dd}}\mathcal{L}(\mathbf{d}^*, \boldsymbol{\lambda}^*)\Delta\mathbf{d} \end{aligned} \tag{3.10}$$

Due to (3.2) and $\lambda_l^* \cdot c_l\left(\mathbf{d}^* + \Delta\mathbf{d}\right) \geq 0$, the value of the Lagrangian function at $\mathbf{d}^* + \Delta\mathbf{d}$ must also fulfill $\mathcal{L}(\mathbf{d}^* + \Delta\mathbf{d}, \boldsymbol{\lambda}^*) \leq \varphi(\mathbf{d}^* + \Delta\mathbf{d})$. Thus, the point $\mathbf{d}^*$ is a strict optimum for problem (3.1) if

$$\underset{\Delta\mathbf{d} \in \mathcal{F}}{\forall} \quad \Delta\mathbf{d}^T \nabla^2_{\mathbf{dd}}\mathcal{L}(\mathbf{d}^*, \boldsymbol{\lambda}^*)\Delta\mathbf{d} > 0 \tag{3.11}$$

This condition is referred to as the *second-order sufficient condition.* If instead of the strict inequality

$$\underset{\Delta\mathbf{d} \in \mathcal{F}}{\forall} \quad \Delta\mathbf{d}^T \nabla^2_{\mathbf{dd}}\mathcal{L}(\mathbf{d}^*, \boldsymbol{\lambda}^*)\Delta\mathbf{d} \geq 0 \tag{3.12}$$

is fulfilled, the inequality is referred to as the *second-order necessary condition.* The equality implies that the point might not be a local optimum for (3.1) if a higher order term in the Taylor approximation causes an improvement of the objective function.

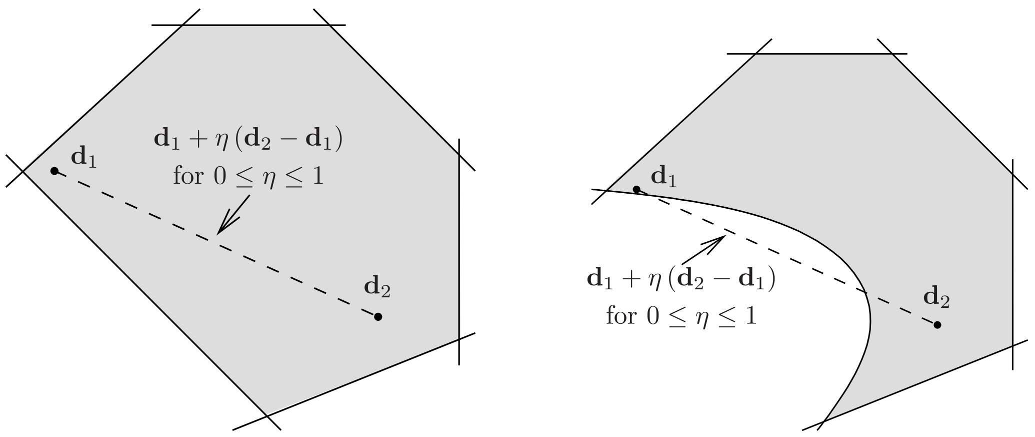
For discrete optimization algorithms neither the first-order nor the second-order optimality conditions can be used to qualify the optimum. This assertion can be easily verified because in many cases a direction of improvement can be found for the relaxation of the discrete problem at the discrete optimum.

## 3.2 Linear Programming in a Continuous Domain

### 3.2.1 Linear Programming Task

A linear program refers to an optimization problem with linear objective function and linear constraints and can be given as:

$$\min_{\mathbf{d}} \; \mathbf{k}^T \cdot \mathbf{d} \text{ s.t.} \quad \begin{aligned} \mathbf{A}_g \cdot \mathbf{d} &\geq \mathbf{b}_g \\ \mathbf{A}_l \cdot \mathbf{d} &\leq \mathbf{b}_l \\ \mathbf{A}_e \cdot \mathbf{d} &= \mathbf{b}_e \end{aligned} \tag{3.13}$$


$\mathbf{d}_1$
$\mathbf{d}_1 + \eta\,(\mathbf{d}_2 - \mathbf{d}_1)$
for $0 \leq \eta \leq 1$
$\mathbf{d}_2$
$\mathbf{d}_1$
$\mathbf{d}_1 + \eta\,(\mathbf{d}_2 - \mathbf{d}_1)$
for $0 \leq \eta \leq 1$
$\mathbf{d}_2$

(a) Convex domain (shaded area) (b) Non-convex domain (shaded area)

Figure 3.1: Convex vs. non-convex domain

$\mathbf{k}$,$\mathbf{b}_g$,$\mathbf{b}_l$,$\mathbf{b}_e$,$\mathbf{A}_g$,$\mathbf{A}_l$, and $\mathbf{A}_e$ are constant vectors and matrices, respectively.

The feasible region of a linear program is the domain

$$\mathbb{D}^N_{lin,feas} = \{\mathbf{d}\,|\mathbf{A}_g \cdot \mathbf{d} \geq \mathbf{b}_g \wedge \mathbf{A}_l \cdot \mathbf{d} \leq \mathbf{b}_l \wedge \mathbf{A}_e \cdot \mathbf{d} = \mathbf{b}_e\} \tag{3.14}$$

If this domain is not empty, it is convex, i.e., each point on a straight line between two feasible points is also feasible (cf. Figure 3.1):

$$\mathbf{d}_1, \mathbf{d}_2 \in \mathbb{D}^N_{lin,feas} \quad \Rightarrow \quad \underset{0 \leq \eta \leq 1}{\forall} \quad \mathbf{d}_1 + \eta\,(\mathbf{d}_2 - \mathbf{d}_1) \in \mathbb{D}^N_{lin,feas} \tag{3.15}$$

As a consequence, the solution is a vertex of the polytope which is defined by the constraints if an unique solution exists for (3.13). It follows that the task of finding the optimum of the problem is equivalent to the task of finding a set of constraints which defines the vertex where the optimum is. This is used in the simplex algorithm.

### 3.2.2 Preprocessing for Simplex Algorithm

For the simplex method an optimization problem of the form

$$\min_{\mathbf{d}}\ \mathbf{k}^T \cdot \mathbf{d} \text{ s.t.} \quad \begin{aligned} \mathbf{A} \cdot \mathbf{d} &= \mathbf{b} \\ \mathbf{d} &\geq \mathbf{0} \end{aligned} \tag{3.16}$$

is required. The following operations can be applied to get from the general form of a linear program (3.13) to (3.16) (e.g., [NW99]):

1. The inequalities in (3.13) can be transformed into equivalent equalities by subtracting or adding slack variables $\boldsymbol{\vartheta} \geq \mathbf{0}$:

$$\mathbf{A}_g\mathbf{d} \geq \mathbf{b}_g \text{ can be transformed into } \left[\begin{array}{c:c} \mathbf{A}_g & -\mathbf{I} \end{array}\right] \cdot \begin{bmatrix} \mathbf{d} \\ \boldsymbol{\vartheta}_g \end{bmatrix} = \mathbf{b}_g \tag{3.17}$$

$$\mathbf{A}_l\mathbf{d} \leq \mathbf{b}_l \text{ can be transformed into } \left[\begin{array}{c:c} \mathbf{A}_l & \mathbf{I} \end{array}\right] \cdot \begin{bmatrix} \mathbf{d} \\ \boldsymbol{\vartheta}_l \end{bmatrix} = \mathbf{b}_l \tag{3.18}$$

   $\mathbf{I}$ is the identity matrix.

2. If any variable $d_m$ can be smaller than zero, i.e., $d_m \geq 0$ is not fulfilled, the variable can be split into a positive part $d_m^+ = \max(0, d_m)$ and a negative part $d_m^- = \max(0, -d_m)$, and can be expressed by $d_m = d_m^+ - d_m^-$. I.e., the optimization program of the form (3.16) with some $d_m \in \mathbb{R}$

$$\min_{\mathbf{d}} \begin{bmatrix} k_1 \\ \vdots \\ k_m \\ \vdots \\ k_N \end{bmatrix}^T \cdot \begin{bmatrix} d_1 \\ \vdots \\ d_m \\ \vdots \\ d_N \end{bmatrix} \quad \text{s.t.} \quad \begin{aligned} &[\mathbf{a}_1, ..., \mathbf{a}_m, ...\mathbf{a}_N] \cdot \begin{bmatrix} d_1 \\ \vdots \\ d_m \\ \vdots \\ d_N \end{bmatrix} = \mathbf{b} \\ &d_m \in \mathbb{R} \\ &d_i \geq 0 \text{ for } i \neq m \end{aligned} \tag{3.19}$$

   can be transformed into

$$\min_{\mathbf{d}} \begin{bmatrix} k_1 \\ \vdots \\ k_m \\ -k_m \\ \vdots \\ k_N \end{bmatrix}^T \cdot \begin{bmatrix} d_1 \\ \vdots \\ d_m^+ \\ d_m^- \\ \vdots \\ d_N \end{bmatrix} \quad \text{s.t.} \quad \begin{aligned} &[\mathbf{a}_1, ..., \mathbf{a}_m, -\mathbf{a}_m, ...\mathbf{a}_N] \cdot \begin{bmatrix} d_1 \\ \vdots \\ d_m^+ \\ d_m^- \\ \vdots \\ d_N \end{bmatrix} = \mathbf{b} \\ &d_m^+ \geq 0;\ d_m^- \geq 0 \\ &d_i \geq 0 \text{ for } i \neq m \end{aligned} \tag{3.20}$$

   Due to the minimization, either $d_m^+$ or $d_m^-$ is equal to zero and the operators $d_m^+ = \max(0, d_m)$ and $d_m^- = \max(0, -d_m)$ can be omitted.

In addition to the required form of the optimization problem in (3.16), the number of parameters in the simplex algorithm is assumed to be greater than the number of equality constraints in (3.16). Otherwise some constraints are redundant, the solution of the optimization problem is uniquely defined by the constraints, or no solution exists. For practical approaches this means that redundant constraints must be deleted before the simplex algorithm is executed. The resulting set of constraints is referred to as the *working set.*

### 3.2.3 Simplex Algorithm

For the simplex algorithm, an optimization problem of the form (3.16) with $r$ linearly independent constraints and $N > r$ parameters is assumed. The optimal solution is a vertex of the ploytope which is defined by the constraints if a unique solution exists. For a problem of this form the following definitions can be made:

**Definition 3.2:**
A *basic feasible point* is feasible, i.e., $\mathbf{d} \in \{\mathbf{d}\,|\mathbf{A}\cdot\mathbf{d} = \mathbf{b}\}$, it is uniquely defined by $r$ linearly independent columns of $\mathbf{A}$, and has at most $r$ non-zero elements. Each non-zero element corresponds to a column of $\mathbf{A}$ that defines the basic feasible point. A basic feasible point corresponds to a vertex of the polytope defined by the constraints [NW99].

**Definition 3.3:**
The *basis set* $\mathbb{B}$ is the index set of the $r$ linearly independent columns of $\mathbf{A}$ which define a basic feasible point.

**Definition 3.4:**
The *non-basis set* $\bar{\mathbb{B}}$ is the index set of the $N - r$ columns of $\mathbf{A}$ which do not define the basic feasible point. I.e., $\bar{\mathbb{B}} = \{1, ..., N\} \setminus \mathbb{B}$.

**Definition 3.5:**
A *basis variable* is a variable which corresponds to an index in $\mathbb{B}$.

**Definition 3.6:**
A *non-basis variable* is a variable which corresponds to an index in $\bar{\mathbb{B}}$. Due to Definition 3.2, the value of a non-basis variable is zero.

**Definition 3.7:**
The *matrix of basic columns* is a matrix that consists of the columns of $\mathbf{A}$ which define the basic feasible point.

To compute the basic feasible points, the columns $\mathbf{a}_i$ of $\mathbf{A}$ with index $i \in \mathbb{B}$ can be collected in a matrix $\mathbf{B}$ and all columns of $\mathbf{A}$ with index $i \in \bar{\mathbb{B}}$ can be collected in a matrix $\mathbf{N}$. Correspondingly, the vectors $\mathbf{k}$ and $\mathbf{d}$ can be split into vectors $\mathbf{k}_B$ and $\mathbf{k}_N$ and $\mathbf{d}_B$ and $\mathbf{d}_N$, respectively. Therefore, the optimization problem (3.16) can be formulated as:

$$\min_{\mathbf{d}} \left[\begin{array}{c:c} \mathbf{k}_B^T & \mathbf{k}_N^T \end{array}\right] \cdot \left[\begin{array}{c} \mathbf{d}_B \\ \hdashline \mathbf{d}_N \end{array}\right] \quad \text{s.t.} \quad \begin{array}{c} \left[\begin{array}{c:c} \mathbf{B} & \mathbf{N} \end{array}\right] \cdot \left[\begin{array}{c} \mathbf{d}_B \\ \hdashline \mathbf{d}_N \end{array}\right] = \mathbf{b} \\ \mathbf{d}_B \geq 0;\ \mathbf{d}_N \geq 0 \end{array} \tag{3.21}$$

With Lagrange multipliers $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}^T = \left[\boldsymbol{\mu}_B^T, \boldsymbol{\mu}_N^T\right]$, the corresponding Lagrangian function is:

$$\mathcal{L}(\mathbf{d}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{k}_B^T\mathbf{d}_B + \mathbf{k}_N^T\mathbf{d}_N - \boldsymbol{\lambda}^T \cdot (\mathbf{B}\mathbf{d}_B + \mathbf{N}\mathbf{d}_N - \mathbf{b}) - \boldsymbol{\mu}_B^T\mathbf{d}_B - \boldsymbol{\mu}_N^T\mathbf{d}_N \tag{3.22}$$

**Algorithm 1:** Simplex Algorithm (cf. [NW99])

```
Input: $\mathbb{B}$, $\bar{\mathbb{B}}$, $\mathbf{k}$, $\mathbf{A}$, $\mathbf{b}$
compute d using $\mathbf{d}_B \overset{(3.26)}{=} \mathbf{B}^{-1}\mathbf{b}$ and $\mathbf{d}_N \overset{(3.25)}{=} \mathbf{0}$    1
repeat                                                                                    2
    build matrices $\mathbf{B}$, $\mathbf{N}$ and vectors $\mathbf{k}_B$, $\mathbf{k}_N$ from $\mathbb{B}$, $\bar{\mathbb{B}}$, $\mathbf{A}$, and $\mathbf{k}$    3
    compute $\boldsymbol{\mu}_N$ using (3.24)                                             4
    if $\boldsymbol{\mu}_N \geq \mathbf{0}$ then // optimal solution found               5
        return d                                                                          6
    else                                                                                  7
        choose $n \in \bar{\mathbb{B}}$ such that corresponding Lagrange multiplier $\mu_n < 0$    8
        set $\mathbf{a}_n$ = $n$-th column of $\mathbf{A}$                                9
        compute $\boldsymbol{\delta} = \mathbf{B}^{-1}\mathbf{a}_n$                       10
        set $d^* = \min_i \frac{d_{B,i}}{\delta_i}$ s.t. $\delta_i > 0$; with $d_{B,i}$, $\delta_i$ is $i$-th component of $\mathbf{d}_B$ and $\boldsymbol{\delta}$,    11
            respectively, and set $p$ = index $i \in \mathbb{B}$ that corresponds to $d^* = \frac{d_{B,i}}{\delta_i}$
        set $\mathbb{B} = \{n\} \cup \mathbb{B} \setminus \{p\}$ and $\bar{\mathbb{B}} = \{p\} \cup \bar{\mathbb{B}} \setminus \{n\}$    12
        update d using $d_n = d^*$, $\boldsymbol{\delta} = \mathbf{B}^{-1}\mathbf{a}_n$, and (3.27)    13
    end                                                                                   14
until Maximum number of loops exceeded                                                    15
return d                                                                                  16
```

At the optimum $\boldsymbol{\mu}_B^T\mathbf{d}_B = 0$ must be fulfilled (cf. (3.7)). Assuming that the constraints for the bounds $\mathbf{d}_B \geq \mathbf{0}$ are inactive, $\boldsymbol{\mu}_B$ is set to be $\boldsymbol{\mu}_B = \mathbf{0}$. With $\boldsymbol{\mu}_B = \mathbf{0}$, it follows from the KKT conditions (cf. Section 3.1) that

$$\nabla_{\mathbf{d}_B}\mathcal{L}(\mathbf{d}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{k}_B - \mathbf{B}^T\boldsymbol{\lambda} \overset{!}{=} \mathbf{0} \quad \Rightarrow \quad \boldsymbol{\lambda} = \mathbf{B}^{-T}\mathbf{k}_B \tag{3.23}$$

$$\nabla_{\mathbf{d}_N}\mathcal{L}(\mathbf{d}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{k}_N - \mathbf{N}^T\boldsymbol{\lambda} - \boldsymbol{\mu}_N \overset{!}{=} \mathbf{0} \quad \overset{(3.23)}{\Rightarrow} \quad \boldsymbol{\mu}_N = \mathbf{k}_N - \left(\mathbf{B}^{-1}\mathbf{N}\right)^T\mathbf{k}_B \tag{3.24}$$

$$\mathbf{d}_N \overset{!}{=} \mathbf{0} \tag{3.25}$$

$$\mathbf{B}\mathbf{d}_B + \mathbf{N}\mathbf{d}_N - \mathbf{b} \overset{!}{=} \mathbf{0} \quad \overset{(3.25)}{\Rightarrow} \quad \mathbf{d}_B = \mathbf{B}^{-1}\mathbf{b} \tag{3.26}$$

and that $\boldsymbol{\mu}_N \geq \mathbf{0}$ if the basic feasible point $\mathbf{d}^T = [\mathbf{d}_B^T, \mathbf{d}_N^T]$ is optimal. I.e., $\boldsymbol{\mu}_N \geq \mathbf{0}$ implies that the choice $\boldsymbol{\mu}_B = \mathbf{0}$ was correct and that a solution of the problem has been found. Otherwise, i.e., if some $\boldsymbol{\mu}_N < \mathbf{0}$, there is a different basis set $\mathbb{B}$ which defines a better solution for the optimization problem. Such a point can be found by exchanging basis and non-basis variables. The simplex algorithm defines a systematic way to perform this exchange. A simplified algorithm is given in Algorithm 1.

The algorithm requires a basic set – i.e., a basic feasible point is given initially – and a description of the optimization problem as an input. In each step, the computations derived in (3.23) and (3.24) are used to decide if the solution is optimal (line 5). If the point is not optimal, the algorithm improves the solution (lines 8 to 13).

For this purpose, a non-basic variable is chosen (line 8) which corresponds to a negative Lagrangian multiplier. This choice is not only intuitively reasonable – because all Lagrangian multipliers must be greater or equal to zero at the optimum – but it can also be shown that this selection results in an improvement of the objective function [NW99]. Several heuristics can be used to decide which parameter should be chosen if more than one Lagrangian multiplier is negative (e.g., [Tho94]). Within this thesis, the parameter which corresponds to the smallest component of $\boldsymbol{\mu}$, i.e., to the most sensitive constraint, is used. This method is suggested, e.g., in [Dan66] and referred to as *Dantzig's rule* or *Non-Basic Gradient Method* [Tho94].

Using the newly selected variable, a basis variable is computed which should become a non-basis variable (line 10 and 11): Defining $\mathbf{B}^{(m)}$ as the matrix of basic columns in iteration $m$, $\mathbf{d}_{B,m}^{(m)}$ and $\mathbf{d}_{B,m}^{(m+1)}$ as the basis variables from iteration $m$ evaluated in iteration $m$ and $m+1$, $d_n^{(m+1)}$ as the new basis variable in iteration $m+1$, and $\mathbf{a}_n$ as the column of $\mathbf{A}$ that corresponds to $d_n^{(m+1)}$,

$$\mathbf{B}^{(m)}\mathbf{d}_{B,m}^{(m)} = \mathbf{B}^{(m)}\mathbf{d}_{B,m}^{(m+1)} + \mathbf{a}_n d_n^{(m+1)} = \mathbf{b} \Rightarrow \mathbf{d}_{B,m}^{(m)} = \mathbf{d}_{B,m}^{(m+1)} + {\mathbf{B}^{(m)}}^{-1}\mathbf{a}_n \cdot d_n^{(m+1)} \quad (3.27)$$

must be fulfilled (cf. (3.26)). I.e., the influence of an increase of $d_n^{(m+1)}$ on each component of $\mathbf{d}_{B,m}^{(m)}$ can be determined by

$$\boldsymbol{\delta} = {\mathbf{B}^{(m)}}^{-1}\mathbf{a}_n \quad (3.28)$$

A component $\delta_i \leq 0$ of $\boldsymbol{\delta}$ implies that the value of a basis variable is not decreased if $d_n^{(m+1)}$ is increased and that the variable can not become a non-basis variable. As a consequence, $d_n^{(m+1)}$ can be increased until the first basis variable $d_{B,i}^{(m+1)}$ which belongs to a positive component of $\boldsymbol{\delta}$ is zero. Considering (3.27) line by line and setting the corresponding component of the vector of basis variables $\mathbf{d}_{B,m}^{(m+1)}$ in iteration $m+1$ to $d_{B,i}^{(m+1)} = 0$, the increase in parameter $d_n^{(m+1)}$ to reach $d_{B,i}^{(m+1)} = 0$ can be computed as

$$d_{B,i}^{(m)} = \delta_i \cdot d_n^{(m+1)} \Leftrightarrow d_n^{(m+1)} = \frac{d_{B,i}^{(m)}}{\delta_i} \quad (3.29)$$

To ensure that all variables are positive after the iteration, as required in (3.16), the basis variable $d_{B,i}^{(m+1)}$ which belongs to the smallest positive value of $d_n^{(m+1)}$ is selected as new non-basis variable and the value of the new basis variable $d_n^{(m+1)}$ can be computed by (cf. line 11 of Algorithm 1)

$$d_n^{(m+1)} = \min_i \frac{d_{B,i}^{(m)}}{\delta_i} \text{ s.t. } \delta_i > 0 \quad (3.30)$$

$d_{B,i}$ and $\delta_i$ is $i$-th component of $\mathbf{d}_B$ and $\boldsymbol{\delta}$, respectively.

After selecting the basis and non-basis variable, the basis and non-basis set are computed and the basis feasible point is updated (lines 12 and 13). The loop is repeated until either a solution has been found (line 6) or the maximum number of loops is reached (line 15).

The algorithm presented so far is not computationally efficient because matrix inversions are required in lines 4 and 10. A more efficient way can be demonstrated using the simplex tableau.

The simplex tableau can be established by writing (3.21) in the form

$$\begin{array}{c:c|c} \mathbf{d}_B^T & \mathbf{d}_N^T & \\ \hline \mathbf{B} & \mathbf{N} & \mathbf{b} \\ \hline \mathbf{k}_B^T & \mathbf{k}_N^T & 0 \end{array} \tag{3.31}$$

The columns which correspond to $\mathbf{d}_B$ and $\mathbf{d}_N$ are referred to as basis and non-basis columns, respectively. Applying a block-wise *Gaussian elimination* by multiplying the second row by $\mathbf{B}^{-1}$ from the left and forcing the elements at the position of $\mathbf{k}_B^T$ in the last row (referred to as objective function row) to be $\mathbf{0}^T$ yields

$$\begin{array}{c:c|c} \mathbf{d}_B^T & \mathbf{d}_N^T & \\ \hline \mathbf{I} & \mathbf{B}^{-1}\mathbf{N} & \mathbf{B}^{-1}\mathbf{b} \overset{(3.26)}{=} \mathbf{d}_B \\ \hline \mathbf{0}^T & \mathbf{k}_N^T - \mathbf{k}_B^T(\mathbf{B}^{-1}\mathbf{N}) \overset{(3.24)}{=} \boldsymbol{\mu}_N^T & -\mathbf{k}_B^T\mathbf{d}_B \end{array} \tag{3.32}$$

Line 8 of Algorithm 1 can now be interpreted as choosing a negative entry in the objective function row. The corresponding column is referred to as the pivot column.

Each non-basis column corresponds to a vector $\boldsymbol{\delta} = \mathbf{B}^{-1}\mathbf{a}$. Hence, line 11 can be interpreted as the task of finding the minimum positive quotient of the value on the right-hand side divided by the value in the pivot column. The corresponding row is referred to as the pivot row. Exchanging a basis and a non-basis variable can be realized by a step of the Gaussian elimination, i.e., the equation system is transformed in such a way that the pivot column is a unity vector with a single one in the pivot row. As a consequence, it is necessary to execute only a Gaussian elimination step instead of the two matrix inversions in Algorithm 1.

For the simplex algorithm presented so far, an initial basis feasible solution must be given. However, the simplex algorithm can be used again to find such a solution. For this purpose, the problem to find a feasible point can be formulated analogously to (3.16) as

$$\min_{\boldsymbol{\chi},\mathbf{d}} \boldsymbol{\chi} \text{ s.t.} \quad \begin{aligned} \mathbf{A}\cdot\mathbf{d} + \boldsymbol{\chi} &= \mathbf{b} \\ \boldsymbol{\chi} &\geq \mathbf{0} \end{aligned} \tag{3.33}$$

Initially $\boldsymbol{\chi} = \mathbf{b} - \mathbf{A}\cdot\mathbf{d}_0 \geq \mathbf{0}$ is the violation of the constraints at a starting point $\mathbf{d}_0$. The solution of this optimization problem is a basis feasible solution of (3.16). I.e., after solving (3.33), the last line of the simplex tableau can be replaced by the objective function of (3.16) and the simplex algorithm can be applied after a transformation which ensures that the objective function row is zero for all basis variables. The procedure of computing a basic feasible solution and of solving the equation system afterward is known as the *two phases simplex algorithm*.

## 3.3 Quadratic Programming in a Continuous Domain

### 3.3.1 Quadratic Programming Task

A quadratic programming task $(QP)$ refers to an optimization problem with quadratic objective function and linear constraints:

$$\min_{\mathbf{d}} \tfrac{1}{2}\mathbf{d}^T\mathbf{H}\mathbf{d} + \mathbf{g}^T\mathbf{d} \quad \text{s.t.} \quad \begin{array}{l} \mathbf{a}_l^T\mathbf{d} \geq b_l; \text{ for } l \in \mathcal{I} \\ \mathbf{a}_l^T\mathbf{d} = b_l; \text{ for } l \in \mathcal{E} \end{array} \tag{3.34}$$

$\mathbf{H}$ is referred to as Hessian matrix, the vector $\mathbf{g}$ describes the linear part of the objective function. The constraints – defined by constant vectors $\mathbf{a}_l$ and constant values $b_l$ – define a convex feasible set (cf. (3.15) and (3.14)).

The quadratic programming problem must be solved in this thesis within the Sequential Quadratic Programming algorithm (Section 3.4.2) and within the new Branch-and-Bound approach in Chapter 4. In both cases, a strictly convex model of the Hessian matrix is used. Assuming a strictly convex Hessian matrix, there is exactly one global minimum for the quadratic optimization problem which can be computed, e.g., by the active set method presented in Section 3.3.2.

### 3.3.2 Active Set Method for Convex Quadratic Programming

Convex quadratic minimization tasks are problems of the form (3.34) with positive definite Hessian matrix $\mathbf{H}$. Such optimization problems have a unique solution which is a global optimum of the problem. This optimum can either be on the surface or inside the polytope which is defined by the constraints.

Similar to the Simplex algorithm (cf. Section 3.2.3), a major task in solving the optimization problem is to identify the set of constraints – if any – which prevent the objective function at the optimum from further improvement. Each of these linear constraints $c_i(\mathbf{d})$ fulfills $c_i(\mathbf{d}) = 0$, i.e., is active (cf. Definition 3.1). If the set of linearly independent, limiting, active constraints can be found, the unique solution for the optimization problem can be computed using the KKT conditions (cf. [NW99]).

In the following, the constraints in an active set are assumed to be linearly independent. However, in practice the claim of linear independence requires preprocessing to delete redundant constraints from the set of constraints and to receive a working set.

Assuming that the active set $\mathcal{A}\left(\mathbf{d}^{(\mu)}\right)$ at a feasible point $\mathbf{d}^{(\mu)}$ is given, the optimum of the quadratic objective function in (3.34) subject to these active constraints can be found by solving the equality constraint optimization problem

$$\begin{array}{ll} \min\limits_{\mathbf{d}^{(\mu+1)}} & \frac{1}{2}\cdot \mathbf{d}^{(\mu+1)^T}\cdot \mathbf{H}\cdot \mathbf{d}^{(\mu+1)} + \mathbf{g}^T\cdot \mathbf{d}^{(\mu+1)} \\ \text{s.t.} & \mathbf{a}_l^T\cdot \mathbf{d}^{(\mu+1)} - b_l = 0;\ \text{for } l\in \mathcal{A}\left(\mathbf{d}^{(\mu)}\right) \end{array} \tag{3.35}$$

The vectors $\mathbf{a}_l$ and the values $b_l$ which correspond to the active constraints can be collected in the matrix $\mathbf{A}^{(\mu)}$ and the vector $\mathbf{b}^{(\mu)}$, respectively. The Lagrangian function of (3.35) can be defined as

$$\begin{aligned} \mathcal{L}(\mathbf{d}^{(\mu+1)},\boldsymbol{\lambda}^{(\mu)}) &= \frac{1}{2}\cdot \mathbf{d}^{(\mu+1)^T}\cdot \mathbf{H}\cdot \mathbf{d}^{(\mu+1)} + \mathbf{g}^T\cdot \mathbf{d}^{(\mu+1)} - \sum_{l\in\mathcal{A}\left(\mathbf{d}^{(\mu)}\right)} \lambda_l^{(\mu)}\left(\mathbf{a}_l^T\mathbf{d}^{(\mu+1)} - b_l\right) \\ &= \frac{1}{2}\cdot \mathbf{d}^{(\mu+1)^T}\cdot \mathbf{H}\cdot \mathbf{d}^{(\mu+1)} + \mathbf{g}^T\cdot \mathbf{d}^{(\mu+1)} - \boldsymbol{\lambda}^{(\mu)^T}\left(\mathbf{A}^{(\mu)}\mathbf{d}^{(\mu+1)} - \mathbf{b}^{(\mu)}\right) \end{aligned} \tag{3.36}$$

The Lagrangian function can also be determined from (3.34) by setting all Lagrangian factors for inactive constraints to zero:

$$\underset{l\in(\mathcal{E}\cup\mathcal{I})\setminus\mathcal{A}\left(\mathbf{d}^{(\mu)}\right)}{\forall} \quad \lambda_l = 0 \tag{3.37}$$

From the KKT conditions (3.3) and (3.5) applied to (3.35) and (3.36) it follows that

$$\left[\begin{array}{c} \nabla_{\mathbf{d}}\mathcal{L}(\mathbf{d}^{(\mu+1)},\boldsymbol{\lambda}^{(\mu)}) \\ \hdashline \mathbf{A}^{(\mu)}\cdot\mathbf{d}^{(\mu+1)} - \mathbf{b}^{(\mu)} \end{array}\right] = \left[\begin{array}{c}\mathbf{0}\\ \hdashline \mathbf{0}\end{array}\right] \Rightarrow \left[\begin{array}{c:c} \mathbf{H} & -\mathbf{A}^{(\mu)^T} \\ \hdashline \mathbf{A}^{(\mu)} & \mathbf{0}\end{array}\right]\cdot\left[\begin{array}{c}\mathbf{d}^{(\mu+1)}\\ \hdashline \boldsymbol{\lambda}^{(\mu)}\end{array}\right] = \left[\begin{array}{c}-\mathbf{g}\\ \hdashline \mathbf{b}^{(\mu)}\end{array}\right] \tag{3.38}$$

or with $\mathbf{d}^{(\mu+1)} = \mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$

$$\left[\begin{array}{c:c} \mathbf{H} & -\mathbf{A}^{(\mu)^T} \\ \hdashline \mathbf{A}^{(\mu)} & \mathbf{0}\end{array}\right] = \left[\begin{array}{c}\Delta\mathbf{d}^{(\mu)}\\ \hdashline \boldsymbol{\lambda}^{(\mu)}\end{array}\right] = \left[\begin{array}{c}-\mathbf{g}-\mathbf{H}\mathbf{d}^{(\mu)}\\ \hdashline \mathbf{b}^{(\mu)}\end{array}\right] \tag{3.39}$$

The values of $\boldsymbol{\lambda}^{(\mu)}$, $\mathbf{d}^{(\mu+1)}$, and $\Delta\mathbf{d}^{(\mu)}$ can be computed efficiently, e.g., using the range-space method [NW99]. The result $\mathbf{d}^{(\mu+1)}$ is optimal for (3.35). In addition, it fulfills the KKT conditions (3.3), (3.5), and (3.7) for the optimization problem (3.34) if the Lagrangian factors for the inactive constraints are chosen according to (3.37). Three cases can be observed for the solution of the equation system considering the remaining KKT conditions:

1. KKT condition (3.4) is violated for the optimization problem (3.34):

$$\underset{l\in(\mathcal{E}\cup\mathcal{I})\setminus\mathcal{A}\left(\mathbf{d}^{(\mu)}\right)}{\exists} \quad \mathbf{a}_l^T\mathbf{d}^{(\mu+1)} - b_l < 0 \tag{3.40}$$

   (3.40) implies that $\mathbf{d}^{(\mu+1)}$ is no optimum of (3.34) and that the objective function at the optimum is limited by at least one additional constraint. I.e., the active set $\mathcal{A}\left(\mathbf{d}^{(\mu)}\right)$ is not complete and any constraint must be added.

2. KKT condition (3.6) is violated for (3.34):
$$\underset{l \in \mathcal{A}\left(\mathbf{d}^{(\mu)}\right) \setminus \mathcal{E}}{\exists} \quad \lambda_l < 0 \tag{3.41}$$
This implies that $\mathbf{d}^{(\mu+1)}$ is no optimum of (3.34) and that the objective function can be further improved if the constraint corresponding to $\lambda_l < 0$ is removed from (3.35). I.e., at least one index of the active set $\mathcal{A}\left(\mathbf{d}^{(\mu+1)}\right)$ – namely one which corresponds to $\lambda_l < 0$ – should be deleted from the active set.
3. All KKT conditions are fulfilled for (3.34). Due to the positive definiteness of $\mathbf{H}$, the point $\mathbf{d}^* = \mathbf{d}^{(\mu+1)}$ defines the unique minimum of (3.34) in this case.

The three observations are used to define the active set algorithm for convex quadratic programming in Algorithm 2 (cf. [NW99]) which is explained below.

Given a feasible initial solution $\mathbf{d}^{(0)}$, which can be computed, e.g., by the simplex algorithm in Section 3.2.3, the set of active constraints can be defined (line 2). Some methods to solve the equation system (3.39) allow the computation of $\Delta\mathbf{d}^{(\mu)}$ and a subsequent computation of the Lagrangian factors [NW99]. In addition, the Lagrangian factors are only required to decide if some constraint in the active set should be forced to be inactive. Therefore, as long as $\Delta\mathbf{d}^{(\mu)} \neq \mathbf{0}$ (lines 5 to 14) no values for $\boldsymbol{\lambda}^{(\mu)}$ must be computed.

However, if $\Delta\mathbf{d}^{(\mu)} \neq \mathbf{0}$, KKT condition (3.4) must be tested for problem (3.34) to ensure that each point $\mathbf{d}^{(\mu+1)} = \mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$ is feasible (lines 6 and 7). If the new point is feasible, the new solution can be accepted and the active set is left unchanged (line 12 to 14). Otherwise, a constraint which is violated at $\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$ must be added to the active set.

The equality constraint subproblem (3.35) may be unsolvable if all constraints which are violated at $\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$ are added to the active set. Thus, only that constraint is selected which is violated first in direction of $\Delta\mathbf{d}^{(\mu)}$. The bound of an inequality constraint is reached for some $\mathbf{d}^{(\mu)}$ and $\Delta\mathbf{d}^{(\mu)}$ if
$$\mathbf{a}_l^T \cdot \left(\mathbf{d}^{(\mu)} + \delta\Delta\mathbf{d}^{(\mu)}\right) - b_l = 0 \Leftrightarrow \delta = \frac{b_l - \mathbf{a}_l^T\mathbf{d}^{(\mu)}}{\mathbf{a}_l^T\Delta\mathbf{d}^{(\mu)}} \tag{3.42}$$
$\mathbf{a}_l^T\Delta\mathbf{d}^{(\mu)} \neq 0$ must be true as the considered constraints are fulfilled at $\mathbf{d}^{(\mu)}$ and violated at $\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$. Computing the value of $\delta$ by (3.42) for all constraints which are violated at $\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$, the constraint which corresponds to the smallest value of $\delta$ is the first constraint which is violated in direction of $\Delta\mathbf{d}^{(\mu)}$. For this smallest value of $\delta$, all constraints are fulfilled at $\mathbf{d}^{(\mu)} + \delta\Delta\mathbf{d}^{(\mu)}$ and the constraint which corresponds to the selected value of $\delta$ is active. The corresponding constraint index is computed in line 8 and added to the active set in line 9. As $\Delta\mathbf{d}^{(\mu)}$ is a descent direction for the objective function, the iterative solution of the optimization problem can be updated to $\mathbf{d}^{(\mu+1)} = \mathbf{d}^{(\mu)} + \delta\Delta\mathbf{d}^{(\mu)}$ (lines 10 and 11).

**Algorithm 2:** Active Set Algorithm (cf. [NW99])

```
Input: optimization problem (3.34), feasible initial solution $\mathbf{d}^{(0)}$
set $\mu = 0$                                                                          1
compute initial active set $\mathcal{A}\left(\mathbf{d}^{(\mu)}\right) = \mathcal{E} \cup \left\{l \in \mathcal{I} \,\middle|\, \mathbf{a}_l \mathbf{d}^{(\mu)} - b_l = 0\right\}$   2
while $\mu < \mu_{max}$ do                                                             3
    compute $\Delta\mathbf{d}^{(\mu)}$ using (3.39)                                    4
    if $\Delta\mathbf{d}^{(\mu)} \neq \mathbf{0}$ then                                 5
        set $\mathcal{V} = \left\{l \,\middle|\, \mathbf{a}_l^T \cdot \left(\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}\right) - b_l < 0\right\}$   6
        if $\mathcal{V} \neq \emptyset$ then                                           7
            set $i = \arg\min_{l \in \mathcal{V}} \frac{b_l - \mathbf{a}_l^T \mathbf{d}^{(\mu)}}{\mathbf{a}_l^T \Delta\mathbf{d}^{(\mu)}}$   8
            set $\mathcal{A}\left(\mathbf{d}^{(\mu+1)}\right) = \mathcal{A}\left(\mathbf{d}^{(\mu)}\right) \cup \{i\}$   9
            set $\delta = \frac{b_i - \mathbf{a}_i^T \mathbf{d}^{(\mu)}}{\mathbf{a}_i^T \Delta\mathbf{d}^{(\mu)}}$   10
            set $\mathbf{d}^{(\mu+1)} = \mathbf{d}^{(\mu)} + \delta\Delta\mathbf{d}^{(\mu)}$   11
        else                                                                           12
            $\mathcal{A}\left(\mathbf{d}^{(\mu+1)}\right) = \mathcal{A}\left(\mathbf{d}^{(\mu)}\right)$   13
            $\mathbf{d}^{(\mu+1)} = \mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$             14
        end                                                                            15
    else                                                                               16
        compute $\boldsymbol{\lambda}^{(\mu)}$ using (3.39)                            17
        if $\exists_{l \in \mathcal{A}\left(\mathbf{d}^{(\mu)}\right) \setminus \mathcal{E}} \lambda_l^{(\mu)} < 0$ then   18
            choose $i = \arg\min_{l \in \mathcal{A}\left(\mathbf{d}^{(\mu)}\right) \setminus \mathcal{E}} \lambda_l^{(\mu)}$   19
            set $\mathbf{d}^{(\mu+1)} = \mathbf{d}^{(\mu)}$                            20
            set $\mathcal{A}\left(\mathbf{d}^{(\mu+1)}\right) = \mathcal{A}\left(\mathbf{d}^{(\mu)}\right) \setminus \{i\}$   21
        else                                                                           22
            return $\mathbf{d}^{(\mu)}$                                                23
        end                                                                            24
    end                                                                                25
    set $\mu := \mu + 1$                                                               26
end                                                                                    27
return $\mathbf{d}^{(\mu)}$                                                            28
```

If no descent direction can be computed for (3.35), i.e., $\Delta\mathbf{d}^{(\mu)} = \mathbf{0}$ (line 16), the Lagrangian multipliers are computed (line 17) and used to decide if KKT condition (3.6) is violated for problem (3.34) (line 18). In case that (3.6) is violated, the most negative

Lagrangian factor is chosen – similar to the choice in the simplex algorithm above – to be deleted from the active set (lines 19 and 21). Otherwise – i.e., if all Lagrangian factors are positive or zero – a solution of optimization problem (3.34) has been found and can be returned (line 23).

The computational efficiency of the algorithm can be further improved if the value $\Delta\mathbf{d}^{(\mu)}$ is only computed when the active set has changed, i.e., before the while loop and after line 9 and line 21. Alternatively, the algorithm can be simplified if the Lagrangian multipliers are computed in each loop by leaving out the outer if-clause (lines 5 and 16) and reordering the instructions.

## 3.4 Nonlinear Optimization using Sequential Quadratic Programming

### 3.4.1 Nonlinear Programming Task

A nonlinear programming problem is an optimization problem of the form (3.1), i.e.,

$$\min_{\mathbf{d}} \varphi(\mathbf{d}) \quad \text{s.t.} \quad \begin{aligned} c_l(\mathbf{d}) &\geq 0; \text{ for } l \in \mathcal{I} \\ c_l(\mathbf{d}) &= 0; \text{ for } l \in \mathcal{E} \end{aligned} \tag{3.43}$$

with nonlinear objective function and nonlinear constraints. The optimization task in analog sizing formulated in Chapter 2 belongs to this class of optimization problem. Within this section, the problem is considered as a problem in a continuous parameter domain.

A nonlinear optimization problem may have many local optima. One of these local optima can be computed by the Sequential Quadratic Programming (SQP) approach which is used in this thesis. The global optimum is typically not required for analog sizing tasks if the task is to fulfill given specifications. Hence, the SQP approach can be used if a good initial solution can be provided for the optimization problem and if local optimality of the solution is sufficient. This can be assumed for most analog sizing problems.

### 3.4.2 Sequential Quadratic Programming Algorithm

Sequential Quadratic Programming uses a sequence of quadratic approximations of objective function and constraints to find a local optimal solution for the optimization problem (3.43). As in the previous sections, the KKT conditions are used to motivate the algorithm. For this purpose, the Lagrangian function of the optimization problem

$$\mathcal{L}(\mathbf{d}, \boldsymbol{\lambda}) = \varphi(\mathbf{d}) - \sum_{l \in (\mathcal{E} \cup \mathcal{I})} \lambda_l c_l(\mathbf{d}) \tag{3.44}$$

is formulated. At an optimum $\mathbf{d}^*$ of (3.43), the active inequality constraints must fulfill $c_l(\mathbf{d}^*) = 0$ and the Lagrangian factor $\lambda_l^*$ for inactive inequality constraints $c_l(\mathbf{d}^*) \geq 0$ must be $\lambda_l^* = 0$ (cf. (3.7)). In addition, (3.3) and (3.5) must be fulfilled at $\mathbf{d}^*$, i.e.,

$$\nabla_{\mathbf{d}} \mathcal{L}(\mathbf{d}^*, \boldsymbol{\lambda}^*) = \nabla_{\mathbf{d}} \varphi(\mathbf{d}^*) - \sum_{l \in \mathcal{A}(\mathbf{d}^*)} \lambda_l^* \nabla_{\mathbf{d}} c_l(\mathbf{d}^*) \overset{!}{=} \mathbf{0} \tag{3.45}$$

$$c_l(\mathbf{d}^*) \overset{!}{=} 0; \quad l \in \mathcal{A}(\mathbf{d}^*) \tag{3.46}$$

Given some point $\mathbf{d}^{(\mu)}$ with a corresponding active set equal to the active set at the optimum, i.e., $\mathcal{A}\left(\mathbf{d}^{(\mu)}\right) = \mathcal{A}(\mathbf{d}^*)$, the *Newton-Raphson* approach to find the zero of a function can be formulated to solve (3.45) and (3.46). Therefore, (3.45) and (3.46) are linearized using a *Taylor approximation* and are set to zero:

$$\begin{aligned}\nabla_{\mathbf{d}} \mathcal{L}(\mathbf{d}^{(\mu+1)}, \boldsymbol{\lambda}^{(\mu+1)}) \approx \; & \nabla_{\mathbf{d}} \mathcal{L}(\mathbf{d}^{(\mu)}, \boldsymbol{\lambda}^{(\mu)}) + \nabla^2_{\mathbf{dd}} \mathcal{L}(\mathbf{d}^{(\mu)}, \boldsymbol{\lambda}^{(\mu)}) \left(\mathbf{d}^{(\mu+1)} - \mathbf{d}^{(\mu)}\right) \\ & + \nabla^2_{\mathbf{d}\boldsymbol{\lambda}} \mathcal{L}(\mathbf{d}^{(\mu)}, \boldsymbol{\lambda}^{(\mu)}) \left(\boldsymbol{\lambda}^{(\mu+1)} - \boldsymbol{\lambda}^{(\mu)}\right) \overset{!}{=} \mathbf{0}\end{aligned} \tag{3.47}$$

$$c_l(\mathbf{d}^{(\mu+1)}) \approx \; c_l(\mathbf{d}^{(\mu)}) + \nabla_{\mathbf{d}} c_l(\mathbf{d}^{(\mu)}) \left(\mathbf{d}^{(\mu+1)} - \mathbf{d}^{(\mu)}\right) \overset{!}{=} 0; \quad l \in \mathcal{A}\left(\mathbf{d}^{(\mu)}\right) \tag{3.48}$$

With

$$\mathbf{H}^{(\mu)} := \nabla^2_{\mathbf{dd}} \mathcal{L}(\mathbf{d}^{(\mu)}, \boldsymbol{\lambda}^{(\mu)}) = \nabla^2_{\mathbf{dd}} \varphi(\mathbf{d}^{(\mu)}) - \sum_{l \in \mathcal{A}\left(\mathbf{d}^{(\mu)}\right)} \lambda_l^{(\mu)} \nabla^2_{\mathbf{dd}} c_l(\mathbf{d}^{(\mu)}) \tag{3.49}$$

setting $\Delta\mathbf{d}^{(\mu)} = \mathbf{d}^{(\mu+1)} - \mathbf{d}^{(\mu)}$, and collecting the vectors $\nabla_{\mathbf{d}} c_l(\mathbf{d}^{(\mu)})$ and the values $c_l(\mathbf{d}^{(\mu)})$ in $\mathbf{J}_c^T(\mathbf{d}^{(\mu)})$ and $\mathbf{c}(\mathbf{d}^{(\mu)})$, (3.47) and (3.48) can be rewritten as

$$\left[\begin{array}{c:c} \mathbf{H}^{(\mu)} & -\mathbf{J}_c^T(\mathbf{d}^{(\mu)}) \\ \hdashline -\mathbf{J}_c(\mathbf{d}^{(\mu)}) & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \Delta\mathbf{d}^{(\mu)} \\ \hdashline \boldsymbol{\lambda}^{(\mu+1)} \end{array}\right] \overset{!}{=} \left[\begin{array}{c} -\nabla_{\mathbf{d}} \varphi(\mathbf{d}^{(\mu)}) \\ \hdashline \mathbf{c}(\mathbf{d}^{(\mu)}) \end{array}\right] \tag{3.50}$$

However, in general the active set at the optimum is not equal to the active set at $\mathbf{d}^{(\mu)}$, i.e., $\mathcal{A}\left(\mathbf{d}^{(\mu)}\right) \neq \mathcal{A}(\mathbf{d}^*)$. Thus, the active set $\mathcal{A}\left(\mathbf{d}^{(\mu)}\right)$ must be modified for the computation of $\Delta\mathbf{d}^{(\mu)}$ to avoid negative Lagrangian factors and constraint violations in the linearization (3.47), (3.48). The modification of the active set can also be interpreted as a modification of the equation system: Considering some inactive constraints as active, i.e., $\lambda_l^{(\mu+1)} > 0$, and some active constraints as inactive, i.e., $\lambda_l^{(\mu+1)} = 0$ can be interpreted as adding and deleting variables $\lambda_l^{(\mu+1)}$ and the corresponding rows and columns from or to the equation system (3.50). At the same time $\mathbf{H}^{(\mu)}$ and $\nabla_{\mathbf{d}} \varphi(\mathbf{d}^{(\mu)})$ are constant as long as the initial point $\mathbf{d}^{(\mu)}$ is left unchanged.

As a consequence, considering a non-constant active set, the computation of $\Delta\mathbf{d}^{(\mu)}$ can be interpreted as solving a linearly constrained quadratic optimization problem (cf. Section 3.3.2):

$$\begin{aligned} \min_{\Delta\mathbf{d}^{(\mu)}} \quad & \tfrac{1}{2} \cdot \Delta\mathbf{d}^{(\mu)^T} \mathbf{H}^{(\mu)} \Delta\mathbf{d}^{(\mu)} + \nabla_{\mathbf{d}} \varphi(\mathbf{d}^{(\mu)} \Delta\mathbf{d}^{(\mu)} \\ \text{s.t.} \quad & \nabla_{\mathbf{d}} c_l(\mathbf{d}^{(\mu)}) \Delta\mathbf{d}^{(\mu)} + c_l(\mathbf{d}^{(\mu)}) \geq 0; \text{ for } l \in \mathcal{I} \\ & \nabla_{\mathbf{d}} c_l(\mathbf{d}^{(\mu)}) \Delta\mathbf{d}^{(\mu)} + c_l(\mathbf{d}^{(\mu)}) = 0; \text{ for } l \in \mathcal{E} \end{aligned} \tag{3.51}$$

**Algorithm 3:** Sequential Quadratic Programming

```
Input: optimization problem (3.43), feasible initial solution $\mathbf{d}^{(0)}$
initialize approximation of Hessian matrix $\mathbf{H}^{(0)}$                                         1
set $\mu = 0$                                                                                        2
compute gradients $\nabla_{\mathbf{d}}\varphi(\mathbf{d}^{(0)})$ and $\nabla_{\mathbf{d}}c_l(\mathbf{d}^{(0)})$, and values $c_l(\mathbf{d}^{(0)})$ for $l \in \mathcal{E} \cup \mathcal{I}$   3
while Stop criterion not fulfilled do                                                                4
    compute $\Delta\mathbf{d}^{(\mu)}$ using Algorithm 2                                              5
    compute step length $\delta$ (cf. Section 3.4.4)                                                 6
    compute $\mathbf{d}^{(\mu+1)} = \mathbf{d}^{(\mu)} + \delta\Delta\mathbf{d}^{(\mu)}$              7
    compute active set $\mathcal{A}\left(\mathbf{d}^{(\mu+1)}\right) = \mathcal{E} \cup \left\{l \in \mathcal{I} \,\middle|\, c_l(\mathbf{d}^{(\mu+1)}) = 0\right\}$   8
    compute gradients $\nabla_{\mathbf{d}}\varphi(\mathbf{d}^{(\mu+1)})$ and $\nabla_{\mathbf{d}}c_l(\mathbf{d}^{(\mu+1)})$, and values $c_l(\mathbf{d}^{(\mu+1)})$ for   9
            $l \in \mathcal{E} \cup \mathcal{I}$
    compute Lagrangian factors $\boldsymbol{\lambda}^{(\mu+1)}$ // for update of Hessian matrix      10
    update Hessian matrix $\mathbf{H}^{(\mu)} \to \mathbf{H}^{(\mu+1)}$ (cf. Section 3.4.3)          11
    set $\mu := \mu + 1$                                                                             12
end                                                                                                  13
return $\mathbf{d}^{(\mu)}$                                                                          14
```

This subproblem of the nonlinear optimization problem can be solved by the active set method in Algorithm 2 if the Hessian matrix $\mathbf{H}^{(\mu)}$ is convex. As the accurate computation of $\mathbf{H}^{(\mu)}$ is computationally too expensive and thus the Hessian matrix is approximated by a model, the convexity can be guaranteed, e.g., if the *BFGS* (Broyden Fletcher Goldfarb Shanno) approach in Section 3.4.3 is used.

The result of subproblem (3.51) is a descent direction for the optimization problem. Nevertheless, the full step $\Delta\mathbf{d}^{(\mu)}$ can result in an insufficient decrease or also an increase of the objective function due to inaccuracies in the model of $\mathbf{H}^{(\mu)}$ and due to the linearization in (3.47). Therefore, an additional effort must be made to find a good solution in the descent direction (cf. Section 3.4.4).

A simplified line search SQP algorithm is presented in Algorithm 3. Besides the active set method explained in Section 3.3.2 (line 5 of Algorithm 3), the mentioned approximation of the Hessian matrix (line 11) and the computation of a step length (line 6) are required which are explained in the next sections. In addition, the Lagrangian factors at the optimum must be estimated in each step of the SQP algorithm to approximate the Hessian matrix. For this estimate the result from the linearly constrained quadratic subproblem can be used (cf. Section 3.4.5). Other approximations (e.g., [GM79]) can also be used. However, these alternative methods require additional computational effort.

The SQP algorithm presented so far can be used for many applications. However, the descent direction computed by the SQP algorithm in this section may lead to a direction where the constraints are always violated, i.e., not only for $\mathbf{d}^{(\mu+1)} = \mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$ but also for each possible step length $\delta > 0$. In analog sizing additional problems may arise if an infeasible intermediate solution is accepted, e.g., if the simulation models of the circuit elements are not defined or if the performances behave strongly nonlinear in an infeasible domain. Hence, no infeasible intermediate solution should be chosen. Concerning this problem, a modification of the SQP algorithm, which guarantees feasibility of all intermediate solutions, is presented in Section 3.4.5.

### 3.4.3 Approximation of the Hessian Matrix

For the SQP algorithm an approximation of the Hessian matrix is required. Two methods – the *Broyden Fletcher Goldfarb Shanno (BFGS)* approach and the *symmetric rank 1 (SR1)* update – are derived in the following (cf. [NW99]). The Lagrangian multipliers are always set to the latest available values and neglected in the following notation, i.e., $\mathcal{L}(\mathbf{d}) := \mathcal{L}(\mathbf{d}, \boldsymbol{\lambda}^{(\mu+1)})$.

Given a step $\Delta\mathbf{d}^{(\mu)}$ in the parameter domain, the gradient $\nabla_{\mathbf{d}}\mathcal{L}(\mathbf{d}^{(\mu)})$ at the point $\mathbf{d}^{(\mu)}$, and the gradient $\nabla_{\mathbf{d}}\mathcal{L}(\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)})$ at $\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$, the Lagrangian function $\mathcal{L}(\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)})$ can be quadratically approximated by

$$\mathcal{L}(\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}) \approx \mathcal{L}(\mathbf{d}^{(\mu)}) + \nabla_{\mathbf{d}}\mathcal{L}(\mathbf{d}^{(\mu)})^T \Delta\mathbf{d}^{(\mu)} + \frac{1}{2}\Delta\mathbf{d}^{(\mu)^T}\mathbf{H}^{(\mu+1)}\Delta\mathbf{d}^{(\mu)} \tag{3.52}$$

The gradient for the quadratic model at $\Delta\mathbf{d}^{(\mu)}$ must fulfill

$$\nabla_{\mathbf{d}}\mathcal{L}(\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}) = \nabla_{\mathbf{d}}\mathcal{L}(\mathbf{d}^{(\mu)}) + \mathbf{H}^{(\mu+1)}\Delta\mathbf{d}^{(\mu)} \tag{3.53}$$

or, with $\Delta\boldsymbol{g}^{(\mu)} = \nabla_{\mathbf{d}}\mathcal{L}(\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}) - \nabla_{\mathbf{d}}\mathcal{L}(\mathbf{d}^{(\mu)})$:

$$\mathbf{H}^{(\mu+1)}\Delta\mathbf{d}^{(\mu)} = \Delta\boldsymbol{g}^{(\mu)} \tag{3.54}$$

Each formula to approximate the Hessian matrix iteratively should modify a given approximation $\mathbf{H}^{(\mu)}$ such that the *secant equation* (3.54) is fulfilled for the update $\mathbf{H}^{(\mu+1)}$.

**BFGS update**

For the BFGS update, the Hessian matrix – and as a consequence the inverse Hessian matrix – is assumed to be a symmetric, positive definite matrix. In each step the inverse of the Hessian matrix should be modified as little as possible subject to the constraints that the secant equation is fulfilled and the inverse Hessian matrix is still positive definite. To achieve this goal, the new inverse Hessian matrix $\mathbf{H}^{(\mu+1)^{-1}}$ must solve

$$\min_{\mathbf{H}} \left\| \mathbf{H}^{-1} - \mathbf{H}^{(\mu)^{-1}} \right\|_{\mathbf{W}} \quad \text{s.t.} \quad \begin{aligned} &\mathbf{H}^{-1} = \mathbf{H}^{-T} \\ &\mathbf{H}^{-1}\Delta\boldsymbol{g}^{(\mu)} = \Delta\mathbf{d}^{(\mu)} \end{aligned} \tag{3.55}$$

where $\|\bullet\|_{\mathbf{W}}$ is the *weighted Forbenius norm* (cf. [NW99]) with weighting matrix $\mathbf{W}$. A solution for this optimization problem for an arbitrary matrix $\mathbf{W}$ is provided in [Gre70]. Setting $\mathbf{W}$ to a matrix that fulfills $\mathbf{W}\Delta\mathbf{d}^{(\mu)} = \Delta\boldsymbol{g}^{(\mu)}$, the BFGS update formula for the inverse Hessian matrix can be derived as

$$\mathbf{H}^{(\mu+1)^{-1}} = \left(\mathbf{I} - \frac{\Delta\mathbf{d}^{(\mu)}\Delta\boldsymbol{g}^{(\mu)^T}}{\Delta\boldsymbol{g}^{(\mu)^T}\Delta\mathbf{d}^{(\mu)}}\right)\mathbf{H}^{(\mu)^{-1}}\left(\mathbf{I} - \frac{\Delta\boldsymbol{g}^{(\mu)}\Delta\mathbf{d}^{(\mu)^T}}{\Delta\boldsymbol{g}^{(\mu)^T}\Delta\mathbf{d}^{(\mu)}}\right) + \frac{\Delta\mathbf{d}^{(\mu)}\Delta\mathbf{d}^{(\mu)^T}}{\Delta\boldsymbol{g}^{(\mu)^T}\Delta\mathbf{d}^{(\mu)}} \tag{3.56}$$

where $\mathbf{I}$ is the identity matrix. This approach can be transformed to an update formula for the Hessian matrix $\mathbf{H}^{(\mu+1)}$ by applying the *Sherman-Morrison-Woodbury* formula (cf. [NW99]):

$$\mathbf{H}^{(\mu+1)} = \mathbf{H}^{(\mu)} - \frac{\mathbf{H}^{(\mu)}\Delta\mathbf{d}^{(\mu)}\Delta\mathbf{d}^{(\mu)^T}\mathbf{H}^{(\mu)}}{\Delta\mathbf{d}^{(\mu)^T}\mathbf{H}^{(\mu)}\Delta\mathbf{d}^{(\mu)}} + \frac{\Delta\boldsymbol{g}^{(\mu)}\Delta\boldsymbol{g}^{(\mu)^T}}{\Delta\boldsymbol{g}^{(\mu)^T}\Delta\mathbf{d}^{(\mu)}} \tag{3.57}$$

However, the assumption of positive definiteness of each approximation $\mathbf{H}^{(\mu)}$ implies that the *curvature condition*

$$\Delta\mathbf{d}^{(\mu)^T}\mathbf{H}^{(\mu)}\Delta\mathbf{d}^{(\mu)} = \Delta\mathbf{d}^{(\mu)^T}\Delta\boldsymbol{g}^{(\mu)} > 0 \tag{3.58}$$

must be fulfilled. This cannot be guaranteed in general if a constrained nonlinear optimization problem is to be solved by an SQP algorithm. To solve this problem, e.g., *Powell's modification* [Pow78] can be applied which suggests replacing the value of $\Delta\boldsymbol{g}$ in (3.57) and (3.56) by

$$\boldsymbol{\eta}^{(\mu)} = \Theta\Delta\boldsymbol{g}^{(\mu)} + (1-\Theta)\,\mathbf{H}^{(\mu)}\Delta\mathbf{d}^{(\mu)} \tag{3.59}$$

with

$$\Theta = \begin{cases} 1, & \Delta\mathbf{d}^{(\mu)^T}\Delta\boldsymbol{g}^{(\mu)} \geq 0.2\Delta\mathbf{d}^{(\mu)^T}\mathbf{H}^{(\mu)}\Delta\mathbf{d}^{(\mu)} \\ \frac{0.8\Delta\mathbf{d}^{(\mu)^T}\mathbf{H}^{(\mu)}\Delta\mathbf{d}^{(\mu)}}{\Delta\mathbf{d}^{(\mu)^T}\mathbf{H}^{(\mu)}\Delta\mathbf{d}^{(\mu)} - \Delta\mathbf{d}^{(\mu)^T}\Delta\boldsymbol{g}^{(\mu)}}, & \text{otherwise} \end{cases} \tag{3.60}$$

This modification guarantees that

$$\Delta\mathbf{d}^{(\mu)^T}\boldsymbol{\eta}^{(\mu)} \geq 0.2\Delta\mathbf{d}^{(\mu)^T}\mathbf{H}^{(\mu)}\Delta\mathbf{d}^{(\mu)} \tag{3.61}$$

and that $\mathbf{H}^{(\mu+1)}$ is positive definite if $\mathbf{H}^{(\mu)}$ is positive definite.

**SR1 update**

In contrast to the BFGS update, the SR1 update does not assume positive definiteness of the Hessian matrix. However, the Hessian matrix should be symmetric and must fulfill the secant equation.

A corresponding update can be derived using the outer product of a vector $\boldsymbol{v}$, i.e.,

$$\mathbf{H}^{(\mu+1)} = \mathbf{H}^{(\mu)} \pm \boldsymbol{v}\boldsymbol{v}^T \tag{3.62}$$

Substituting (3.62) into (3.54), it can be concluded that $\boldsymbol{v}$ is a multiple of $\Delta\boldsymbol{g}^{(\mu)} - \mathbf{H}^{(\mu)}\Delta\mathbf{d}^{(\mu)}$ [NW99]. Furthermore, it can be derived that the resulting update formula for the Hessian matrix is

$$\mathbf{H}^{(\mu+1)} = \mathbf{H}^{(\mu)} + \frac{\left(\Delta\boldsymbol{g}^{(\mu)} - \mathbf{H}^{(\mu)}\Delta\mathbf{d}^{(\mu)}\right)\left(\Delta\boldsymbol{g}^{(\mu)} - \mathbf{H}^{(\mu)}\Delta\mathbf{d}^{(\mu)}\right)^T}{\left(\Delta\boldsymbol{g}^{(\mu)} - \mathbf{H}^{(\mu)}\Delta\mathbf{d}^{(\mu)}\right)^T \Delta\mathbf{d}^{(\mu)}} \tag{3.63}$$

An update for the inverse matrix can also be derived using the Sherman-Morrison-Woodbury equation. However, it will not be discussed any further because it is not required in this thesis.

The approximation of the Hessian matrix derived by the SR1 update is not guaranteed to be positive definite. This is a drawback if the update is used in an SQP approach which uses line search (cf. Section 3.4.4) because the quadratic subproblem in the SQP approach can have multiple local optima if the Hessian matrix is indefinite or negative definite. However, if no positive definite model is required, the ability of the SR1 update to generate indefinite models can be seen as an advantage, allowing a more accurate model of nonlinear (possibly indefinite) functions in many cases (cf. Chapter 5).

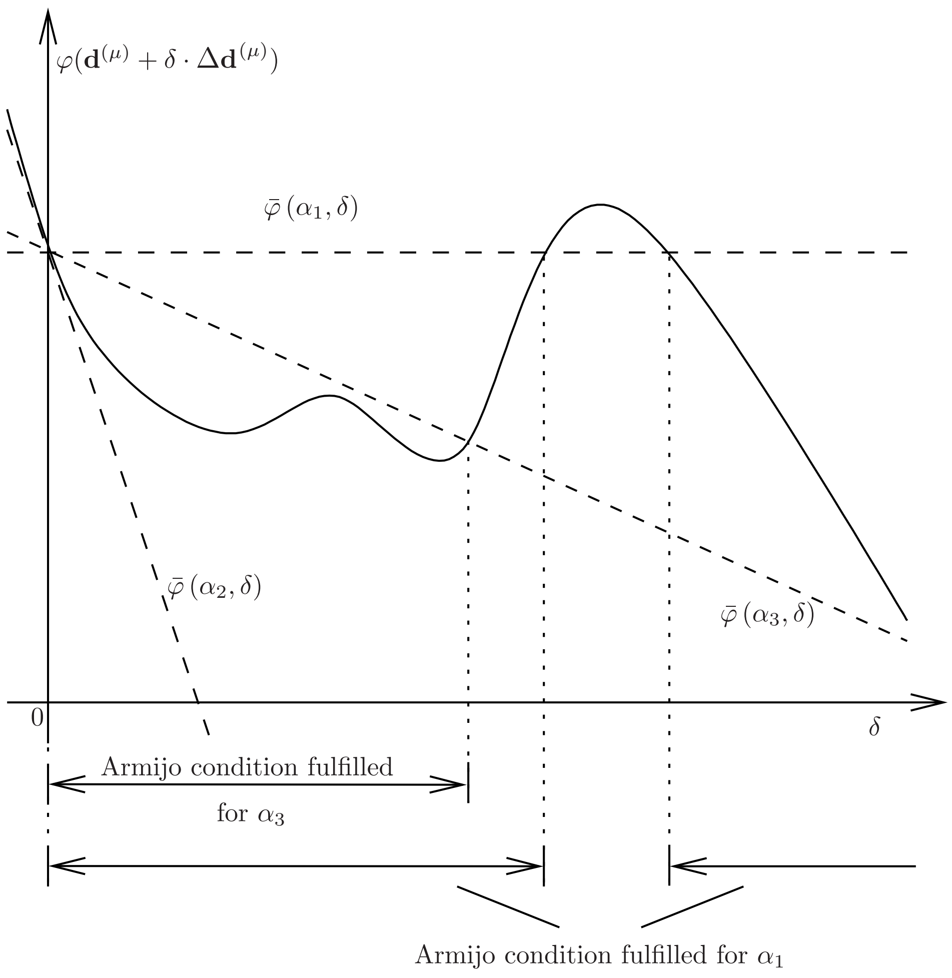
### 3.4.4 Step Computation

In each step of the SQP approach in Section 3.4.2 the optimum of a quadratic subproblem is computed. The direction to this optimum is a descent direction for the actual, nonlinear optimization problem. However, due to inaccuracies of the quadratic model, the full step does not necessarily yield sufficient improvement of the optimization task. This problem can be solved using *line search* or *trust region* approaches.

**Line Search Approaches**

Typically, the computation of the minimum in a given descent direction of a minimization problem is computationally too expensive. Therefore, line search approaches take the descent direction $\Delta\mathbf{d}^{(\mu)}$ of the optimization problem and search for a step size $\delta$ such that the newly computed point

$$\mathbf{d}^{(\mu+1)} = \mathbf{d}^{(\mu)} + \delta \cdot \Delta\mathbf{d}^{(\mu)} \tag{3.64}$$



$\varphi(\mathbf{d}^{(\mu)} + \delta \cdot \Delta\mathbf{d}^{(\mu)})$
$\bar{\varphi}(\alpha_1, \delta)$
$\bar{\varphi}(\alpha_2, \delta)$
$\bar{\varphi}(\alpha_3, \delta)$
0
$\delta$
Armijo condition fulfilled
for $\alpha_3$
Armijo condition fulfilled for $\alpha_1$

Figure 3.2: Visualization of the Armijo condition for $\alpha_1 = 0$, $\alpha_2 = 1$, and $0 < \alpha_3 < 1$, with $\bar{\varphi}(\alpha_i, \delta) = \varphi\left(\mathbf{d}^{(\mu)}\right) + \alpha_i \cdot \delta \cdot \nabla\varphi^T\left(\mathbf{d}^{(\mu)}\right) \cdot \Delta\mathbf{d}^{(\mu)}$

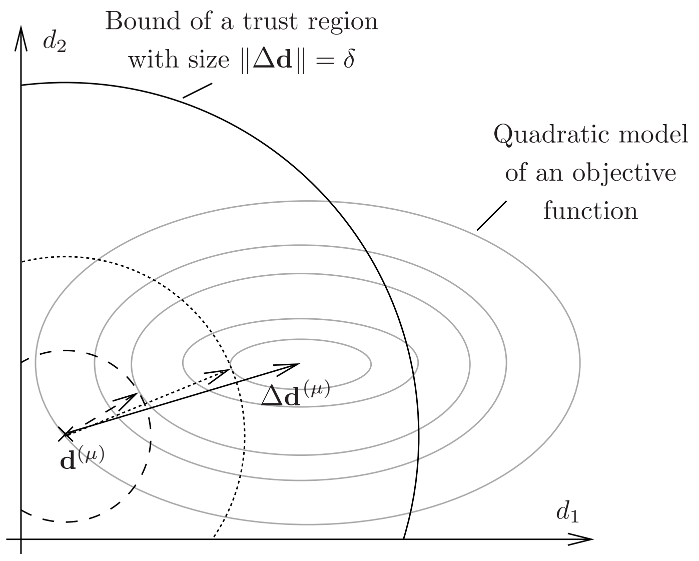
provides sufficient improvement. Such a step is typically defined as a step that fulfills the *Armijo condition*

$$\varphi\left(\mathbf{d}^{(\mu)} + \delta \cdot \Delta\mathbf{d}^{(\mu)}\right) \leq \varphi\left(\mathbf{d}^{(\mu)}\right) + \alpha \cdot \delta \cdot \nabla\varphi^T\left(\mathbf{d}^{(\mu)}\right) \cdot \Delta\mathbf{d}^{(\mu)} \tag{3.65}$$

with a small positive scalar $0 < \alpha < 1$, e.g., $\alpha = 10^{-4}$ [NW99] (cf. Figure 3.2). The Armijo condition does not guarantee a sufficiently large step $\delta$. Thus, the *Wolfe conditions* or the *Goldstein conditions* are used in many practical approaches as a criterion to identify suitable step lengths. However, within this thesis a backtracking approach (cf. [NW99] and Section 3.4.5) is used and no further criterion is required to ensure sufficiently large step lengths.

### Trust Region Approaches

Trust region approaches ensure sufficient accuracy of the model, using an additional constraint which limits the step size in each iteration step. This constraint is, e.g.,



— Trust region and computed step $\Delta\mathbf{d}^{(\mu)}$ for $\left\|\Delta\mathbf{d}^{(\mu)}\right\| \leq \delta$

····· Trust region and computed step $\Delta\mathbf{d}^{(\mu)}$ for $\left\|\Delta\mathbf{d}^{(\mu)}\right\| \leq \frac{\delta}{2}$

– - Trust region and computed step $\Delta\mathbf{d}^{(\mu)}$ for $\left\|\Delta\mathbf{d}^{(\mu)}\right\| \leq \frac{\delta}{4}$

Figure 3.3: Visualization of the influence of a trust region

defined as the Euclidean norm of the step, which should be smaller than a certain scalar $\delta > 0$ (cf. [NW99]). If this simple form of a trust region approach should be used within the SQP approach in Section 3.4.2, the quadratic subproblem (3.51) can be rewritten as:

$$\begin{aligned} \min_{\Delta\mathbf{d}^{(\mu)}} \quad & \tfrac{1}{2} \cdot \Delta\mathbf{d}^{(\mu)^T} \mathbf{H}^{(\mu)} \Delta\mathbf{d}^{(\mu)} + \nabla_{\mathbf{d}} \varphi(\mathbf{d}^{(\mu)}) \Delta\mathbf{d}^{(\mu)} \\ \text{s.t.} \quad & \nabla_{\mathbf{d}} c_l(\mathbf{d}^{(\mu)}) \Delta\mathbf{d}^{(\mu)} + c_l(\mathbf{d}^{(\mu)}) \geq 0; \text{ for } l \in \mathcal{I} \\ & \nabla_{\mathbf{d}} c_l(\mathbf{d}^{(\mu)}) \Delta\mathbf{d}^{(\mu)} + c_l(\mathbf{d}^{(\mu)}) = 0; \text{ for } l \in \mathcal{E} \\ & \left\|\Delta\mathbf{d}^{(\mu)}\right\| \leq \delta \end{aligned} \tag{3.66}$$

The size of the trust region $\delta$ is adjusted in each step of a trust-region based algorithm such that $\Delta\mathbf{d}^{(\mu)}$ is sufficiently large and the model at $\Delta\mathbf{d}^{(\mu)}$ is sufficiently accurate. Modifying the value of $\delta$ does not only change the step size – as in line search approaches – but can also influence the search direction (cf. Figure 3.3).

To decide if a model is accurate enough, the value of the model for a given step size is compared with the actual value of the problem at this point. Defining the quadratic model in (3.66) as $m(\Delta\mathbf{d}^{(\mu)})$ and the real function value as $\varphi(\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)})$, a measure for the similarity $\psi$ of model and reality can be formulated as (cf. [NW99])

$$\psi = \frac{\varphi\left(\mathbf{d}^{(\mu)}\right) - \varphi\left(\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}\right)}{m(\mathbf{0}) - m\left(\Delta\mathbf{d}^{(\mu)}\right)} \tag{3.67}$$

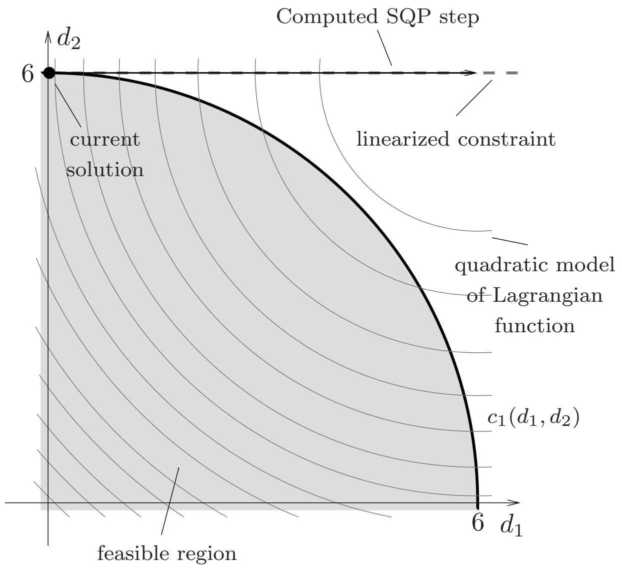


Figure 3.4: Infeasible descent direction in an SQP algorithm (cf. Appendix A.3.1)

Where $\psi \approx 1$ indicates a high similarity of model and real function, $\psi \ll 1$ indicates an insufficient model, and $\psi > 1$ indicates a model which overestimates the real function. Therefore, in case of $\psi \ll 1$ the size of the trust region $\delta$ in (3.66) must be decreased. If $\psi \approx 1$ or $\psi > 1$ the trust region can be increased.

### 3.4.5 Feasible Sequential Quadratic Programming

The SQP approach presented so far can solve an optimization problem only if the constraints can be violated at intermediate solutions of the optimization run. Otherwise, the algorithm may get stuck at suboptimal points (cf. Figure 3.4). However, for the analog sizing task the objective function is often strongly nonlinear if the sizing constraints are violated, e.g., if some transistors are in the sub-threshold region. As a consequence, the SQP approach can end up with solutions that violate some performance specifications and even constraints. To overcome this problem, a Feasible Sequential Quadratic Programming (FSQP) [LT01,Law98] approach is used within this thesis. This approach guarantees that each intermediate solution of the optimization run is feasible. The presented algorithm does not consider equality constraints, which are, however, not required to model the constraints of analog sizing tasks formulated in Chapter 2.2.

The FSQP algorithm is presented in Algorithms 4 and 5 and explained in the following. The values used for the heuristic parameters in this thesis are directly given in the Algorithms. The values are defined as suggested from [LT01] and were evaluated and validated for usage in analog sizing. Alternative value ranges are provided in [LT01].

**Algorithm 4:** Feasible Sequential Quadratic Programming (cf. [LT01])

```
Input: optimization problem (3.43), feasible initial solution $\mathbf{d}^{(0)}$
set $\mu = 0$, $\alpha = 0.1$, $\beta = 0.5$, $\tau = 2.5$, $\epsilon = 10^{-6}$, $\hat{\epsilon} = 10^{-2}$ (cf. [LT01])          1
get $\mathbf{c}(\mathbf{d})$ and sensitivities of constraints and performances $\mathbf{J}_c$ and $\mathbf{J}_\varepsilon$ at $\mathbf{d} = \mathbf{d}^{(0)}$          2
initialize Hessian matrix $\mathbf{H}^{(0)}$ and gradient $\mathbf{g}^{(0)}$ of Lagrangian function          3
initialize tilting factor $\eta_l = \hat{\epsilon}^2$, $C_l = 0$ for linear constraints, and $C_l = 1$ for          4
           nonlinear constraints with $l = 1, ..., N_c$
repeat          5
    compute $\Delta\mathbf{d}$, $\gamma$, $\boldsymbol{\lambda}$, and active set $\mathcal{A}$ by solving (3.68) with Algorithm 2          6
    if $\|\Delta\mathbf{d}\| \le \epsilon$ then          7
        break          8
    else if for all $l \in \mathcal{I}$: $c_l(\mathbf{d}^{(\mu)} + \Delta\mathbf{d}) \ge 0$ then          9
        set correction $\widehat{\Delta\mathbf{d}} = \mathbf{0}$          10
    else          11
        compute correction $\widehat{\Delta\mathbf{d}}$ by solving (3.70) with Algorithm 2          12
        if $\left\|\widehat{\Delta\mathbf{d}}\right\| \ge \|\Delta\mathbf{d}\|$ then set correction $\widehat{\Delta\mathbf{d}} = \mathbf{0}$          13
    end          14
    set $i = 0$; set of violated constraints $\mathcal{W} = \emptyset$          15
    repeat          16
        //arc-search
        set $\delta = \beta^i$; $i = i + 1$          17
        set $\mathcal{W} = \mathcal{W} \cup \left\{ l \,\middle|\, c_l\left(\mathbf{d}^{(\mu)} + \delta\Delta\mathbf{d} + \delta^2\widehat{\Delta\mathbf{d}}\right) < 0 \right\}$          18
    until $\varphi\left(\mathbf{d}^{(\mu)} + \delta\Delta\mathbf{d} + \delta^2\widehat{\Delta\mathbf{d}}\right) \le \varphi\left(\mathbf{d}^{(\mu)}\right) + \alpha \cdot \delta \cdot \nabla_{\mathbf{d}}\varphi\left(\mathbf{d}^{(\mu)}\right)^T \Delta\mathbf{d}$          19
        $\wedge\, c_l\left(\mathbf{d}^{(\mu)} + \delta\Delta\mathbf{d} + \delta^2\widehat{\Delta\mathbf{d}}\right) \ge 0$ for $l \in \mathcal{I}$
    set $\mathbf{d}^{(\mu+1)} = \mathbf{d}^{(\mu)} + \delta\Delta\mathbf{d} + \delta^2\widehat{\Delta\mathbf{d}}$          20
    get $\mathbf{c}(\mathbf{d})$ and sensitivities of constraints $\mathbf{J}_c$ and performances $\mathbf{J}_\varepsilon$ at $\mathbf{d} = \mathbf{d}^{(\mu+1)}$          21
    compute $\mathbf{g}^{(\mu+1)}$          22
    compute $\mathbf{H}^{(\mu+1)}$ using (3.57) and (3.59)          23
    get values $C_l$, $\eta_l$ for $l \in \mathcal{I}$ using Algorithm 5          24
    set $\mu := \mu + 1$          25
until maximum number of iterations exceeded          26
return $\mathbf{d}^{(\mu)}$          27
```

## FSQP Algorithm

After the initialization of the FSQP algorithm (Algorithm 4, lines 1 to 4) a descent direction for the optimization problem is computed in each step of the algorithm. In

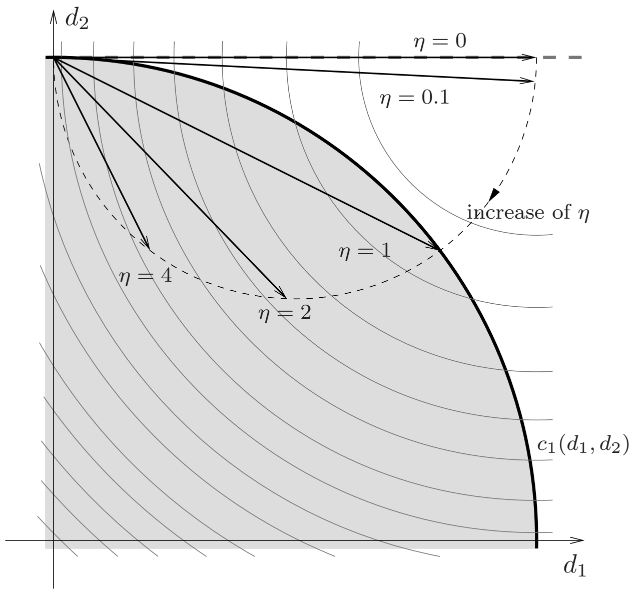


Figure 3.5: Effect of tilting for the example in Figure 3.4 (cf. Appendix A.3.1)

contrast to a standard SQP algorithm – which solves the subproblem (3.51) – the descent direction (line 6) is computed using

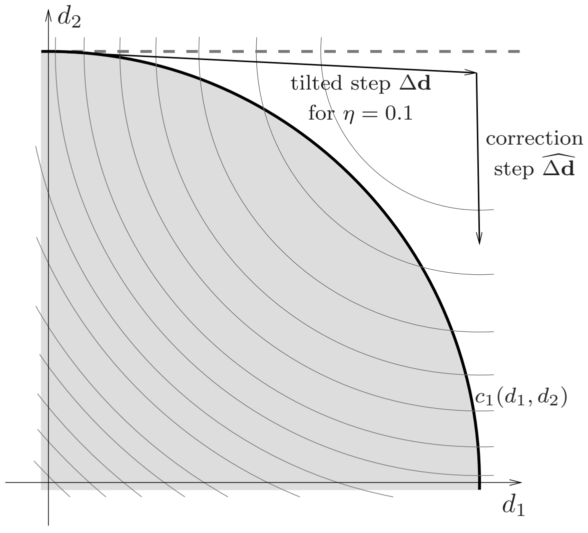
$$\begin{aligned} \min_{\Delta\mathbf{d},\gamma} \quad & \tfrac{1}{2} \cdot \Delta\mathbf{d}^T \mathbf{H}^{(\mu)} \Delta\mathbf{d} + \gamma \\ \text{s.t.} \quad & \gamma - \nabla_{\mathbf{d}} \varphi(\mathbf{d}^{(\mu)}) \cdot \Delta\mathbf{d} \geq 0 \\ & \gamma \cdot \eta_l + \nabla_{\mathbf{d}} c_l(\mathbf{d}^{(\mu)}) \cdot \Delta\mathbf{d} + c_l(\mathbf{d}^{(\mu)}) \geq 0; \text{ for } l \in \mathcal{I} \end{aligned} \tag{3.68}$$

I.e., the linear term of the objective function in the quadratic subproblem is substituted by an additional parameter $\gamma$ which is bound to be greater than the linear term. Due to the minimization $\gamma \leq 0$ is fulfilled if $\mathbf{H}^{(\mu)}$ is positive definite and if the current point, i.e, a step with size $\Delta\mathbf{d} = \mathbf{0}$, is feasible [Law98]. As a consequence, the resulting direction $\Delta\mathbf{d}$ is a descent direction for the minimization problem.

Compared to the quadratic subproblem of a standard SQP approach (3.51), the constraints are modified in (3.68) by adding the value of $\gamma$ weighted by the individual *tilting factors* $\eta_l$ with $l \in \mathcal{I}$. Choosing $\eta_l > 0$, i.e., $\gamma \cdot \eta_l \leq 0$, the tilting factor can be interpreted as an additional safety margin to avoid a constraint violation (cf. Figure 3.5). The values $\eta_l$ are dynamically assigned during the optimization algorithm (line 24) using Algorithm 5 which is explained below.

Besides the step direction $\Delta\mathbf{d}$, an approximation for the active set $\mathcal{A}$ and for the Lagrange multipliers $\boldsymbol{\lambda}$ at the optimum is computed if (3.68) is solved. Considering the Lagrangian multiplier $\lambda_0$ as the multiplier for the additional constraint in (3.68) – i.e., the multiplier corresponding to $\gamma - \nabla_{\mathbf{d}} \varphi(\mathbf{d}^{(\mu)}) \cdot \Delta\mathbf{d} \geq 0$ – the approximation of the Lagrangian multipliers can be further improved by setting (cf. [LT01])

$$\boldsymbol{\lambda} = \frac{1}{\lambda_0} \boldsymbol{\lambda} \tag{3.69}$$



$d_2$
tilted step $\Delta\mathbf{d}$
for $\eta = 0.1$
correction
step $\widehat{\Delta\mathbf{d}}$
$c_1(d_1, d_2)$
$d_1$

Figure 3.6: Effect of correction for the example in Figure 3.5 with $\eta = 0.1$ (cf. Appendix A.3.1)

if $\lambda_0$ is sufficiently large[1]. In this thesis the modification is used for $\lambda_0 > 10^{-2}$.

Using the formulation of (3.68) with a subsequent line search ensures that some step can be found which leads to an improvement of the objective function if such a step exists. However, the step can still be small. To increase the step size, a step correction $\widehat{\Delta\mathbf{d}}$ (cf. Figure 3.6) is computed if some constraints are violated for the full step $\Delta\mathbf{d}$ (line 12). The correction is computed by a second order correction [Law98, NW99] and is the solution of (cf. [LT01]):

$$\begin{aligned} \min_{\widehat{\Delta\mathbf{d}}} \quad & \tfrac{1}{2} \cdot \left(\Delta\mathbf{d} + \widehat{\Delta\mathbf{d}}\right)^T \cdot \mathbf{H}^{(\mu)} \cdot \left(\Delta\mathbf{d} + \widehat{\Delta\mathbf{d}}\right) + \nabla_{\mathbf{d}}\varphi(\mathbf{d}^{(\mu)})^T \cdot \left(\Delta\mathbf{d} + \widehat{\Delta\mathbf{d}}\right) \\ \text{s.t.} \quad & \nabla_{\mathbf{d}} c_l(\mathbf{d}^{(\mu)}) \cdot \widehat{\Delta\mathbf{d}} + c_l(\mathbf{d}^{(\mu)} + \Delta\mathbf{d}) \geq 0;\ l \in \mathcal{V}_L \\ & \nabla_{\mathbf{d}} c_l(\mathbf{d}^{(\mu)}) \cdot \widehat{\Delta\mathbf{d}} + c_l(\mathbf{d}^{(\mu)} + \Delta\mathbf{d}) \geq \min\left(10^{-2}\left\|\Delta\mathbf{d}\right\|, \left\|\Delta\mathbf{d}\right\|^{\tau}\right);\ l \in \mathcal{V}_{NL} \end{aligned} \qquad (3.70)$$

$\mathcal{V}_{NL}$ is the index set of nonlinear constraints which are active or violated at $\mathbf{d}^{(\mu)} + \Delta\mathbf{d}$. $\mathcal{V}_L$ is the index set of active linear constraints at $\mathbf{d}^{(\mu)} + \Delta\mathbf{d}$. The right-hand side for the nonlinear constraints in (3.70) is heuristically chosen for sufficiently large step sizes in non-linearly constrained problems. The correction is set to zero if all constraints are fulfilled for the full step $\Delta\mathbf{d}$ (line 10) or if the correction tends to dominate the resulting step (line 13).

[1]It can be deduced from the KKT conditions that at the optimum $\lambda_0 = 1$. However, the corresponding constraint must be active for the approximation and $\lambda_0$ should be sufficiently large to avoid numerical difficulties.

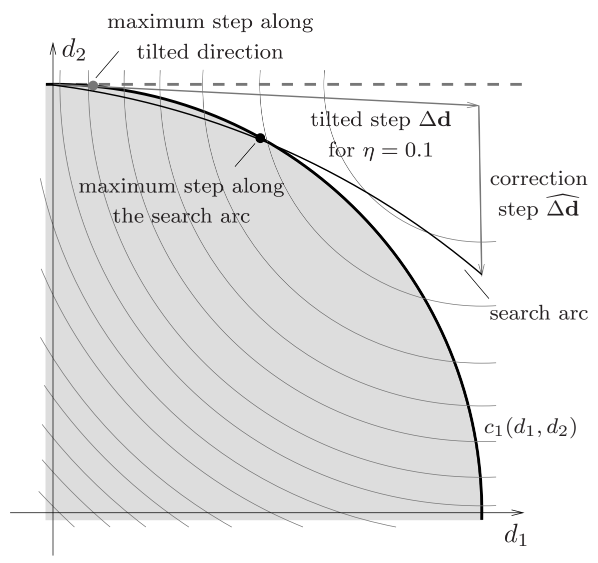


Figure 3.7: Arc search for tilted direction and correction step in Figure 3.6 with $\eta = 0.1$ (cf. Appendix A.3.1)

After the computation of a tilted direction $\Delta\mathbf{d}$ and a correction step $\widehat{\Delta\mathbf{d}}$, the new point is computed using an *arc search* (lines 15 to 19) along

$$\mathbf{d}^{(\mu)} + \delta\Delta\mathbf{d} + \delta^2\widehat{\Delta\mathbf{d}} \tag{3.71}$$

where delta is the step length along the arc (cf. Figure 3.7). This arc search is a generalization of the line search in Section 3.4.4 and searches for a point which does not only fulfill the constraints but also provides sufficient improvement (cf. (3.65)). A sufficiently large step is guaranteed by starting from the full step and shrinking this step until the first sufficient solution has been found. The set of violated constraints during the arc search $\mathcal{W}$ (line 18) is required for the reassignment of the tilting factors explained below.

After the computation of the new point (line 20) the quadratic model is updated (lines 21 to 23) and the tilting factors are reassigned (line 24). The loop is repeated, until a maximum number of iterations is reached (line 26) or the step $\Delta\mathbf{d}$ becomes too small (line 7). Further stop criteria can be added at these positions.

## Computation of Tilting Parameters

One key parameter for the efficiency of the FSQP algorithm is the size of the tilting factor $\eta_l$ for each constraint. These factors are initialized by a small value (Algorithm 4, line 4) and updated in each step as shown in Algorithm 5. The tilting factors for linear constraints are kept at zero[2].

The tilting parameters are reassigned during the FSQP run so that each value is large enough to consider the non-linearity of the corresponding constraint. At the same

[2]Tilting factors are not used, i.e., $\eta_l = 0$, for linear constraints because they are introduced to consider non-linearity.

**Algorithm 5:** Reassign Tilt (cf. [LT01])

```
Input: $\mathbf{d}^{(\mu+1)}$, quadratic subproblem at $\mathbf{d}^{(\mu+1)}$, $\Delta\mathbf{d}$, $\delta$, $\mathcal{A}$, $\mathcal{W}$, $C_l$ with $l \in \mathcal{I}$
set $\hat{\epsilon} = 10^{-2}$, $\kappa = 2$, $\underline{C} = 10^{-3}$, $\overline{C} = 10^{3}$, $\overline{D} = 10^{-3}$                         1
if $\delta < 1 \wedge \mathcal{W} \neq \emptyset$ then // $\delta < 1$ and constraint violation for any $\hat{\delta} > \delta$   2
    set $C_l = \min(\kappa \cdot C_l,\ \overline{C})$ for each $l \in \mathcal{W}$                                  3
else if $\delta < 1$ then // $\delta < 1$ due to insufficient improvement for all $\hat{\delta} > \delta$  4
    set $C_l = \max(\frac{C_l}{\kappa},\ \underline{C})$ for each $l \in \mathcal{I}$ and $c_l(\mathbf{d})$ nonlinear                5
end                                                                                6
if $\|\Delta\mathbf{d}\| < \hat{\epsilon}$ then                                                              7
    compute approximated next step $\widetilde{\Delta\mathbf{d}}$ and Lagrangian multipliers $\widetilde{\boldsymbol{\lambda}}$ using        8
            the quadratic subproblem at $\mathbf{d}^{(\mu)}$ and the active set $\mathcal{A}$,
            and solving (3.72)
    if an unique solution $\widetilde{\Delta\mathbf{d}}$, $\widetilde{\boldsymbol{\lambda}}$ exists $\wedge \left\|\widetilde{\Delta\mathbf{d}}\right\| < \overline{D} \wedge \widetilde{\boldsymbol{\lambda}} \geq 0$ then            9
        set $\eta_l = C_l \cdot \left\|\widetilde{\Delta\mathbf{d}}\right\|^2$                                         10
    else                                                                          11
        set $\eta_l = C_l \cdot \|\Delta\mathbf{d}\|^2$                                            12
    end                                                                           13
end                                                                               14
if $\|\Delta\mathbf{d}\| \geq \hat{\epsilon}$ then                                                            15
    set $\eta_l = C_l \cdot \hat{\epsilon}^2$                                                       16
end                                                                               17
return new values of $C_l$ and $\eta_l$ for $l \in \mathcal{I}$                                        18
```

time the tilting should be small enough so that it does not prevent the algorithm from converging to an optimum which is close to the constraints. To achieve this goal, the value of $\eta_l$ is controlled by a factor $C_l$ and the step length, where $C_l$ is bounded to $\underline{C} \leq C_l \leq \overline{C}$ .

In the first part of Algorithm 5 (line 1 to 6), the value of the controlling factor $C_l$ is modified. Three cases can be considered:

1. $\delta = 1$, i.e., the full step has been chosen during the arc search: This indicates that the selection of the tilting factors was good in the previous step and the controlling parameters $C_l$ are left unchanged.

2. $\delta < 1 \wedge \mathcal{W} \neq \emptyset$, i.e., the full step was not accepted during the arc search and at least one constraint has been violated for a step $\hat{\delta}$ with $\delta < \hat{\delta} \leq 1$: This indicates that the non-linearity of the constraints violated during the arc search – the indexes are stored in $\mathcal{W}$ (cf. Algorithm 4, line 18) – is underestimated. To improve the

estimate in the next iteration, the controlling factor $C_l$ for each constraint violated during the arc search is increased by a factor $\kappa$ if it does not exceed the upper bound $\overline{C}$.

3. $\delta < 1 \wedge \mathcal{W} = \emptyset$, i.e., the full step was not accepted during the arc search and no constraint has been violated for a step $\hat{\delta}$ with $\delta < \hat{\delta} \leq 1$: This indicates that the full step was not accepted due to insufficient improvement at larger step lengths $\hat{\delta}$. Better results might be found if the tilting allows solutions closer to the bounds. Thus, the controlling factor $C_l$ for each nonlinear constraint is decreased by a factor $\kappa$ as long as it does not fall below the lower bound $\underline{C}$.

After assigning the controlling factors, the new value of $\eta_l$ is computed (Algorithm 5, lines 7 to 17). As long as the step into the tilted direction $\mathbf{\Delta d}$ is sufficiently large, the tilt $\eta_l$ is directly computed as a fixed multiple of the controlling factors (line 16). However, as the value $C_l$ for nonlinear constraints is bounded from below, a fixed multiple of this value can prevent the algorithm from converging to a local optimum which is at the bounds of the feasible region. Hence, if the length of $\mathbf{\Delta d}$ falls below a certain level (line 7), the value of $\eta_l$ is chosen depending on the step size $\widetilde{\mathbf{\Delta d}}$ which is predicted for the next step by solving the least squares problem:

$$\begin{aligned} \min_{\widetilde{\mathbf{\Delta d}}} \quad & \tfrac{1}{2} \cdot \widetilde{\mathbf{\Delta d}}^T \mathbf{H}^{(\mu+1)} \widetilde{\mathbf{\Delta d}} + \nabla_{\mathbf{d}} \varphi(\mathbf{d}^{(\mu+1)})^T \cdot \widetilde{\mathbf{\Delta d}} \\ \text{s.t.} \quad & \nabla_{\mathbf{d}} c_l(\mathbf{d}^{(\mu+1)}) \cdot \widetilde{\mathbf{\Delta d}} + c_l(\mathbf{d}^{(\mu+1)}) = 0;\ l \in \mathcal{A}^{(\mu)} \end{aligned} \tag{3.72}$$

I.e., the step in the next iteration is approximated using the model for the next step $(\mu + 1)$ and the active set $\mathcal{A}^{(\mu)}$ from the last computation of the tilted step. The prediction is locally valid if none of the constraints becomes inactive in the next step. Thus, an (existing) solution for the prediction is only accepted if the predicted step $\widetilde{\mathbf{\Delta d}}$ is small, and if no considered constraint is predicted to be inactive after the step (i.e., the Lagrangian multipliers for the constraints in (3.72) are $\widetilde{\boldsymbol{\lambda}} \geq 0$). If the prediction is accepted, the value of $\eta_l$ is set proportional to the length of $\widetilde{\mathbf{\Delta d}}$ (line 10). Otherwise, the value of $\eta_l$ is set proportional to $\mathbf{\Delta d}$, which ensures a decrease of the tilting factors for the next step (line 12).

### Parallelization for Simulation-Based Optimization

The FSQP algorithm presented solves a nonlinear optimization task and guarantees that each intermediate solution is feasible. Therefore, it is an appropriate approach to solve an analog sizing task in a continuous domain However, the computation of a solution requires many computationally expensive simulations. Thus, as many simulations as possible should be executed in parallel. The new approach presented in the following is similar to the one presented in [MG09].

Circuit simulations are required for automated sizing of analog circuits whenever the circuit parameters are mapped onto performances or constraints (cf. Chapter 2). An analysis of the FSQP approach shows that simulations are required for the computation of constraints and sensitivities (lines 2 and 21 of Algorithm 4), for the computation of the correction (line 9 of Algorithm 4), and during the arc search (line 19 of Algorithm 4).

For the computation of the sensitivities the central form of finite differences is used in this thesis (cf. Chapter 2.3.2). As a consequence, $2 \cdot N$ points must be simulated where $N$ is the number of parameters considered in the FSQP approach[3, 4]. The simulations can be subdivided into different simulation types, e.g., AC, DC, and transient simulations, such that the total number of simulations sums up to $2 \cdot N \cdot N_{sim}$ where $N_{sim}$ is the number of simulation types. All these simulations can be executed in parallel and the simulations in lines 2 and 21 of Algorithm 4 can be considered as fully parallelizable.

For the computation of the correction in line 12, only one simulation is required to evaluate the constraints at the full tilted step $\Delta\mathbf{d}$. As a consequence, no parallelization is possible at this point of the algorithm.

It follows that the most critical point with respect to parallelization is the arc search. In the approach in [LT01] the points are sequentially evaluated. However, the candidate step sizes are known before the first simulation is executed. As a consequence, all required simulations could be executed in parallel. For practical approaches, however, parallelization is typically limited – especially for small and medium-sized enterprises – by the number of resources, e.g., the number of CPUs or simulator licenses. To consider this limitation for parallelization, the arc search (lines 16 to 19 of Algorithm 4) is replaced by the parallelized arc search in Algorithm 6.

In the parallel arc search algorithm, the degree of parallelization can be defined by an integral factor $N_{par}$. This value can be, e.g., the number of resources. As long as no solution has been found and the step lengths do not get too small, $N_{par}$ steps are computed (line 5) and evaluated in each iteration of the parallel arc search.

The constraints for an analog sizing task can often be evaluated much faster than other circuit properties, e.g., if only DC properties of the circuit are used as constraints (cf. [Mas10]). As a consequence, it is useful to evaluate only the constraints at a certain parameter point and to avoid the more expensive computation of the performances if any constraint is violated. This is realized in line 6, where the constraints are tested for each of the $N_{par}$ step lengths in the current iteration, and in line 7, where the performances are evaluated only for feasible step lengths. In certain cases it may also be reasonable to execute the simulations for the computation of performances and constraints in one step. However, this can be influenced by the setup for the simulator and does not affect the algorithm if known simulation results are stored.

Similar to the standard line search, each feasible point which fulfills the modified Armijo condition (3.65) is considered as a possible solution. The best solution can be taken from all of these candidates to consider as much information as possible (line 9). As a

[3]The performance and constraint values at the central point are – except for the initialization – known from the result of the previous arc search.

[4]It is assumed in this section that operating and process conditions do not vary. If this assumption is not true, the number of points which must be simulated is further increased (cf. Chapter 4.4).

**Algorithm 6:** Parallel arc search

```
Input: current point $\mathbf{d}^{(\mu)}$, tilted direction $\Delta\mathbf{d}$, correction $\widehat{\Delta\mathbf{d}}$
set $\alpha = 0.1$, $\beta = 0.5$, $\epsilon = 10^{-6}$, $i = 0$                                        1
initialize $\mathcal{W} = \emptyset$, $\delta_{result} = 0$                                          2
get parallelization factor $N_{par}$                                                 3
while $\beta^i > \epsilon \wedge \delta_{result} = 0$ do                                           4
    compute step lengths $\mathcal{T} = \{\delta_j = \beta^j \,|j = i, ..., i + N_{par} - 1\}$                5
    set set of feasible candidate step lengths                                     6
        $\mathcal{T}_c = \left\{ \delta_j \in \mathcal{T} \;\middle|\; \underset{l \in \mathcal{I}}{\forall} \; c_l \left( \mathbf{d}^{(\mu)} + \delta_j \Delta\mathbf{d} + \delta_j^2 \widehat{\Delta\mathbf{d}} \right) \geq 0 \right\}$
    set  set of possible solution step lengths                                      7
        $\mathcal{T}_\varphi = \left\{ \delta_j \in \mathcal{T}_c \middle| \varphi \left( \mathbf{d}^{(\mu)} + \delta_j \Delta\mathbf{d} + \delta_j^2 \widehat{\Delta\mathbf{d}} \right) \leq \varphi \left( \mathbf{d}^{(\mu)} \right) + \alpha \delta_j \nabla_{\mathbf{d}} \varphi \left( \mathbf{d}^{(\mu)} \right)^T \Delta\mathbf{d} \right\}$
    if $\mathcal{T}_\varphi \neq \emptyset$ then                                                    8
        set $\delta_{result} = \arg \min_{\delta_j \in \mathcal{T}_\varphi} \varphi \left( \mathbf{d}^{(\mu)} + \delta_j \Delta\mathbf{d} + \delta_j^2 \widehat{\Delta\mathbf{d}} \right)$                    9
    end                                                                            10
    set $\mathcal{W} = \mathcal{W} \cup \left\{ l \;\middle|\; \underset{\delta_j \in \mathcal{T} \wedge \delta_j > \delta_{result}}{\exists} \; c_l \left( \mathbf{d}^{(\mu)} + \delta_j \Delta\mathbf{d} + \delta_j^2 \widehat{\Delta\mathbf{d}} \right) < 0 \right\}$    11
    set $i = i + N_{par}$                                                            12
end                                                                                13
return step length $\delta_{result}$ and set of violated constraints $\mathcal{W}$                     14
```

consequence, better step lengths can be found if more step lengths are considered in one iteration, i.e., if more resources are available and $N_{par}$ is increased.

For the computation of the tilting factors, the indexes of the constraints which are violated during the arc search are required. For this purpose, the index set of these constraints is also stored during the parallel arc search (line 11).

## 3.5 Discrete Programming using Branch-and-Bound

The approaches presented so far are used to find an optimal solution in a continuous design space. However, this thesis focuses on the sizing of circuits considering discrete parameters. To solve this task, nonlinear discrete optimization is required which is tackled by a Branch-and-Bound approach (e.g., [NW88]) within this thesis.

### 3.5.1 Discrete Programming Task

It is assumed within this thesis that the discrete parameters of the sizing problem are ordered (cf. Chapter 2.1.1). It is further assumed that the first $N_d$ parameters are elements of a discrete valued domain $\mathbf{d}_d \in \mathbb{D}_d^{N_d}$ and that the subsequent $N_c$ parameters are defined on a continuous domain $\mathbf{d}_c \in \mathbb{D}_c^{N_c}$. The domain for the continuous and discrete parameters is defined as $\mathbb{D}^N = \mathbb{D}_d^{N_d} \times \mathbb{D}_c^{N_c}$ with parameter point $\mathbf{d} \in \mathbb{D}^N$. Additionally, it has been stated in Chapter 1 and Chapter 2 that the performances and constraints over the relaxed parameter domain $\mathbb{D}_{\text{rel}}^N$ (2.14) can be assumed to be continuously differentiable.

Under these assumptions, an optimization problem over the discrete domain $\mathbb{D}^N$ can be formulated similar to (3.43) as

$$\min_{\mathbf{d}_d,\mathbf{d}_c} \varphi(\mathbf{d}_d, \mathbf{d}_c) \quad \text{s.t.} \quad \begin{aligned} & c_l(\mathbf{d}_c, \mathbf{d}_d) \geq 0; \text{ for } l \in \mathcal{I} \\ & c_l(\mathbf{d}_c, \mathbf{d}_d) = 0; \text{ for } l \in \mathcal{E} \\ & \mathbf{d}_c \in \mathbb{D}_c^{N_c};\ \mathbf{d}_d \in \mathbb{D}_d^{N_d} \end{aligned} \tag{3.73}$$

$\mathcal{I}$ is the index set of inequality constraints. $\mathcal{E}$ is the index set of equality constraints, which is typically an empty set for analog sizing approaches. The relaxation of problem (3.73) can be given as

$$\min_{\mathbf{d}} \varphi(\mathbf{d}) \quad \text{s.t.} \quad \begin{aligned} & c_l(\mathbf{d}) \geq 0; \text{ for } l \in \mathcal{I} \\ & c_l(\mathbf{d}) = 0; \text{ for } l \in \mathcal{E} \\ & \mathbf{d} = \begin{bmatrix} \mathbf{d}_c \\ \mathbf{d}_d \end{bmatrix} \in \mathbb{D}_{\text{rel}}^N \end{aligned} \tag{3.74}$$

I.e., the relaxed problem is a nonlinear optimization problem on a continuous parameter domain.

### 3.5.2 Bounding

For discrete optimization problems no optimality criterion like the KKT conditions can be given to evaluate whether a point is optimal. The optimality criterion for the discrete minimization problem (3.73) can only be formulated as:

**Definition 3.8:**
A discrete point $\mathbf{d}^*$ is a minimum for (3.73) if for each feasible discrete solution $\mathbf{d} \in \mathbb{D}^N$ $\varphi(\mathbf{d}) \geq \varphi(\mathbf{d}^*)$ is fulfilled.

It follows that a solution must be compared with every other potential solution to determine whether the solution is optimal. If a sequence of solution candidates is generated during an algorithm, it is obviously sufficient only to compare each new candidate with

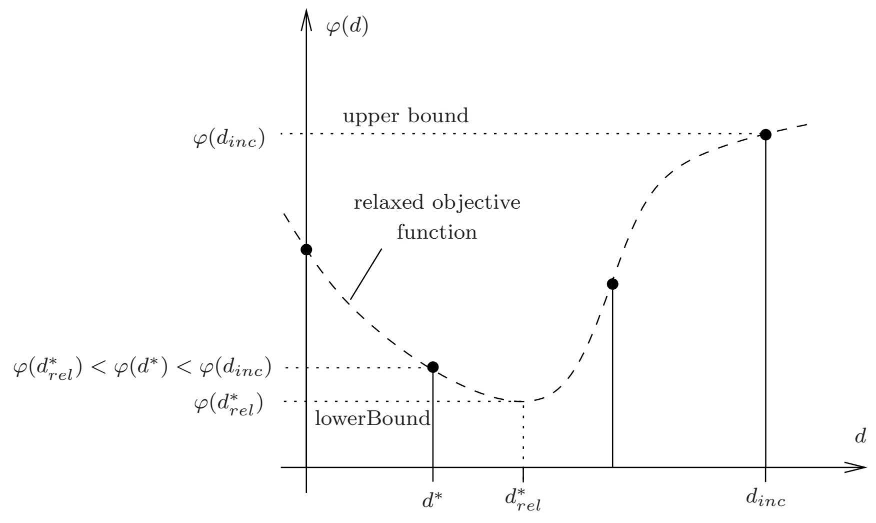


Figure 3.8: Bounds for a discrete optimum at $d^*$

the best solution found so far. In a Branch-and-Bound algorithm, this best solution is referred to as incumbent $\mathbf{d}_{\text{inc}}$.

**Definition 3.9:**
The *incumbent* $\mathbf{d}_{\text{inc}} \in \mathbb{D}^N$ is the best feasible discrete solution found so far during a Branch-and-Bound algorithm. The objective function value at $\mathbf{d}_{\text{inc}}$, i.e., $\varphi(\mathbf{d}_{\text{inc}})$, is referred to as the *incumbent value.*

It follows from the definitions above that each incumbent value $\varphi(\mathbf{d}_{\text{inc}})$ is an upper bound for the minimum value $\varphi(\mathbf{d}^*)$. In addition to this upper bound, a lower bound can be given for the optimum solution: Using the definition of the relaxed domain $\mathbb{D}^N_{\text{rel}}$, which guarantees $\mathbb{D}^N \subseteq \mathbb{D}^N_{\text{rel}}$, it can be concluded that the objective function value $\varphi(\mathbf{d}^*_{\text{rel}})$ at the optimal point $\mathbf{d}^*_{\text{rel}}$ for the relaxed optimization problem (3.74) fulfills $\varphi(\mathbf{d}^*_{\text{rel}}) \leq \varphi(\mathbf{d}^*)$. I.e., the optimal solution is bounded by

$$\varphi(\mathbf{d}^*_{\text{rel}}) \leq \varphi(\mathbf{d}^*) \leq \varphi(\mathbf{d}_{\text{inc}}) \tag{3.75}$$

for some feasible $\mathbf{d}_{\text{inc}} \in \mathbb{D}^N$ and the solution of (3.74) $\mathbf{d}^*_{\text{rel}}$ (see Figure 3.8).

The properties defined so far are also valid for any subdomain $\widetilde{\mathbb{D}}^N \subseteq \mathbb{D}^N$. I.e., given the relaxation

$$\widetilde{\mathbb{D}}^N_{\text{rel}} = \left\{ \mathbf{d} \,\middle|\, \widetilde{\mathbf{d}}_L \leq \mathbf{d} \leq \widetilde{\mathbf{d}}_U \right\} \tag{3.76}$$

of the subdomain $\widetilde{\mathbb{D}}^N$ with the component-wise upper and lower parameter bounds $\widetilde{\mathbf{d}}_L$ and $\widetilde{\mathbf{d}}_U$, respectively, the optimum of (3.74) over this relaxed subdomain is

$$\begin{aligned} \widetilde{\mathbf{d}}^*_{\text{rel}} = \arg\min_{\mathbf{d}} \varphi(\mathbf{d}) \quad \text{s.t.} \quad & c_l(\mathbf{d}) \geq 0; \text{ for } l \in \mathcal{I} \\ & c_l(\mathbf{d}) = 0; \text{ for } l \in \mathcal{E} \\ & \mathbf{d} \in \widetilde{\mathbb{D}}^N_{\text{rel}} \end{aligned} \tag{3.77}$$

and $\varphi(\widetilde{\mathbf{d}}^*_{\text{rel}})$ is a lower bound for the minimum in the discrete subdomain $\widetilde{\mathbb{D}}^N \subseteq \mathbb{D}^N$. $\widetilde{\mathbf{d}}^*_{\text{rel}}$ can be used to qualify whether the subdomain $\widetilde{\mathbb{D}}^N$ can contain a discrete solution which is better than a known discrete solution $\mathbf{d}_{\text{inc}} \in \mathbb{D}^N$: A discrete solution better than $\mathbf{d}_{\text{inc}} \in \mathbb{D}^N$ can be contained in $\widetilde{\mathbb{D}}^N$ if and only if a feasible solution $\widetilde{\mathbf{d}}^*_{\text{rel}} \in \widetilde{\mathbb{D}}^N_{\text{rel}}$ exists such that $\varphi(\widetilde{\mathbf{d}}^*_{\text{rel}}) < \varphi(\mathbf{d}_{\text{inc}})$.

If the qualification shows that $\widetilde{\mathbb{D}}^N$ does not contain a solution better than $\mathbf{d}_{\text{inc}}$, $\widetilde{\mathbb{D}}^N$ can be pruned from the parameter domain $\mathbb{D}^N$ without affecting the solution of the optimization problem. The rules to define which subdomains can be pruned – called *pruning rules* – can be formulated as:

**Pruning Rules:**

1. If $\widetilde{\mathbb{D}}^N_{\text{rel}}$ does not contain any feasible point, it follows that also $\widetilde{\mathbb{D}}^N \subseteq \widetilde{\mathbb{D}}^N_{\text{rel}}$ cannot contain a feasible solution and that the subdomain $\widetilde{\mathbb{D}}^N$ can be pruned from $\mathbb{D}^N$. This is referred to as *pruning by infeasibility*.
2. If the optimum $\widetilde{\mathbf{d}}^*_{\text{rel}} \in \widetilde{\mathbb{D}}^N_{\text{rel}}$ of a relaxed subdomain is worse than or equal to the incumbent $\mathbf{d}_{\text{inc}} \in \mathbb{D}^N$, i.e., $\varphi(\widetilde{\mathbf{d}}^*_{\text{rel}}) \geq \varphi(\mathbf{d}_{\text{inc}})$, $\widetilde{\mathbb{D}}^N \subseteq \mathbb{D}^N$ cannot contain a point better than $\mathbf{d}_{\text{inc}}$ and the subdomain $\widetilde{\mathbb{D}}^N$ can be pruned from $\mathbb{D}^N$. This is referred to as *pruning by value dominance*.
3. $\widetilde{\mathbf{d}}^*_{\text{rel}}$ is a new incumbent if $\widetilde{\mathbf{d}}^*_{\text{rel}}$ is discrete and a better solution than $\mathbf{d}_{\text{inc}}$, i.e., $\widetilde{\mathbf{d}}^*_{\text{rel}} \in \mathbb{D}^N$ and $\varphi(\widetilde{\mathbf{d}}^*_{\text{rel}}) < \varphi(\mathbf{d}_{\text{inc}})$. However, assuming that $\widetilde{\mathbf{d}}^*_{\text{rel}}$ is a global solution for the current subdomain, $\widetilde{\mathbb{D}}^N$ cannot contain any solution better than $\widetilde{\mathbf{d}}^*_{\text{rel}}$ and thus $\widetilde{\mathbb{D}}^N \setminus \left\{\widetilde{\mathbf{d}}^*_{\text{rel}}\right\}$ can be pruned from $\mathbb{D}^N$. This is referred to as *pruning by optimality*.

The pruning rules allow for a given discrete optimization problem, a given domain $\mathbb{D}^N$, and a given subdomain $\widetilde{\mathbb{D}}^N$ to qualify whether $\widetilde{\mathbb{D}}^N$ can be deleted from $\mathbb{D}^N$ without affecting the optimum. In addition, they define which discrete points must be considered as solution candidates (pruning rule 3). However, the questions of how to generate the subdomains of $\mathbb{D}^N$ and how to ensure that discrete solution candidates are generated in this process are still open.

### 3.5.3 Branching

To generate subdomains and discrete solution candidates, a given relaxed domain $\widetilde{\mathbb{D}}^N_{\text{rel}} \subseteq \mathbb{D}^N_{\text{rel}}$ is considered in each step of the Branch-and-Bound algorithm. The resulting relaxed problem is given in (3.77) and can be solved, e.g., by the FSQP algorithm in Section 3.4.5. $\widetilde{\mathbb{D}}^N_{\text{rel}}$ can contain a discrete solution better than $\mathbf{d}_{\text{inc}}$ if and only if a feasible solution $\widetilde{\mathbf{d}}^*_{\text{rel}} \in \widetilde{\mathbb{D}}^N_{\text{rel}}$ exists and $\varphi(\widetilde{\mathbf{d}}^*_{\text{rel}}) < \varphi(\mathbf{d}_{\text{inc}})$. Thus, the search in $\widetilde{\mathbb{D}}^N_{\text{rel}}$ must be refined and new subdomains of $\widetilde{\mathbb{D}}^N_{\text{rel}}$ must be generated if the domain can contain a new incumbent and if $\widetilde{\mathbf{d}}^*_{\text{rel}}$ is not discrete. Otherwise the computational effort to search for a discrete solution in $\widetilde{\mathbb{D}}^N_{\text{rel}}$ can be saved.

The subdomains which are generated should be non-overlapping and should contain the same discrete solutions as $\widetilde{\mathbb{D}}^N_{\text{rel}}$, i.e., if $\widetilde{\mathbb{D}}^N_{\text{rel}}$ is subdivided into two subdomains $\mathbb{D}^N_{\text{A}}$ and $\mathbb{D}^N_{\text{B}}$, the subdomains must fulfill:

$$\left(\left(\mathbb{D}^N_{\text{A}} \cup \mathbb{D}^N_{\text{B}}\right) \cap \mathbb{D}^N = \widetilde{\mathbb{D}}^N_{\text{rel}} \cap \mathbb{D}^N\right) \wedge \left(\mathbb{D}^N_{\text{A}} \cap \mathbb{D}^N_{\text{B}} = \emptyset\right) \tag{3.78}$$

The first part of this logic equation ensures that all discrete points in $\widetilde{\mathbb{D}}^N_{\text{rel}}$ are also considered in the subdomains; the second part guarantees that each point is considered only once in the Branch-and-Bound algorithm below. In addition to claim (3.78), the method to subdivide the relaxed domain should ensure that after a finite number of subdivisions the solution for each subdomain is discrete.

To generate such subdomains, $\widetilde{\mathbb{D}}^N_{\text{rel}}$ can be subdivided by *rounding constraints*. For rounding constraints, one component $\widetilde{d}_j$ of $\widetilde{\mathbf{d}}^*_{\text{rel}}$ is chosen which should, but does not yet take a discrete value. For this component the value is forced to be greater than the next higher discrete value or smaller than the next lower discrete value. I.e., with (2.15):

$$c_{\text{up}}(d_j) = d_j - \left\lceil \widetilde{d}_j \right\rceil \geq 0 \tag{3.79}$$

$$c_{\text{down}}(d_j) = \left\lfloor \widetilde{d}_j \right\rfloor - d_j \geq 0 \tag{3.80}$$

Two subdomains which fulfill (3.78) can be defined as

$$\mathbb{D}^N_{\text{up}} = \left\{ \mathbf{d} \in \widetilde{\mathbb{D}}^N_{\text{rel}} \middle| d_j - \left\lceil \widetilde{d}_j \right\rceil \geq 0 \right\} \tag{3.81}$$

$$\mathbb{D}^N_{\text{down}} = \left\{ \mathbf{d} \in \widetilde{\mathbb{D}}^N_{\text{rel}} \middle| \left\lfloor \widetilde{d}_j \right\rfloor - d_j \geq 0 \right\} \tag{3.82}$$

Figure 3.9 shows that the claim in (3.78) is fulfilled if the relaxed domain is subdivided according to (3.81), (3.82). It can also be seen that a strip $\left\lfloor \widetilde{d}_j \right\rfloor \leq d_j \leq \left\lceil \widetilde{d}_j \right\rceil$ is not considered in the subdomains such that after a finite number of subdivisions the result of each subdomain is discrete.

The subdomains, which are generated during a Branch-and-Bound algorithm, can also be interpreted in terms of a tree structure (cf. Section 3.5.4 Figure 3.10). The branches in this graph correspond to the rounding constraints which define the subdomains. Thus, the step of subdividing the parameter domain is referred to as *branching*. The subdomains are considered in the subproblems of the form (3.77), which correspond to the nodes of the graph. The solution of the subproblem in each node gives an approximation of the lower bound for this node and the child nodes. The generation of lower and upper bounds for the solution is also referred to as *bounding*. Pruning a subdomain from the parameter domain can be interpreted as cutting a node from the tree or – in case of pruning rule 3 – as a leaf of the tree.

### 3.5.4 Branch-and-Bound Algorithm

With the strategies explained above, a general Branch-and-Bound approach for a discrete optimization problem can be realized. In this thesis a recursive depth-first search

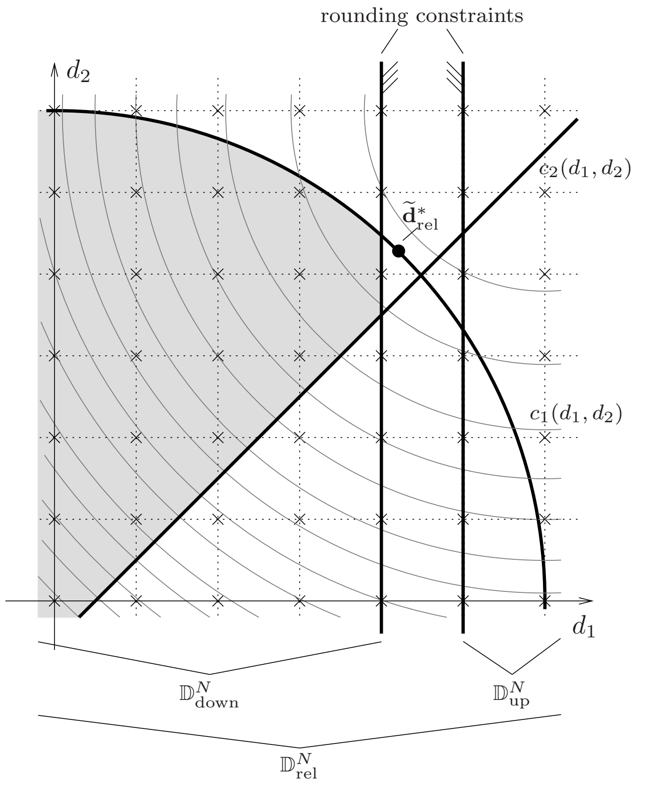

rounding constraints
$d_2$
$d_1$
$c_2(d_1, d_2)$
$c_1(d_1, d_2)$
$\widetilde{\mathbf{d}}^*_{\mathrm{rel}}$
$\mathbb{D}^N_{\mathrm{down}}$
$\mathbb{D}^N_{\mathrm{up}}$
$\mathbb{D}^N_{\mathrm{rel}}$

Figure 3.9: Non-overlapping subdomains generated by rounding constraints

is used which is shown in Algorithm 7 and illustrated in Figure 3.10. The computation strategy for a discrete solution can be interpreted as a discretization of a continuous solution using a sequence of successive branching steps. During branching, (3.81), (3.82) are used to subdivide the current domain into subdomains. Two degrees of freedom, namely the parameter which is chosen for branching (line 10) and the decision as to which subdomain is considered next (lines 12 and 13) can be identified in the algorithm. These choices do not affect the general convergence of the algorithm but can significantly influence the efficacy of the algorithm in practical cases. Some typical methods are presented in [AKM05]. In the example below, always the first non-discrete parameter is selected for branching and a discrete solution is searched for at first in the subdomain constructed by (3.82). However, the branching heuristic should be tailored to the problem considered. Thus, the heuristics used within this thesis are explained in Chapter 4 where the Branch-and-Bound algorithm is used to solve different discrete optimization problems.

The Branch-and-Bound algorithm (Algorithm 7) is started with the incumbent value $\varphi(\mathbf{d}_{\mathrm{inc}})$ set to infinity such that the first discrete feasible solution is always an incumbent. Additionally, the discrete parameter domain $\mathbb{D}^N$ and the relaxed parameter domain $\mathbb{D}^N_{\mathrm{rel}}$ are given and the relaxed domain is defined as the current subdomain $\widetilde{\mathbb{D}}^N_{\mathrm{rel}} = \mathbb{D}^N_{\mathrm{rel}}$.

In each step of the algorithm, the solution $\widetilde{\mathbf{d}}^*_{\mathrm{rel}}$ of problem (3.77) is computed for the current subdomain (cf. line 4 and Figure 3.10a), which defines a lower bound $\varphi(\widetilde{\mathbf{d}}^*_{\mathrm{rel}})$ for

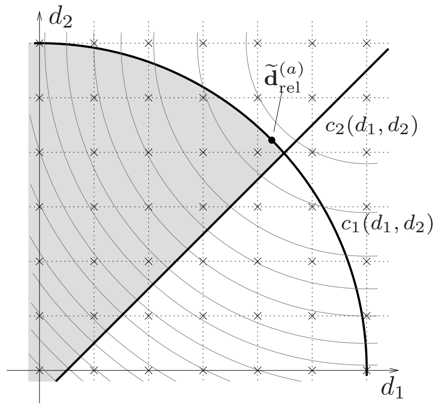
**Algorithm 7:** Branch-and-Bound

```
Input: discrete domain $\mathbb{D}^N$, current subdomain $\widetilde{\mathbb{D}}^N_{\text{rel}}$, incumbent $\mathbf{d}_{\text{inc}}$
if $\left\{\mathbf{d} \in \widetilde{\mathbb{D}}^N_{rel} \middle| \left(\underset{l \in \mathcal{E}}{\forall} c_l(\mathbf{d}) = 0\right) \wedge \left(\underset{l \in \mathcal{I}}{\forall} c_l(\mathbf{d}) \geq 0\right)\right\} = \emptyset$ then    1
    //pruning by infeasibility
    return $\mathbf{d}_{\text{inc}}$    2
end    3
compute $\widetilde{\mathbf{d}}^*_{\text{rel}}$ using (3.77)    4
if $\varphi(\widetilde{\mathbf{d}}^*_{rel}) \geq \varphi(\mathbf{d}_{inc})$ then    5
    //pruning by value dominance
    return $\mathbf{d}_{\text{inc}}$    6
else if $\widetilde{\mathbf{d}}^*_{rel} \in \mathbb{D}^N$ then    7
    //pruning by optimality
    return $\widetilde{\mathbf{d}}^*_{\text{rel}}$    8
else    9
    Select branching parameter $d_j$    10
    set $\mathbb{D}^N_{\text{up}}$, $\mathbb{D}^N_{\text{down}}$ according to (3.81) and (3.82)    11
    set $\mathbf{d}_{\text{inc}}$ = Branch-and-Bound($\mathbb{D}^N$, $\mathbb{D}^N_{down}$, $\mathbf{d}_{inc}$)    12
    set $\mathbf{d}_{\text{inc}}$ = Branch-and-Bound($\mathbb{D}^N$, $\mathbb{D}^N_{up}$, $\mathbf{d}_{inc}$)    13
    return $\mathbf{d}_{\text{inc}}$    14
end    15
```

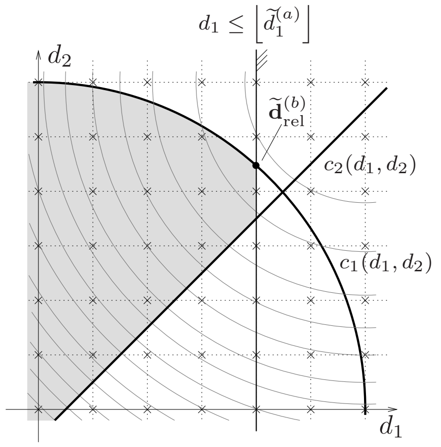
the current subproblem. If $\widetilde{\mathbf{d}}^*_{\text{rel}}$ is a feasible, non-discrete solution and if $\varphi(\widetilde{\mathbf{d}}^*_{\text{rel}})$ is better than the incumbent value, rounding constraints are used to generate new subdomains according to (3.81) and (3.82), and the Branch-and-Bound algorithm is called recursively for both subdomains (lines 9 to 14). In Figure 3.10, this happens in 3.10a and 3.10b. The subproblem in Figure 3.10b and 3.10c is generated using rounding constraint (3.80) and solved in the first and second recursion of the algorithm which is started by the function call in line 12. In Figure 3.10d and 3.10e the subdomains are generated by (3.79).

Before new subdomains are constructed, the pruning rules are applied in each recursion of the algorithm (lines 1 to 2 and 5 to 8):

1. If $\widetilde{\mathbf{d}}^*_{\text{rel}}$ is discrete and $\varphi(\widetilde{\mathbf{d}}^*_{\text{rel}})$ is the best solution as yet, $\widetilde{\mathbf{d}}^*_{\text{rel}}$ is the new incumbent (line 7 and Figure 3.10c; pruning rule 3). The assignment as new incumbent is performed in the parent recursion (lines 12 and 13).

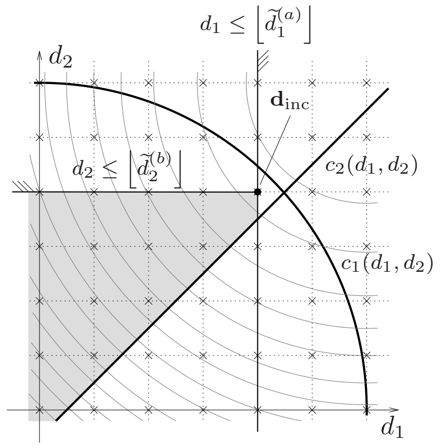


(a) Continuous solution of relaxed problem.



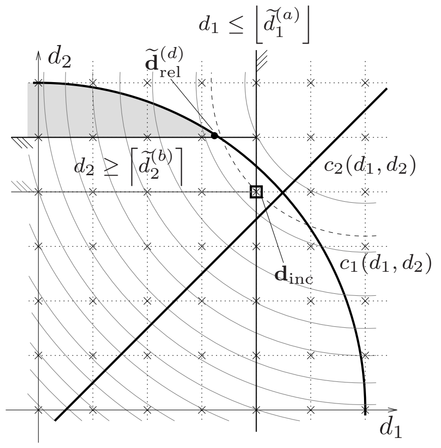
$d_1 \leq \left\lfloor \widetilde{d}_1^{(a)} \right\rfloor$
$d_2$
$\widetilde{\mathbf{d}}_{\mathrm{rel}}^{(b)}$
$c_2(d_1, d_2)$
$c_1(d_1, d_2)$
$d_1$

(b) Subproblem generated using a rounding constraint



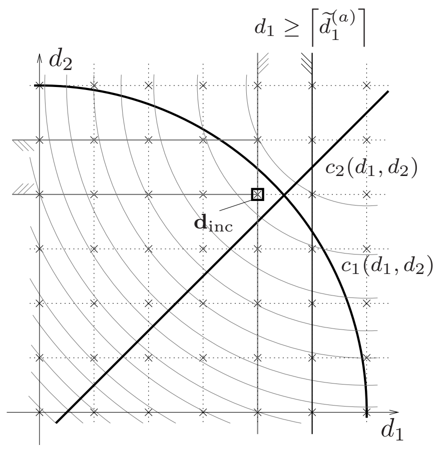
$d_1 \leq \left\lfloor \widetilde{d}_1^{(a)} \right\rfloor$
$d_2$
$\mathbf{d}_{\mathrm{inc}}$
$d_2 \leq \left\lfloor \widetilde{d}_2^{(b)} \right\rfloor$
$c_2(d_1, d_2)$
$c_1(d_1, d_2)$
$d_1$

(c) Pruning by optimality



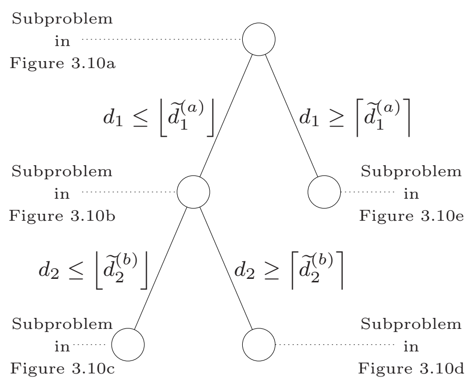
$d_1 \leq \left\lfloor \widetilde{d}_1^{(a)} \right\rfloor$
$d_2$
$\widetilde{\mathbf{d}}_{\mathrm{rel}}^{(d)}$
$d_2 \geq \left\lceil \widetilde{d}_2^{(b)} \right\rceil$
$c_2(d_1, d_2)$
$\mathbf{d}_{\mathrm{inc}}$
$c_1(d_1, d_2)$
$d_1$

(d) Pruning by value dominance



$d_1 \geq \left\lceil \widetilde{d}_1^{(a)} \right\rceil$
$d_2$
$c_2(d_1, d_2)$
$\mathbf{d}_{\mathrm{inc}}$
$c_1(d_1, d_2)$
$d_1$

(e) Pruning by infeasibility



Subproblem in Figure 3.10a
$d_1 \leq \left\lfloor \widetilde{d}_1^{(a)} \right\rfloor$
$d_1 \geq \left\lceil \widetilde{d}_1^{(a)} \right\rceil$
Subproblem in Figure 3.10b
Subproblem in Figure 3.10e
$d_2 \leq \left\lfloor \widetilde{d}_2^{(b)} \right\rfloor$
$d_2 \geq \left\lceil \widetilde{d}_2^{(b)} \right\rceil$
Subproblem in Figure 3.10c
Subproblem in Figure 3.10d

(f) Search tree of Branch-and-Bound

Figure 3.10: Visualization of a Branch-and-Bound algorithm (cf. Appendix A.3.2)

2. If the optimal point in a subdomain is feasible and not better than the incumbent, the subdomain should be pruned (pruning rule 2; Figure 3.10d). This is achieved by returning from the current recursion without modifying the incumbent in line 5.
3. Also, if the current subdomain does not include any feasible point (Figure 3.10e), the subdomain should be pruned (pruning rule 1). In this case, the algorithm returns from the current recursion in line 1. The optimal solution for the subdomain (line 4) does not have to be computed in this case.

The pruning of the current subdomain is achieved in each case by considering non-overlapping domains in each recursion (cf. (3.78)). Figure 3.10f shows the visualization of the algorithm as a branching tree. Each node in this figure corresponds to a recursion of the algorithm. In each of these nodes the optimum for the current subdomain is computed, which defines a lower bound for the discrete solution in the current and in the subsequent nodes. Each time a leaf of the tree is reached, the current subdomain is pruned.

# Chapter 4

# Branch-and-Bound for Analog Sizing

In this chapter, the algorithms described in Chapter 3 are used within a new approach to solve the discrete analog sizing problem introduced in Chapter 2. It is assumed for the new Branch-and-Bound approach for analog sizing that the circuit can be evaluated for each point in a bounded continuous domain although the final result of the sizing must be a discrete point. Typical examples for parameters in such a discrete sizing task are the lengths and widths of transistors which should be snapped to a manufacturing grid or the multipliers of CMOS transistors if BSIM [BSI11] models are used. Such a discrete sizing task can also be solved by a Branch-and-Bound approach as presented in Chapter 3.5. However, to increase the efficiency of the new approach, several innovations are applied to develop a new Branch-and-Bound approach for analog sizing.

The general structure of the new method, which consists of an outer Branch-and-Bound approach and a prediction method for the discrete solution of the problem, is explained in Section 4.1. A detailed description of the outer Branch-and-Bound approach and the new prediction method is given in Sections 4.2 and 4.3.

For the sake of a simple description, only design parameters are considered in Sections 4.1 to 4.3. Section 4.4 enlarges the approach to consider variations in operating and process parameters.

## 4.1 Structure of the New Approach

The approach for analog sizing presented in this section consists of a new simulation-based Branch-and-Bound approach – referred to as outer Branch-and-Bound – and of a new prediction method for the discrete solution. The prediction method predicts a discrete solution of the sizing task, using a non-simulation-based Branch-and-Bound approach – referred to as inner Branch-and-Bound. In this concept for analog sizing, the outer Branch-and-Bound approach ensures the (local) convergence of the algorithm while the prediction method is used to reduce the number of required simulations and, as a consequence, the required runtime of the approach significantly.

The sizing task which should be solved by the new approach can be formulated as a minimization problem according to (2.56) on Page 36

$$\min_{\mathbf{d}\in\mathbb{D}^N} \varphi_{lsq}(\mathbf{d}) \text{ s.t. } \mathbf{c}(\mathbf{d}) \geq \mathbf{0} \tag{4.1}$$

The mapping of the multi-objective problem to a scalar is realized by the truncated least squares objective function (cf. (2.47) on Page 33):

$$\varphi_{lsq}(\mathbf{d}) = \sum_{i=1}^{N_\varepsilon} \left(\eta_i \cdot \max\left(0, \varepsilon_i(\mathbf{d}) + \gamma\right)\right)^2 \tag{4.2}$$

### 4.1.1 Simulation-Based Branch-and-Bound

The outer, simulation-based Branch-and-Bound algorithm (cf. Figure 4.1) is also a new stand-alone method to solve the analog sizing task according to (4.1). It consists of a simulation-based approach for sizing in a continuous domain and of additional blocks which realize the innovations of the new Branch-and-Bound approach.

The continuous sizing approach is realized as a simulation-based FSQP approach according to Chapter 3.4.5 and ensures an efficient computation of the sizing in the continuous domain by solving a minimization problem of the form (cf. (4.1), (2.14))

$$\min_{\mathbf{d}\in\mathbb{D}^N_{\text{rel}}} \varphi_{lsq}(\mathbf{d}) \text{ s.t. } \mathbf{c}(\mathbf{d}) \geq \mathbf{0} \tag{4.3}$$

The structure of this continuous approach is identical to the structure of state-of-the-art approaches (cf. Figure 1.7 on Page 9).

The objective function value of (4.3) at the solution computed by the FSQP approach is a lower bound for the discrete solution in the currently considered continuous domain and could be used in a standard Branch-and-Bound approach to solve the analog sizing task (cf. Chapter 3.5.4). However, in such an approach many subproblems must be solved and the runtime would be impracticable high if the solution of each subproblem was computed by a simulation-based FSQP approach. In the new simulation-based Branch-and-Bound approach in Figure 4.1, the components of the Branch-and-Bound approach are improved to reduce the runtime by considering the specific characteristics of the analog sizing task:

1. The pruning rules are revised for analog sizing (Section 4.2.1), considering that the sizing task according to (2.52) is to fulfill the specifications for the performances rather than to find an optimal sizing. This leads to the new Branch-and-Bound algorithm in Algorithm 8 (Section 4.2.1).

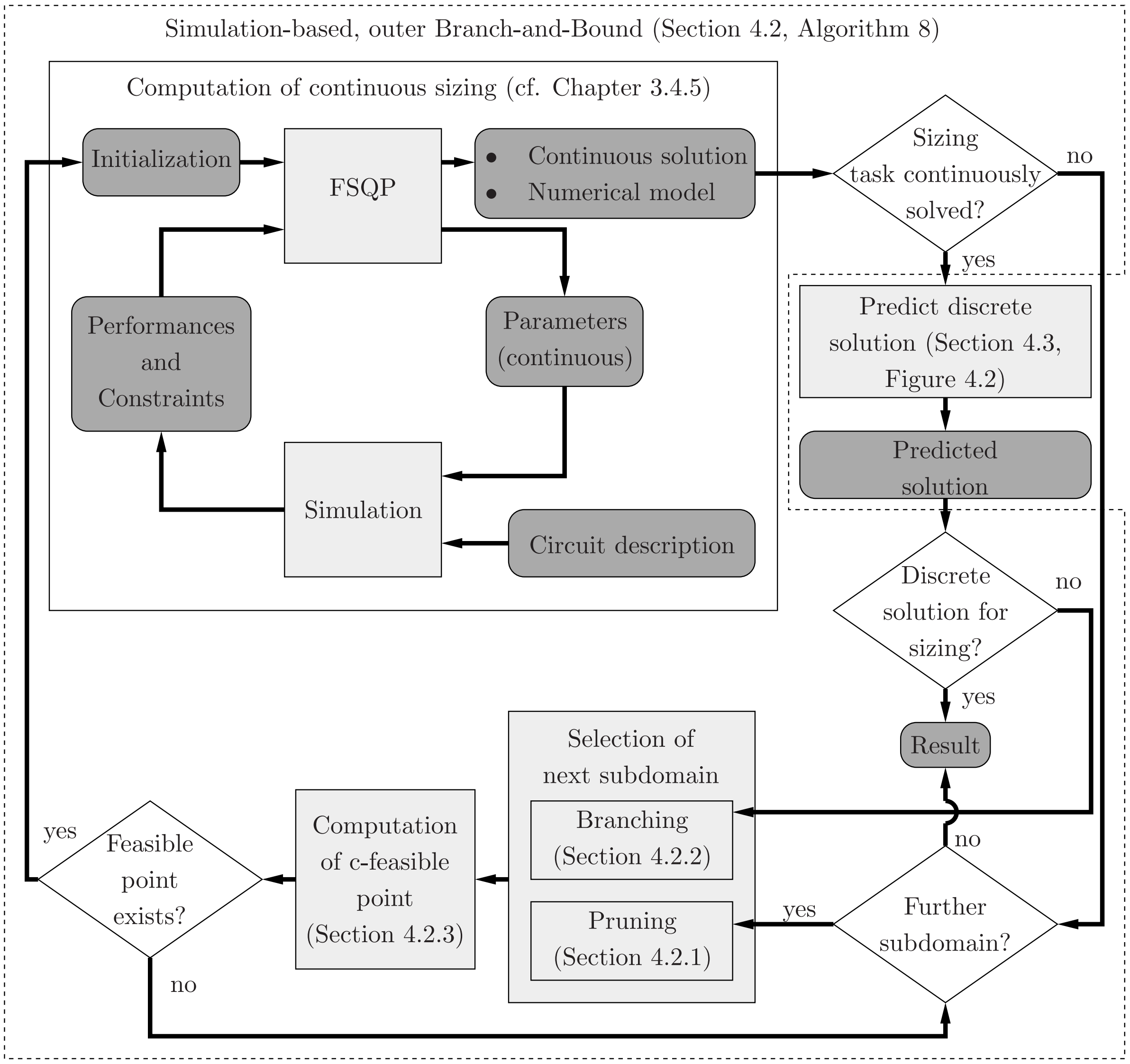

Simulation-based, outer Branch-and-Bound (Section 4.2, Algorithm 8)
Computation of continuous sizing (cf. Chapter 3.4.5)
Initialization
FSQP
• Continuous solution
• Numerical model
Performances and Constraints
Parameters (continuous)
Simulation
Circuit description
Sizing task continuously solved?
no
yes
Predict discrete solution (Section 4.3, Figure 4.2)
Predicted solution
Discrete solution for sizing?
no
yes
Result
Selection of next subdomain
Branching (Section 4.2.2)
Pruning (Section 4.2.1)
Further subdomain?
no
yes
Computation of c-feasible point (Section 4.2.3)
Feasible point exists?
yes
no

Figure 4.1: Structure of the new discrete sizing approach for continuously simulated parameters

2. New branching heuristics are developed for the analog sizing task to reduce the number of subproblems which must be solved to find a discrete solution (Section 4.2.2). For this purpose, a direction of improvement is computed for the sizing task and branching into this direction is preferred.

In Figure 4.1, the new pruning rules are realized by the decision blocks in the diagram. If a subdomain does not contain a solution for the sizing task or if a discrete point has been found – i.e., if the new pruning rules are applied – the decisions either lead to a result which is returned or cause the pruning of the current subdomain. The returned result can be non-optimal but solves the sizing task if such a solution exists locally.

The new branching rules are realized in the "Selection of next subdomain" block in Figure 4.1 which also implements the recursive call of the Branch-and-Bound approach. Branching is only applied if a continuous solution for the sizing task exists in the current subdomain and if the result computed by the FSQP approach is non-discrete.

Besides the modifications above which are used to speed up the Branch-and-Bound algorithm – predominantly by reducing the number of required simulations – the approach must also be modified for the use of the efficient, simulation-based FSQP algorithm: The FSQP approach requires a c-feasible initial solution, i.e., a solution which fulfills the constraints of the optimization problem. However, in each recursion of the Branch-and-Bound approach, rounding constraints are added to construct two new subdomains. These rounding constraints are generated by forcing one component of the design point to be greater or lower than this component of the continuous solution $\widetilde{\mathbf{d}}^*_{\mathrm{rel}}$ rounded to the next discrete values (cf. (3.79) and (3.80)). As a consequence, $\widetilde{\mathbf{d}}^*_{\mathrm{rel}}$ is not feasible in both new subdomains and cannot be taken as initial solution for the FSQP approach.

A new, fast, three-step approach is implemented in the Branch-and-Bound algorithm for analog sizing to find a c-feasible initial solution for the FSQP approach, keeping the required number of simulations low (cf. Section 4.2.3). The approach starts with a discretization of the continuous solution, using a rounding operation. If the point which is computed this way violates some constraints, a goal attainment approach is used to find a feasible solution without additional simulations. Computationally expensive simulations are only required if none of the previously computed candidates can be used as initial solution for FSQP.

## 4.1.2 Discrete Sizing Approach for Continuously Simulated Parameters

The enhancements presented in Section 4.1.1 lead to a significant reduction of the runtime of the outer Branch-and-Bound algorithm. However, the number of subproblems which must be solved by a simulation-based FSQP approach can be still high. For further reduction of the runtime of the algorithm, another new enhancement is added to the structure of the Branch-and-Bound algorithm for analog sizing (cf. Figure 4.1) which predicts the discrete solution of the sizing task using only a low number of simulations (cf. Section 4.3). The inner structure of the new block is shown in Figure 4.2.

The prediction algorithm requires a linearly constrained quadratic model of the optimization problem as an input. Such a model is computed during the FSQP approach and can be used without additional effort. A prediction for the discrete solution of the sizing task is computed on this model by an inner Branch-and-Bound approach (cf. Section 4.3.3).

The inner Branch-and-Bound approach uses an active set algorithm (cf. Chapter 3.3.2) to compute a continuous solution of the model. The pruning rules are defined according

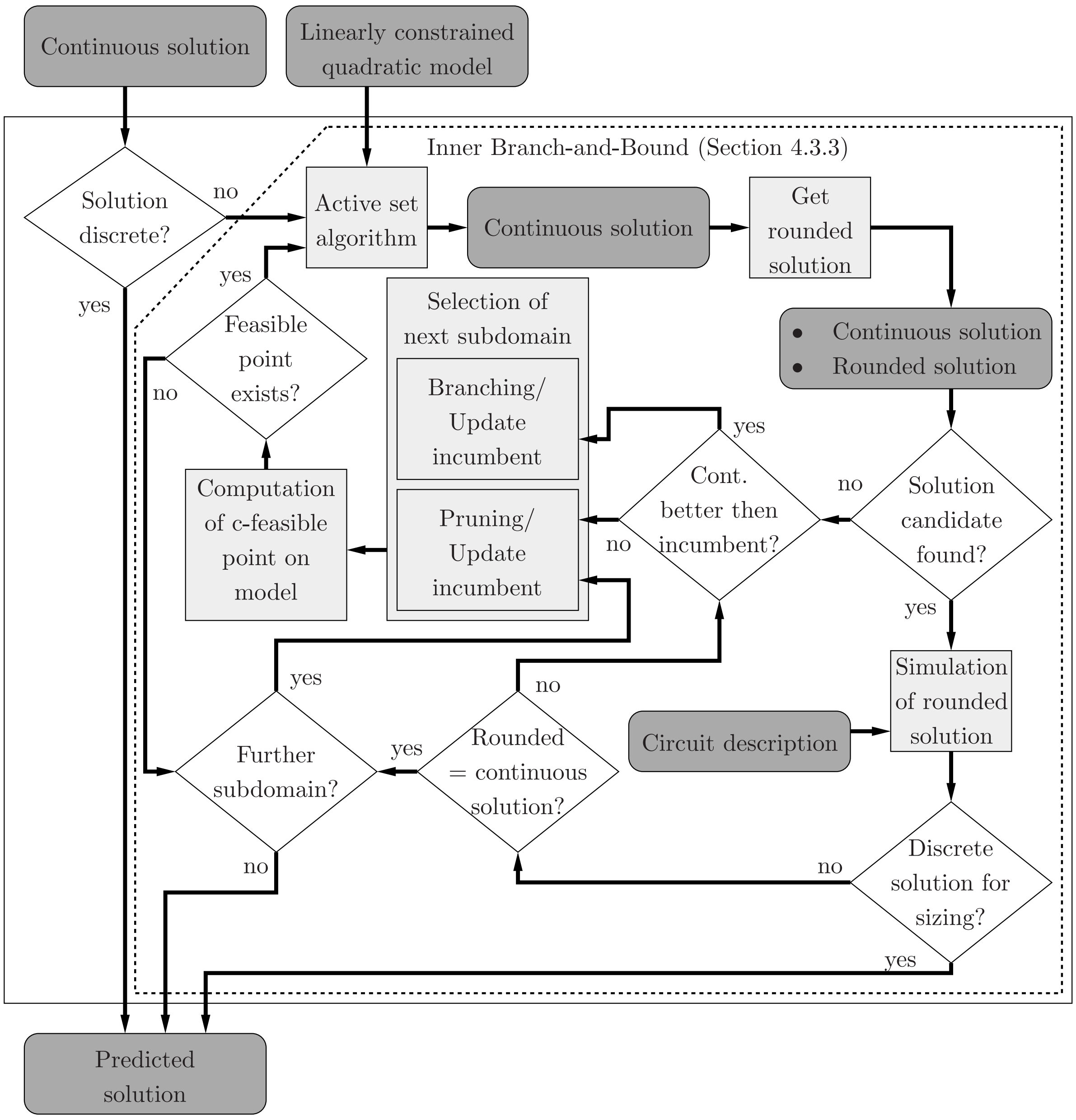

Continuous solution
Linearly constrained quadratic model
Inner Branch-and-Bound (Section 4.3.3)
Solution discrete?
no
yes
Active set algorithm
Continuous solution
Get rounded solution
Continuous solution
Rounded solution
Feasible point exists?
yes
no
Selection of next subdomain
Branching/ Update incumbent
Pruning/ Update incumbent
Computation of c-feasible point on model
Cont. better then incumbent?
yes
no
Solution candidate found?
no
yes
Simulation of rounded solution
Circuit description
Further subdomain?
yes
no
Rounded = continuous solution?
yes
no
Discrete solution for sizing?
no
yes
Predicted solution

Figure 4.2: Structure of the prediction algorithm for a discrete solution

to Chapter 3.5.2. For branching, a new heuristic is used which considers the requirements of the task to predict a discrete solution. To increase efficiency, the inner Branch-and-Bound approach is revised and a new algorithm is presented in Algorithm 10 which is, in addition, highly parallelized. Without further modifications, the inner Branch-and-Bound algorithm would compute the discrete optimum on the linearly constrained quadratic model provided. However, the computation of the optimum can be costly due to the exponential runtime behavior of Branch-and-Bound approaches, and optimality is not required to predict a solution of the sizing task (cf. Section 4.3).

To keep runtime low, the linearly constrained quadratic program is formulated in such a way that each feasible discrete point in the model is a solution candidate for the sizing task (cf. Section 4.3.2). I.e., a discrete solution which is computed by the active set algorithm on the model is a potential solution. Each solution candidate is evaluated by simulations. If it solves the sizing task, the inner Branch-and-Bound algorithm is stopped.

To find solution candidates for the analog sizing task early in the inner Branch-and-Bound algorithm, the – possibly continuous – solution of the active set algorithm is rounded to the next discrete point, and the rounded point is considered as a potential solution for the sizing task if it is feasible in the model. Experiments have shown that the runtime of the inner Branch-and-Bound algorithm can be typically reduced by considering these rounded solutions of the active set algorithm.

Comparing the inner Branch-and-Bound algorithm in Figure 4.2 to the outer Branch-and-Bound algorithm in Figure 4.1, the main differences in the structure are the use of an active set algorithm instead of the computationally expensive, simulation-based FSQP approach and the consideration of the rounded solution. In addition, the incumbent solution is considered and the solution of the active set algorithm is tested for discreteness – by comparing the rounded solution to the continuous solution – to decide if further branching is required or if the current subdomain should be pruned. These differences are due to the modified pruning rules in the outer Branch-and-Bound approach and to the sub-optimality of the rounded solution in the inner Branch-and-Bound approach: A solution which does not solve the sizing task in the outer Branch-and-Bound approach corresponds to a domain which cannot contain a solution of the sizing task, i.e., the corresponding domain can be pruned. In contrast, the inner Branch-and-Bound approach searches for the optimum of the model and is stopped sooner only if a solution of the sizing task has been found. I.e., the three pruning rules in Chapter 3.5.2 are applied instead of the two new pruning rules in Section 4.2.1 and the incumbent solution is required for the inner Branch-and-Bound algorithm. In addition, the rounded solution of the active set algorithm is not necessarily optimal and the corresponding domain can contain better discrete solutions for the model. Therefore, only the result of the active set algorithm is considered for pruning. Nevertheless, each discrete point can become an incumbent if it is better than the currently best discrete point and the incumbent can be updated by the rounded solution before branching.

Due to inaccuracies of the model, the prediction algorithm cannot guarantee that an existing solution for the discrete sizing task is found and the outer Branch-and-Bound approach is required to ensure the convergence of the new Branch-and-Bound based approach for analog sizing. However, the prediction of the discrete solution yields good results in practical experiments, where a significant speed-up can be achieved using this method. In many cases, a discrete solution of the sizing task can be found in the first recursion of the outer Branch-and-Bound approach, i.e., the computationally expensive simulation-based FSQP approach must be executed only once.

## 4.2 Outer Branch-and-Bound approach

The outer Branch-and-Bound approach described in this section guarantees the (local) convergence of the presented approach. Specific characteristics of the discrete analog sizing task are considered to make the approach efficient. For this purpose, the pruning rules are revised and a new Branch-and-Bound algorithm is presented in Section 4.2.1. Furthermore, a new branching heuristic is presented in Section 4.2.2 and the computation of a feasible initial solution is discussed in Section 4.2.3, which is required to enable the use of the highly efficient FSQP approach presented in Chapter 3.4.5.

### 4.2.1 Pruning Rules for Analog Sizing

The sizing task is fulfilled according to (2.52) if a discrete design parameter point $\mathbf{d}^*$ has been computed where each design constraint and each performance specification is fulfilled. Applying the Branch-and-Bound algorithm in Chapter 3.5.4 to such a problem, the pruning rules can be simplified to:

1. If the relaxed subdomain $\widetilde{\mathbb{D}}_{\text{rel}}^N$ does not contain any solution $\widetilde{\mathbf{d}}_{\text{rel}}^*$ for the sizing task in the continuous domain, it follows that the current discrete subdomain $\widetilde{\mathbb{D}}^N \subseteq \widetilde{\mathbb{D}}_{\text{rel}}^N$ cannot contain a solution for the sizing task. Hence, $\widetilde{\mathbb{D}}^N$ can be pruned. This is referred to as *pruning by non-satisfiability* in the following.

2. If the optimum in the relaxed subdomain $\widetilde{\mathbb{D}}_{\text{rel}}^N$ is discrete and solves the sizing task, the result can be accepted as solution for the complete problem. I.e., no further subdomain has to be considered and each possible subdomain can be pruned. This is referred to as *pruning by acceptance* in the following.

The first pruning rule proposed can be interpreted as the combination of the rules for *pruning by value dominance* and *pruning by infeasibility* in Chapter 3.5.2, where the value dominance is anticipated by the assumption that a solution for the sizing task exists according to (2.52). The second pruning rule proposed can be interpreted as *pruning by optimality* applied to the analog sizing task.

The proposed Branch-and-Bound algorithm for analog sizing is shown in Algorithm 8. In addition to the modified pruning conditions, a stop criterion is added in line 1 which stops the algorithm if any discrete solution for the sizing task has been found. Two further key points in the algorithm are the computation of a solution in the relaxed domain (line 3) and the branching rules which are used in lines 9 and 10 to define the subdomains of the current domain in each recursion (lines 11 and 12). The enhancements for these open points are discussed in Sections 4.2.2 and 4.2.3.

**Algorithm 8:** Outer Branch-and-Bound Approach (cf. Figure 4.1)

```
Input: discrete domain $\mathbb{D}^N$, current subdomain $\widetilde{\mathbb{D}}^N_{\text{rel}}$, incumbent $\mathbf{d}_{\text{inc}}$
if $\mathbf{d}_{inc}$ solves the sizing task then                                                  1
    //solution has been found in any subdomain
    return $\mathbf{d}_{\text{inc}}$                                                                2
compute feasible solution $\widetilde{\mathbf{d}}^*_{\text{rel}}$ using (4.3), $\widetilde{\mathbb{D}}^N_{\text{rel}}$, and FSQP     3
if No feasible solution exists or $\widetilde{\mathbf{d}}^*_{rel}$ does not solve continuous sizing task then   4
    //pruning by non-satisfiability
    return $\mathbf{d}_{\text{inc}}$                                                                5
else if $\widetilde{\mathbf{d}}^*_{rel} \in \mathbb{D}^N$ then                                     6
    //pruning by acceptance
    return $\widetilde{\mathbf{d}}^*_{\text{rel}}$                                                  7
else                                                                                               8
    Select branching parameter $d_j$                                                               9
    set $\mathbb{D}^N_1$, $\mathbb{D}^N_2$ according to (4.5) and (4.6)                            10
    set $\mathbf{d}_{\text{inc}}$ = Outer Branch-and-Bound($\mathbb{D}^N$, $\mathbb{D}^N_1$, $\mathbf{d}_{inc}$)    11
    set $\mathbf{d}_{\text{inc}}$ = Outer Branch-and-Bound($\mathbb{D}^N$, $\mathbb{D}^N_2$, $\mathbf{d}_{inc}$)    12
    return $\mathbf{d}_{\text{inc}}$                                                               13
end                                                                                                14
```

### 4.2.2 Branching Rules for Analog Sizing

The new gradient-based heuristic which is used for the branching rules in Algorithm 8 considers that – to avoid unnecessary simulations – the FSQP approach is stopped as soon as any solution $\widetilde{\mathbf{d}}^*_{\text{rel}}$ for the sizing task has been found. However, due to the offset value $\gamma \neq 0$ which is used in the objective function (4.2), the gradient of the Lagrange function at this continuous solution is typically not equal to zero, and a direction of improvement $\Delta\mathbf{d}$ – i.e., an SQP step – can be computed for the objective function at $\widetilde{\mathbf{d}}^*_{\text{rel}}$ (cf. (3.51) on Page 55). The idea of the branching rule presented is to prefer solutions in direction of $\Delta\mathbf{d}$.

For this purpose, a parameter is selected for branching so that it corresponds to a component $\widetilde{d}_j$ of $\widetilde{\mathbf{d}}^*_{\text{rel}}$, which should be discretized, and so that it corresponds to an absolute value of the components $\Delta d_j$ of $\Delta\mathbf{d}$ which is as large as possible. I.e., the index $j$ of the branching parameter is computed by:

$$j = \arg\max_i |\Delta d_i| \text{ s.t. } \widetilde{d}_i \text{ should be discretized} \tag{4.4}$$

The largest component is selected, as the current parameter point would be modified mostly in this direction if another SQP step were performed at $\widetilde{\mathbf{d}}^*_{\text{rel}}$. I.e., the next step of an SQP approach would cause the strongest modification of $\widetilde{\mathbf{d}}^*_{\text{rel}}$ toward the next discrete value for the component with the greatest value $|\Delta d_i|$.

With the result of (4.4), the new subdomains $\mathbb{D}_1^N$, $\mathbb{D}_2^N$ in line 10 of Algorithm 8 are generated from the current relaxed domain $\widetilde{\mathbb{D}}^N_{\text{rel}}$ by:

$$\mathbb{D}_1^N = \begin{cases} \left\{ \mathbf{d} \in \widetilde{\mathbb{D}}^N_{\text{rel}} \middle| d_j - \left\lceil \widetilde{d}_j \right\rceil \geq 0 \right\} & \text{if } \Delta d_j > 0 \\ \left\{ \mathbf{d} \in \widetilde{\mathbb{D}}^N_{\text{rel}} \middle| \left\lfloor \widetilde{d}_j \right\rfloor - d_j \geq 0 \right\} & \text{if } \Delta d_j \leq 0 \end{cases} \tag{4.5}$$

$$\mathbb{D}_2^N = \begin{cases} \left\{ \mathbf{d} \in \widetilde{\mathbb{D}}^N_{\text{rel}} \middle| \left\lfloor \widetilde{d}_j \right\rfloor - d_j \geq 0 \right\} & \text{if } \Delta d_j > 0 \\ \left\{ \mathbf{d} \in \widetilde{\mathbb{D}}^N_{\text{rel}} \middle| d_j - \left\lceil \widetilde{d}_j \right\rceil \geq 0 \right\} & \text{if } \Delta d_j \leq 0 \end{cases} \tag{4.6}$$

$\widetilde{d}_j$ and $d_j$ is the $j$-th component of $\widetilde{\mathbf{d}}^*_{\text{rel}}$ and $\mathbf{d}$, respectively.

The subdomains are defined in (4.5) and (4.6) such that the direction of improvement of the objective function is considered first in the recursive Branch-and-Bound approach in Algorithm 8. If the gradient of the Lagrangian function at the solution found by the FSQP approach is zero and no direction of improvement can be computed[1], the first parameter which should be discretized is selected for branching, and the subdomain $\left\{ \mathbf{d} \in \widetilde{\mathbb{D}}^N_{\text{rel}} \middle| \left\lfloor \widetilde{d}_j \right\rfloor - d_j \geq 0 \right\}$ is considered first.

### 4.2.3 Computation of Feasible Initial Solutions

In each recursion of Algorithm 8 the sizing task must be solved for the current relaxed (i.e., continuous) subdomain (line 3). In this thesis, the simulation-based FSQP algorithm described in Chapter 3.4.5 is used to compute such a solution[2]. The FSQP algorithm requires a feasible initial solution as a starting point. Such a feasible initial solution must be computed for each subproblem which is considered in a Branch-and-Bound recursion because the rounding constraints implanted additionally are violated for the solution of the previous recursion. The number of simulations required to find such a feasible initial point should be as low as possible. Thus, it is computed using a linear model of the constraints if a rounding operation is not sufficient:

Assuming that $\widetilde{\mathbf{d}}^*_{\text{rel}}$ is the solution of the relaxed problem for the current recursion, $\mathbf{J}_c(\widetilde{\mathbf{d}}^*_{\text{rel}})$ and $\mathbf{c}(\widetilde{\mathbf{d}}^*_{\text{rel}})$ are the Jacobian matrix of the constraints and the constraint values

[1]Such cases appeared in practical experiments only if the specifications for the circuits were weak, i.e., if the sizing task was easy to solve.

[2]It can be noted that – contrary to the assumption in the FSQP approach – equality constraints appear during the Branch-and-Bound approach if the upper and the lower bound has the same value for any parameter. Such equality constraints do not affect the usability of the FSQP approach because they can be considered by parameter reductions.

at $\widetilde{\mathbf{d}}^*_{\mathrm{rel}}$. If branching constraints are added for component $j$ according to (3.79), (3.80) on Page 74, a feasible point in one of the resulting subdomains can be approximated using a goal attainment approach of the form

$$\begin{aligned} \min_{\chi_{up},\mathbf{d}} \chi_{up} \text{ s.t.} \quad & \mathbf{J}_c(\widetilde{\mathbf{d}}^*_{\mathrm{rel}}) \cdot \left(\mathbf{d} - \widetilde{\mathbf{d}}^*_{\mathrm{rel}}\right) + \mathbf{c}\left(\widetilde{\mathbf{d}}^*_{\mathrm{rel}}\right) \geq \mathbf{0} \\ & d_j - \lceil \widetilde{d}_j \rceil + \chi_{up} \geq 0 \\ & \chi_{up} \geq 0 \end{aligned} \tag{4.7}$$

if the branching constraint is $d_j - \lceil \widetilde{d}_j \rceil \geq 0$ (cf. (3.79)) and of the form

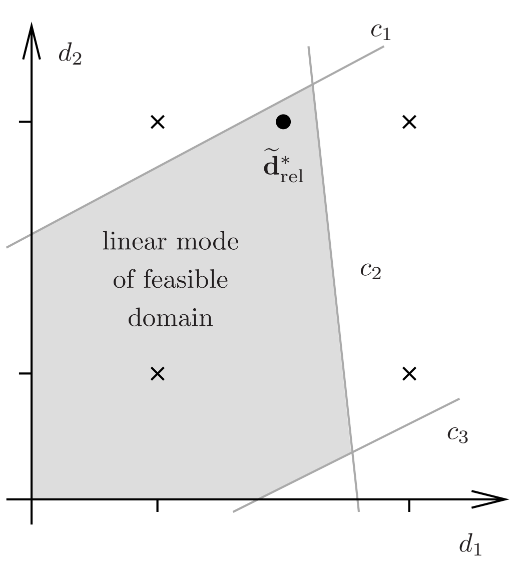
$$\begin{aligned} \min_{\chi_{down},\mathbf{d}} \chi_{down} \text{ s.t.} \quad & \mathbf{J}_c(\widetilde{\mathbf{d}}^*_{\mathrm{rel}}) \cdot \left(\mathbf{d} - \widetilde{\mathbf{d}}^*_{\mathrm{rel}}\right) + \mathbf{c}\left(\widetilde{\mathbf{d}}^*_{\mathrm{rel}}\right) \geq \mathbf{0} \\ & \lfloor \widetilde{d}_j \rfloor - d_j + \chi_{down} \geq 0 \\ & \chi_{down} \geq 0 \end{aligned} \tag{4.8}$$

if the branching constraint is $\lfloor \widetilde{d}_j \rfloor - d_j \geq 0$ (cf. (3.80)). In (4.7) and (4.8), $\widetilde{d}_j$ is a component of $\widetilde{\mathbf{d}}^*_{\mathrm{rel}}$. The additional parameters $\chi_{up}$ and $\chi_{down}$ describe the minimum required change in parameter $d_j$ to fulfill the rounding constraint and can be interpreted as slack variables which are used to relax the additional constraint. The optimization problems can be solved by the simplex algorithm in Chapter 3.2.3. This optimization can be interpreted as restricting the relaxed constraint.

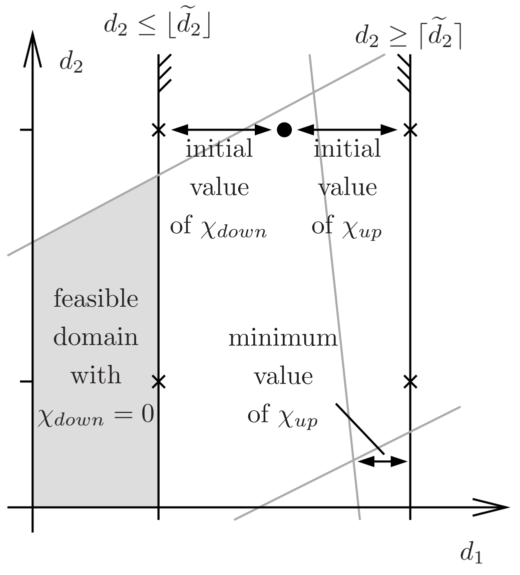
The linear problems (4.7), (4.8) have an initial solution for $\mathbf{d} = \widetilde{\mathbf{d}}^*_{\mathrm{rel}}$ and $\chi_{up} = \lceil \widetilde{d}_j \rceil - \widetilde{d}_j$ and $\chi_{down} = \widetilde{d}_j - \lfloor \widetilde{d}_j \rfloor$ (cf. Figure 4.3). The minima of (4.7) and (4.8) are approximations for feasible points in the corresponding subdomains of the sizing task if $\chi_{up} = 0$ for (4.7) and $\chi_{down} = 0$ for (4.8) (cf. $\chi_{down}$ in Figure 4.3b), respectively. Otherwise, i.e., if $\chi_{up} \neq 0$ or $\chi_{down} \neq 0$, the linear model of the corresponding subdomain does not contain a feasible point (cf. $\chi_{up}$ in Figure 4.3b).

It cannot be guaranteed that the approximated feasible solution fulfills the nonlinear design constraints of the underlying sizing task. To overcome this problem, a feasible solution is computed for the relaxed subdomain by applying the simulation-based FSQP approach to an optimization problem of the form (2.53) if the initial point computed on linearized constraints does not fulfill the constraints of the actual sizing task (cf. Chapter 2.5.2). However, the higher computational effort of the simulation-based FSQP approach can typically be avoided by using the approximation derived by (4.7), (4.8).

In practical cases, the area where the specifications are fulfilled is typically a connected domain. As a consequence, the continuous solution of the sizing task in a newly constructed subdomain is often close to the continuous solution of the subdomain in the previous recursion. It can be deduced that it is preferable for the feasible initial solution for a newly generated subproblem to be close to the solution $\widetilde{\mathbf{d}}^*_{\mathrm{rel}}$ of the problem in this previous recursion. For this purpose, a line search in the directions from $\widetilde{\mathbf{d}}^*_{\mathrm{rel}}$ to the feasible initial solution which is computed by the approach presented in this section is applied in the outer Branch-and-Bound algorithm.


$d_2$
$c_1$
$\widetilde{\mathbf{d}}^*_{\text{rel}}$
linear mode of feasible domain
$c_2$
$c_3$
$d_1$

(a) Feasible continuous domain of a linearly constrained discrete problem with constraints $c_1$, $c_2$, and $c_3$.


$d_2 \leq \lfloor \widetilde{d}_2 \rfloor$
$d_2 \geq \lceil \widetilde{d}_2 \rceil$
$d_2$
initial value of $\chi_{down}$
initial value of $\chi_{up}$
feasible domain with $\chi_{down} = 0$
minimum value of $\chi_{up}$
$d_1$

(b) Existing feasible points can be found by minimization over $\chi_{up}$ and $\chi_{down}$.

Figure 4.3: Visualization of a linear model of the feasible domain computed at $\widetilde{\mathbf{d}}^*_{\text{rel}}$ and of optimization problems (4.7), (4.8).

## 4.3 Prediction of a Discrete Solution

The approach presented in Section 4.2 solves the discrete sizing task if local optimization is sufficient. However, in each recursion of the outer Branch-and-Bound approach a simulation-based optimization must be executed to find the solution of the relaxed problem.

To increase the efficiency of the new Branch-and-Bound algorithm for analog sizing, the discrete solution is predicted on a linearly constrained quadratic model of the sizing task and the circumstance that the objective function value at each discrete feasible point is an upper bound for the discrete optimum is made use of. In a standard Branch-and-Bound approach this means that a feasible approximation of the discrete solution can be considered as an incumbent (cf. Chapter 3.5.4) if it is better than each discrete solution found previously. As a consequence, each subdomain with a continuous solution worse than the incumbent solution can be pruned. In case the analog sizing task is defined as in this thesis and using the proposed pruning rules for analog sizing, the simulation-based Branch-and-Bound approach can be stopped as soon as the approximated discrete solution fulfills specifications and constraints of the underlying problem.

### 4.3.1 Consideration of a Rounded Solution

The first idea to approximate the discrete solution is to compute the relaxed solution $\widetilde{\mathbf{d}}_{\text{rel}}^{*}$ for each subproblem, using the simulation-based FSQP approach, then to round each of these relaxed solutions to the next discrete point $\left\lceil \widetilde{\mathbf{d}}_{\text{rel}}^{*} \right\rfloor$ (cf. (2.19)), and to stop the algorithm if a point $\left\lceil \widetilde{\mathbf{d}}_{\text{rel}}^{*} \right\rfloor$ fulfills the specifications and constraints. The rounded solution does not solve a discrete optimization problem or a discrete sizing task in general (cf. Chapter 1.2.2). However, experiments have shown that – due to the significant runtime reduction which can be achieved if $\left\lceil \widetilde{\mathbf{d}}_{\text{rel}}^{*} \right\rfloor$ solves the sizing task – the rounded solution should be considered as an extension of the Branch-and-Bound algorithm in practice. It can also be observed that the rounded solution is typically a good choice if $\widetilde{\mathbf{d}}_{\text{rel}}^{*}$ is close to a discrete solution.

Cases where the relaxed solution $\widetilde{\mathbf{d}}_{\text{rel}}^{*}$ is close to $\left\lceil \widetilde{\mathbf{d}}_{\text{rel}}^{*} \right\rfloor$ appear in particular in later recursions of the Branch-and-Bound algorithm when many rounding constraints have been added to the problem and, as a consequence, many parameters are discretized. It follows that considering the rounded solution can help to reduce the number of recursions – i.e., the number of subproblems to be solved and the runtime – of a Branch-and-Bound approach. In addition, it guarantees that the suggested Branch-and-Bound approach is as fast as continuous optimization with subsequent rounding if this state-of-the-art approach can find a solution, but can still solve the sizing task if the state-of-the-art approach fails.

### 4.3.2 Model-Based Prediction Approach

The number of subproblems and, as a consequence, the runtime of the algorithm is still high in practical cases if only the rounded solution is considered in each recursion. To increase the efficiency of the algorithm for analog sizing, a prediction method is suggested in this thesis to speed up the search for a discrete solution. This is shown in the structure of the new approach in Figure 4.1 which consists of the outer Branch-and-Bound approach and a block which realizes the prediction of a discrete solution. The structure in this figure is realized by replacing the computation of a relaxed solution in line 3 of Algorithm 8 by Algorithm 9. I.e., not only the solution of the relaxed problem but also a discrete solution candidate is computed in each step of the Branch-and-Bound algorithm. The approximation of the discrete solution in Algorithm 9 is realized in two steps:

1. The relaxed solution computed by the FSQP approach is rounded to the next point (line 2).
2. The discrete solution is predicted by an inner Branch-and-Bound algorithm applied to a linearly constrained quadratic model (line 5).

**Algorithm 9:** Solve Subproblem (replaces line 3 of Algorithm 8)

```
Input: discrete domain $\mathbb{D}^N$, current subdomain $\widetilde{\mathbb{D}}^N_{\text{rel}}$
compute feasible solution $\widetilde{\mathbf{d}}^*_{\text{rel}}$ for (4.3) and $\widetilde{\mathbb{D}}^N_{\text{rel}}$ using FSQP      1
if $\mathbf{c}\left(\left\lceil \widetilde{\mathbf{d}}^*_{rel} \right\rfloor\right) \geq \mathbf{0}$ and $\widetilde{\mathbf{d}}^*_{rel}$ fulfills performance specifications then      2
    //return rounded solution
    return $\left\lceil \widetilde{\mathbf{d}}^*_{\text{rel}} \right\rfloor$      3
else      4
    compute approximated discrete solution $\mathbf{d}_{approx}$ by solving (4.9) for $\widetilde{\mathbb{D}}^N_{\text{rel}} \cap \mathbb{D}^N$      5
end      6
if $\mathbf{c}(\mathbf{d}_{approx}) \geq \mathbf{0}$ and $\mathbf{d}_{approx}$ fulfills performance specifications then      7
    //return approximation computed by solving (4.9)
    return $\mathbf{d}_{approx}$      8
else      9
    //return non-discrete solution
    return $\widetilde{\mathbf{d}}^*_{\text{rel}}$      10
end      11
```

The consideration of the rounded solution discussed in Section 4.3.1 requires the simulation of only one additional – namely the rounded – point. Due to this small additional effort the rounded solution is considered before the prediction of a discrete solution using the inner Branch-and-Bound algorithm, although it is neglected in the structure of Figure 4.1 for the sake of a simple description. The prediction algorithm returns either a discrete solution for the sizing task or the solution in the relaxed domain computed by the FSQP approach.

For the approximation of the discrete solution in line 5 advantage is taken of the inner structure of the FSQP approach: In each step of an SQP approach a linearly constrained quadratic optimization problem (cf. (3.51)) is computed. This problem is an approximation for the optimization task which must be solved to compute the required sizing of the analog circuit. As a consequence, the discrete optimum of this model is a candidate for the discrete solution of the sizing task if the last model is used which was computed during the FSQP approach. Additional linear approximations of the performances can be used as further constraints for the model to reduce the size of the feasible domain in the quadratic optimization problem and to avoid a search in regions where no solution for the sizing task is to be expected.

Using the linearly constrained quadratic model from the FSQP approach and linear approximations of the performances, the discrete solution for the current subproblem in a certain Branch-and-Bound recursion of Algorithm 8 can be approximated by solving the optimization problem

$$\begin{aligned} \min_{\mathbf{d} \in \mathbb{D}^N \cap \widetilde{\mathbb{D}}^N_{\text{rel}}} \quad & \widehat{\varphi}(\mathbf{d}) := \tfrac{1}{2}\left(\mathbf{d} - \widetilde{\mathbf{d}}^*_{\text{rel}}\right)^T \mathbf{H} \left(\mathbf{d} - \widetilde{\mathbf{d}}^*_{\text{rel}}\right) + \mathbf{g}^T \left(\mathbf{d} - \widetilde{\mathbf{d}}^*_{\text{rel}}\right) \\ \text{s.t.} \quad & -\mathbf{J}_\varepsilon \cdot \left(\mathbf{d} - \widetilde{\mathbf{d}}^*_{\text{rel}}\right) - \boldsymbol{\varepsilon}\left(\widetilde{\mathbf{d}}^*_{\text{rel}}\right) \geq \mathbf{0} \\ & \mathbf{J}_c \cdot \left(\mathbf{d} - \widetilde{\mathbf{d}}^*_{\text{rel}}\right) + \mathbf{c}\left(\widetilde{\mathbf{d}}^*_{\text{rel}}\right) \geq \mathbf{0} \end{aligned} \tag{4.9}$$

over the discrete domain $\mathbb{D}^N \cap \widetilde{\mathbb{D}}^N_{\text{rel}}$. In (4.9) $\mathbf{H}$ is the Hessian matrix of the Lagrangian function which is computed during the FSQP approach, $\mathbf{J}_\varepsilon$ and $\mathbf{J}_c$ are the Jacobian matrices for performance-to-specification gaps, and for constraints at the solution in the relaxed domain $\widetilde{\mathbf{d}}^*_{\text{rel}}$. $\boldsymbol{\varepsilon}\left(\widetilde{\mathbf{d}}^*_{\text{rel}}\right)$ and $\mathbf{c}\left(\widetilde{\mathbf{d}}^*_{\text{rel}}\right)$ are the values of the performance gaps and of the constraints at $\widetilde{\mathbf{d}}^*_{\text{rel}}$. The additional performance constraints are formulated in such a way that the values of the performance gaps should be negative, i.e., the performance specifications should be fulfilled, in the linear model. The linearly constrained quadratic optimization problem over a discrete domain in (4.9) is solved by the inner Branch-and-Bound algorithm in Section 4.3.3.

The new Branch-and-Bound approach for analog sizing predicts a discrete solution by the model-based approach in each recursion of the outer Branch-and-Bound approach in Section 4.2. As a consequence, the incumbent – i.e., the upper bound for the discrete solution – can be updated in each recursion. Hence, the pruning rules can be applied sooner and a speed-up of the algorithm can be achieved also in case the approach is used to find an optimal solution if the approximation can be computed fast enough. For the analog sizing task according to (2.52) – i.e., with the objective of finding any discrete point which fulfills specifications and constraints – experiments show that in many cases already the first approximation of the discrete solution solves the problem. I.e., in many cases the simulation-based FSQP approach must be executed only once.

### 4.3.3 Inner Branch-and-Bound Algorithm

The discrete solution of the sizing task can be approximated in each recursion of Algorithm 8 by solving the linearly constrained quadratic model in (4.9). To solve the problem, an inner Branch-and-Bound algorithm is used. This inner Branch-and-Bound approach uses an active set method to compute relaxed solutions $\hat{\mathbf{d}}_{\text{rel}}$ on the model (cf. Chapter 3.3.2). In addition, the branching heuristic is adapted to the requirements of the inner Branch-and-Bound algorithm and, as a consequence of the changed branching heuristics, the Branch-and-Bound algorithm is restructured.

Given a non-discretized optimum $\hat{\mathbf{d}}_{\text{rel}}$ of the relaxation of (4.9), branching constraints of the form (3.79), (3.80) should be added in each recursion of the Branch-and-Bound algorithm for some parameter which must be discretized. In the presented approach for the inner Branch-and-Bound algorithm the parameters which have the highest influence on the objective function of the model $\widehat{\varphi}(\mathbf{d})$ should be considered first. This heuristic to select a branching parameter – together with the selection heuristic for the next subdomain – has shown a low recursion depth in practice. In addition, the influence of the parameters which must be discretized is low after a low number of recursions. As a consequence, considering the relaxed solution rounded to the next discrete point as possible incumbent for the Branch-and-Bound algorithm on the model leads to good results (similar to Section 4.3.1).

To select a parameter with a high influence, the mean influence $\Phi_j$ of parameter $j$ on the objective function during branching is defined as the mean value of the absolute influences in both possible branching directions:

$$\Phi_j = \frac{\left|\widehat{\varphi}\left(\left\lceil\hat{\mathbf{d}}_{\text{rel}}\right\rceil_j\right) - \widehat{\varphi}\left(\hat{\mathbf{d}}_{\text{rel}}\right)\right| + \left|\widehat{\varphi}\left(\left\lfloor\hat{\mathbf{d}}_{\text{rel}}\right\rfloor_j\right) - \widehat{\varphi}\left(\hat{\mathbf{d}}_{\text{rel}}\right)\right|}{2} \tag{4.10}$$

The value of the mean influence defined in this way depends on the distance from $\hat{\mathbf{d}}_{\text{rel}}$ to the rounded values $\left\lceil\hat{\mathbf{d}}_{\text{rel}}\right\rceil_j$ and $\left\lfloor\hat{\mathbf{d}}_{\text{rel}}\right\rfloor_j$ and on the sensitivity of the objective function at $\hat{\mathbf{d}}_{\text{rel}}$ against variations of parameter $j$. I.e., the mean influence of a parameter is high if a small change in the parameter causes a great increase or decrease of the objective function or if the distance to the next discrete values of the component is large. Based on (4.10), the branching index $b$ is defined as the index corresponding to the largest value of $\Phi_j$:

$$b = \arg\max_{j\in\mathbb{I}_{add}} \Phi_j \tag{4.11}$$

$\mathbb{I}_{add}$ is the index set of all parameters, which should be discretized but are not discrete in the current recursion.

After choosing a branching parameter, one of the two possible subdomains (cf. (3.81), (3.82)) must be selected to be considered first. This decision is made for the inner Branch-and-Bound approach based on the best achievable objective function values for the two relaxed subproblems. To compute these values for both subdomains, the linearly constrained quadratic problem for each subdomain must be solved before the next recursion is started. However, it can be seen in the Branch-and-Bound algorithm in Algorithm 7 on page 76 that in a standard Branch-and-Bound approach both subproblems are considered as well. Therefore, the decision which relaxed subdomain contains the better solution can be made without any additional computational effort by restructuring the Branch-and-Bound algorithm[3]. The resulting Branch-and-Bound algorithm denoted as *Best-Child-First Branch-and-Bound* is provided in Algorithm 10.

In contrast to the standard Branch-and-Bound algorithm introduced in Chapter 3.5.4, the new algorithm requires the optimum for the current domain as an input and starts with the pruning rules (lines 3 to 8). Afterwards, the branching parameter is selected (line 10) and the two possible subproblems which can be generated by branching along

[3]This restructuring is not shown in Figure 4.2 for the sake of a clear structure.

**Algorithm 10:** Best-Child-First Branch-and-Bound

```
Input: discrete domain $\mathbb{D}^N$, current subdomain $\widehat{\mathbb{D}}^N_{rel}$, incumbent $\mathbf{d}_{\text{inc}}$,
       optimum $\hat{\mathbf{d}}_{\text{rel}}$ for relaxed domain $\widehat{\mathbb{D}}^N_{rel}$
if constraints and specifications are fulfilled for the sizing task at $\mathbf{d}_{inc}$ then          1
    //Solution found; break Branch-and-Bound
    return $\mathbf{d}_{\text{inc}}$                                                                       2
else if any $c_l(\hat{\mathbf{d}}_{rel}) < 0$ then                                                         3
    //pruning by infeasibility
    return $\mathbf{d}_{\text{inc}}$                                                                       4
else if $\widehat{\varphi}\left(\hat{\mathbf{d}}_{rel}\right) \geq \widehat{\varphi}\left(\mathbf{d}_{inc}\right)$ then    5
    //pruning by value dominance
    return $\mathbf{d}_{\text{inc}}$                                                                       6
else if $\hat{\mathbf{d}}_{rel} \in \mathbb{D}^N$ then                                                     7
    //pruning by optimality
    return $\hat{\mathbf{d}}_{\text{rel}}$                                                                 8
else                                                                                                       9
    compute branching parameter index $b$ using (4.11)                                                    10
    set $\mathbb{D}^N_{\text{down}} = \left\{\mathbf{d} \in \widehat{\mathbb{D}}^N_{rel} \middle| \left\lfloor \widehat{d_b} \right\rfloor - d_b \geq 0\right\}$ and $\mathbb{D}^N_{\text{up}} = \left\{\mathbf{d} \in \widehat{\mathbb{D}}^N_{rel} \middle| d_b - \left\lceil \widehat{d_b} \right\rceil \geq 0\right\}$    11
    compute optima of subdomains $\hat{\mathbf{d}}_{down}$ and $\hat{\mathbf{d}}_{up}$ by solving (4.9) for    12
            $\mathbb{D}^N_{\text{down}}$ and $\mathbb{D}^N_{\text{up}}$ using the active set algorithm in Algorithm 2
    if $\widehat{\varphi}\left(\hat{\mathbf{d}}_{down}\right) \leq \widehat{\varphi}\left(\hat{\mathbf{d}}_{up}\right)$ then    13
        set $\mathbf{d}_{\text{inc}}$ = Best-Child-First Branch-and-Bound($\mathbb{D}^N$, $\mathbb{D}^N_{down}$, $\mathbf{d}_{inc}$, $\hat{\mathbf{d}}_{down}$)    14
        set $\mathbf{d}_{\text{inc}}$ = Best-Child-First Branch-and-Bound($\mathbb{D}^N$, $\mathbb{D}^N_{up}$, $\mathbf{d}_{inc}$, $\hat{\mathbf{d}}_{up}$)    15
    else                                                                                                  16
        set $\mathbf{d}_{\text{inc}}$ = Best-Child-First Branch-and-Bound($\mathbb{D}^N$, $\mathbb{D}^N_{up}$, $\mathbf{d}_{inc}$, $\hat{\mathbf{d}}_{up}$)    17
        set $\mathbf{d}_{\text{inc}}$ = Best-Child-First Branch-and-Bound($\mathbb{D}^N$, $\mathbb{D}^N_{down}$, $\mathbf{d}_{inc}$, $\hat{\mathbf{d}}_{down}$)    18
    end                                                                                                   19
    return $\mathbf{d}_{\text{inc}}$                                                                      20
end                                                                                                       21
```

this parameter (line 11) are evaluated (line 12). The order for the consideration of these subproblems is determined in lines 13 to 18 such that the subproblem with the better relaxed optimum is preferred. The new recursion is started with an already computed optimal solution for the subproblem and with the subdomain under consideration. To simplify the algorithm, infeasible solutions are treated as feasible solutions with infinite objective function values in the current recursion and the corresponding subdomains are pruned if a recursion is started with an infeasible point.

Although the surrogate problem in (4.9) is explicitly given and can be solved without additional simulations, the runtime of an optimization with the inner Branch-and-Bound approach can still be high due to its exponential behavior. However, the structure of the underlying problem can be exploited to reduce the runtime: The linear constraints for the problem do not only consider a linearization for the constraints but also a linear model of the performances such that not only the optimal but also any other feasible discrete solution in the model is a solution candidate for the sizing task. Thus, suboptimal discrete solutions for the quadratic problem found during the inner Branch-and-Bound approach, i.e., each candidate for a new incumbent solution, can be evaluated by simulations and the inner and outer Branch-and-Bound approach can be stopped if such a suboptimal solution solves the sizing task. The evaluation of such a suboptimal solution is reasonable in the common case that the runtime of the inner Branch-and-Bound algorithm is higher than the runtime of a single simulation. To consider suboptimal solutions, an additional stop criterion is added to the inner Branch-and Bound algorithm (line 2).

On behalf of the implementation, the simulations required to evaluate suboptimal solutions can be parallelized and started in separate threads such that the inner Branch-and-Bound approach can continue the search for an optimal solution during the simulation. In addition, the first $N_{core}$ recursive calls of the inner Branch-and-Bound algorithm in this thesis are started in separate threads so that many numerical problems can be solved at the same time. $N_{core}$ can be, e.g., the number of available CPUs or cores. For the inner Branch-and-Bound approach the rounded relaxed solution is also used to further speed up the algorithm by considering the rounded solution as possible incumbent if it is feasible for the optimization problem (4.9).

## 4.4 Discrete Tolerance Design

The Branch-and-Bound algorithm presented so far solves a discrete analog sizing task for fixed operating and process conditions and if each continuous point can be simulated. In this section, the approach is enlarged to consider variations in operating and process parameters, i.e., to compute a tolerance design as defined in Chapter 2.5.4.

As discussed in Chapter 2.5.4, the task of finding a tolerance design is equivalent to the task of finding a sizing for the design parameters, which ensures that the performance specifications are fulfilled at the worst case points with respect to operating and process parameters (cf. (2.60), (2.62)). As a consequence, a relaxed solution for the required tolerance design can be computed by the FSQP approach in Chapter 3.4.5 if each performance evaluation – i.e., also each performance evaluation required to compute the sensitivities of the performances (Algorithm 4 on page 63, line 21) – is done at the corresponding worst case point.

The worst case points for operating parameters depend on the selected design parameter point $\hat{\mathbf{d}}$ and the selected process parameters $\hat{\mathbf{s}}$ and can be computed by solving the optimization problem (cf. (2.62)):

$$\mathbf{o}_{wci} = \arg \max_{\mathbf{o} \in \mathbb{T}_o^{N_o}} \varepsilon_i \left( \hat{\mathbf{d}}, \mathbf{o}, \hat{\mathbf{s}} \right) \tag{4.12}$$

Analogously, the worst case points for process parameters depend on the selected design parameter point $\hat{\mathbf{d}}$ and the selected operating parameters $\hat{\mathbf{o}}$ and can be computed by solving the optimization problem (cf. (2.60)):

$$\mathbf{s}_{wci} = \arg \max_{\mathbf{s} \in \mathbb{T}_s^{N_s}} \varepsilon_i \left( \hat{\mathbf{d}}, \hat{\mathbf{o}}, \mathbf{s} \right) \tag{4.13}$$

A first approach to compute the tolerance design using the FSQP approach in Chapter 3.4.5 could be to determine the worst case points exactly each time a function evaluation is required. However, such a procedure would cause an extremely high runtime of the algorithm because the accurate computation of the worst case points requires many time-consuming circuit simulations.

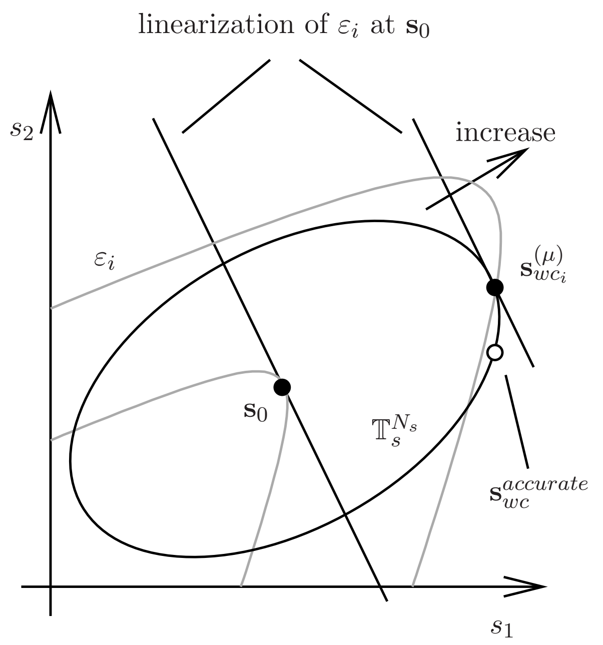
To reduce the runtime for the computation of a tolerance design to a practicable amount, the concept in [SSPG02] is used in this thesis, i.e., the worst case points are successively approximated over the iterations of the FSQP algorithm. The following assumptions are used for the approach described below:

1. The change of the worst case points is small if the design parameter point is shifted by a small value.
2. The worst case points can be computed using a gradient-based method.
3. The operating and process parameters are on the boundary of the operating region (2.21) and of the tolerance domain (2.59), respectively.

Although the assumptions can be violated for analog circuits, e.g., if mismatch parameters are considered [Sch04], the approach is applicable in most practical problems. If the suggested method does not provide the required accuracy for a certain sizing task, the accuracy of the method can be increased using an advanced but computationally more expensive method to approximate the worst case points during the FSQP approach.

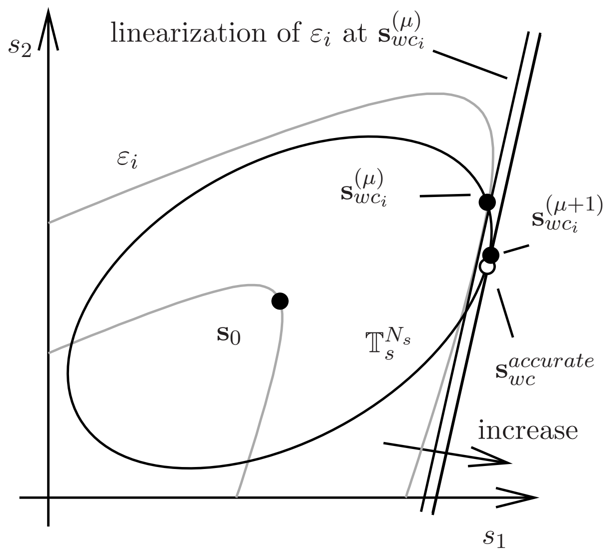
## 4.4.1 Approximation of worst case points and worst case sensitivity

In the presented approach, the worst case points are successively computed based on a linear model for each performance-to-specification gap. The worst case in this linear model is unique and results in an approximation of the worst case points on the bounds of the operating region $\mathbb{T}_o^{N_o}$ and of the tolerance domain for the process parameters $\mathbb{T}_s^{N_s}$ (cf. Figure 4.4a).



linearization of $\varepsilon_i$ at $\mathbf{s}_0$
$s_2$
increase
$\varepsilon_i$
$\mathbf{s}_{wc_i}^{(\mu)}$
$\mathbf{s}_0$
$\mathbb{T}_s^{N_s}$
$\mathbf{s}_{wc}^{accurate}$
$s_1$

(a) Approximation of worst case point $\mathbf{s}_{wc_i}^{(\mu)}$ based on linear model at $\mathbf{s}_0$ and accurately computed worst case point $\mathbf{s}_{wc}^{accurate}$



$s_2$
linearization of $\varepsilon_i$ at $\mathbf{s}_{wc_i}^{(\mu)}$
$\varepsilon_i$
$\mathbf{s}_{wc_i}^{(\mu)}$
$\mathbf{s}_{wc_i}^{(\mu+1)}$
$\mathbf{s}_0$
$\mathbb{T}_s^{N_s}$
$\mathbf{s}_{wc}^{accurate}$
increase
$s_1$

(b) Approximation of worst case point $\mathbf{s}_{wc_i}^{(\mu+1)}$ based on linear model at $\mathbf{s}_{wc_i}^{(\mu)}$ and accurately computed worst case point $\mathbf{s}_{wc}^{accurate}$

Figure 4.4: Approximation of the worst case point for performance gap $\varepsilon_i$ based on linear models at the nominal point $\mathbf{s}_0$ and at a previous approximation of the worst case point $\mathbf{s}_{wc_i}^{(\mu)}$

However, to increase the accuracy of each approximated worst case point, the linearization should be done as close to the solution as possible (cf. Figure 4.4b). For this purpose, the linear model is computed by Algorithm 11 using the worst case points of the previous step. I.e., after computing a new point in the FSQP algorithm (Algorithm 4 on page 63, line 20), the worst case operating point for each performance is approximated by solving the linear optimization problem (Algorithm 11 line 3):

$$\mathbf{o}_{wc_i}^{(\mu+1)} = \arg \max_{\mathbf{o} \in \mathbb{T}_o^{N_o}} \nabla_{\mathbf{o}} \varepsilon_i \left( \mathbf{d}^{(\mu+1)}, \mathbf{o}, \mathbf{s}_{wc_i}^{(\mu)} \right) \Big|_{\mathbf{o}=\mathbf{o}^{(\mu)}}^{T} \cdot \mathbf{o} \tag{4.14}$$

The linear model for performance gap $i$ and for variations in the operating parameters is computed at the design parameter point in the current iteration and at the worst case operating and process parameter point of the previous iteration. The approximation of the worst case points over a rectangular tolerance region – but with a linear model computed at the nominal operating parameters – is also known as *classical worst case analysis* [Gra07].

After the computation of the new worst case operating parameters, the worst case process parameters are approximated by solving (Algorithm 11 line 7):

$$\mathbf{s}_{wc_i}^{(\mu+1)} = \arg \max_{\mathbf{s} \in \mathbb{T}_s^{N_s}} \nabla_{\mathbf{s}} \varepsilon_i \left( \mathbf{d}^{(\mu+1)}, \mathbf{o}_{wc_i}^{(\mu+1)}, \mathbf{s} \right) \Big|_{\mathbf{s}=\mathbf{s}^{(\mu)}}^{T} \cdot \mathbf{s} \tag{4.15}$$

The linear model for performance gap $i$ and for variations in the process parameters is computed at the design parameter and worst case operating parameter point in the current iteration and at the worst case process parameter point of the previous iteration. The approximation of the worst case points over an ellipsoidal tolerance region – but with a linear model computed at the nominal process parameters – is also known as *realistic worst case analysis* [Gra07].

The computation of the worst case points for the operating parameters before the worst case points for the process parameters can be motivated by the expected change of these points: A change in a worst case point for the operating parameters $\mathbf{o}_{wcj}$ which is caused by a – possibly small – alteration of the linear model of performance $j$ means that $\mathbf{o}_{wcj}$ is shifted from one edge of the tolerance box $\mathbb{T}_o^{N_o}$ to another. As a consequence, the influence on performance $j$ and on the worst case point for the process parameters $\mathbf{s}_{wcj}$ may be high. In contrast, a worst case point of the process parameters $\mathbf{s}_{wcj}$ lies on an ellipsoidal boundary curve such that a small alteration of the linear model of performance $j$ causes only a small change in this parameter and a smaller influence on performance $j$ and on the worst case point of the operating parameters $\mathbf{o}_{wcj}$ can be expected. It can be concluded that it is reasonable for the sake of an accurate approximation to compute the linear models with respect to the process parameters and the worst case points for the process parameters after determining the worst case points for the operating parameters.

Once all worst case points are approximated in Algorithm 11, the sensitivity of each performance gap size against variations in the process parameters is computed in line 9

**Algorithm 11:** Computation of Worst Case Sensitivity

```
Input: $\mathbf{d}^{(\mu+1)}$, $\mathbf{o}_{wc1}^{(\mu)}$, ...,$\mathbf{o}_{wcN_\varepsilon}^{(\mu)}$, $\mathbf{s}_{wc1}^{(\mu)}$, ...,$\mathbf{s}_{wcN_\varepsilon}^{(\mu)}$
compute sensitivities against variations in operating conditions                     1
    $$\mathbf{j}_{o,i} = \nabla_{\mathbf{o}} \varepsilon_i \left( \mathbf{d}^{(\mu+1)}, \mathbf{o}, \mathbf{s}_{wci}^{(\mu)} \right) \Big|_{\mathbf{o}=\mathbf{o}^{(\mu)}}$$
    for each performance $i$
foreach $i = 1, ..., N_\varepsilon$ do                                                 2
    //compute worst case point w.r.t. operating parameters
    compute $\mathbf{o}_{wci}^{(\mu+1)} = \arg \max_{\mathbf{o} \in \mathbb{T}_o^{N_o}} \mathbf{j}_{o,i}^T \cdot \mathbf{o}$    3
end                                                                                    4
compute sensitivities against variations in process parameters                       5
    $$\mathbf{j}_{s,i} = \nabla_{\mathbf{s}} \varepsilon_i \left( \mathbf{d}^{(\mu+1)}, \mathbf{o}_{wci}^{(\mu+1)}, \mathbf{s} \right) \Big|_{\mathbf{s}=\mathbf{s}^{(\mu)}}$$
    for each performance $i$
foreach $i = 1, ..., N_\varepsilon$ do                                                 6
    //compute worst case point w.r.t. operating parameters
    compute $\mathbf{s}_{wci} = \arg \max_{\mathbf{s} \in \mathbb{T}_s^{N_s}} \mathbf{j}_{s,i}^T \cdot \mathbf{s}$    7
end                                                                                    8
compute sensitivities against variations in design parameters                        9
    $$\mathbf{j}_{d,i} = \nabla_{\mathbf{d}} \varepsilon_i \left( \mathbf{d}, \mathbf{o}_{wci}^{(\mu+1)}, \mathbf{s}_{wci}^{(\mu+1)} \right) \Big|_{\mathbf{d}=\mathbf{d}^{(\mu+1)}}$$
    for each performance $i$
set $\mathbf{J}_\varepsilon = [\mathbf{j}_{d,1}, ..., \mathbf{j}_{d,N_\varepsilon}]$                                  10
return $\mathbf{o}_{wc1}^{(\mu+1)}$, ...,$\mathbf{o}_{wcN_\varepsilon}^{(\mu+1)}$, $\mathbf{s}_{wc1}^{(\mu+1)}$, ...,$\mathbf{s}_{wcN_\varepsilon}^{(\mu+1)}$, $\mathbf{J}_\varepsilon$       11
```

of Algorithm 11 and the algorithm returns the results to the FSQP algorithm. For the remainder of the FSQP algorithm it is assumed that the change in the worst case points due to manipulations of the design parameters, e.g., during the arc search, can be neglected. Therefore, operating and process parameters are fixed to the latest available worst case values for the remainder of the FSQP iteration.

To enable high parallelization of the simulations, all worst case operating points are approximated in a first step and all worst case process parameters are approximated in a second step. Thus, all simulations required to compute the sensitivity against variations in the operating parameters can be parallelized, all simulations for the computation of the sensitivity against variations in the process parameters can be parallelized, and all simulation to derive the sensitivity against variations in the design parameters can be parallelized as well.

### 4.4.2 Branch-and-Bound for Tolerance Design

In addition to the modification in the FSQP approach, slight changes are also required for the newly developed Branch-and-Bound approach in Section 4: In the inner Branch-and-Bound approach (Algorithm 10 on page 94) the performances are evaluated at discrete design points. However, due to the discretization, the design parameter point is shifted and thus the worst case parameter points can differ from the worst case points computed during the FSQP algorithm. To consider this problem, the approximation of the worst case parameters is refined at each discrete point by an algorithm similar to Algorithm 11 but without computation of the sensitivity against variations in the design parameters. To avoid unnecessary computations, the refined approximation of the worst cases is only computed if the specifications of the performances are fulfilled at the discrete point considering the worst case points computed during the FSQP algorithm[4].

It is worth mentioning that by using Algorithm 11 the approximation of the worst case operating point is a point on the boundary of the operating region $\mathbb{T}_o^{N_o}$ also in case of discrete operating parameters. As a consequence, the result of the approximation is a discrete operating point if the boundaries of this box are chosen so that they are values of the discrete operating parameter set. I.e., no modification of the algorithm is required for discrete operation parameters if assumption 3 holds true. This validates the statement in Chapter 2.1.2 that discrete operating parameters can be treated as continuous parameters in the newly developed approach in this section.

[4]Otherwise any worst case point computed during the FSQP approach violates the corresponding performance specification for the discretized design parameter point. However, the result at the real worst case point must be even worse or equal to this result and the discrete point is rejected as it is no solution for the sizing task.

# Chapter 5

# Model-Based Analog Sizing

The approach presented in Chapter 4 is a highly efficient method to solve a discrete analog sizing task if a circuit can be simulated at each continuous point in a relaxed design space. However, in some analog sizing scenarios the simulations can only be executed at discrete points. Such design scenarios are addressed by the new approach in this chapter. The new method can be classified as a model-based programming approach and consists of a module to construct a model for the sizing task and a module to solve this problem.

The structure of the approach is discussed and motivated in Section 5.1. The model used for the sizing task and the generation of this model is presented in Section 5.2. Section 5.3 shows the algorithm used to compute a solution for the sizing task on this model.

The consideration of variations of process and operating parameters to compute a tolerance design is presented in Section 5.4 so that these parameters can be neglected in Sections 5.1 to 5.3 for the sake of a simple description.

## 5.1 Structure of the New Approach

The required evaluation of points which are not on a predefined discrete grid in approaches as, e.g., in the approach described in Chapter 4, is reasoned by the computation of intermediate points in a continuous domain. It follows that a method to solve problems which can only be evaluated for discrete points must ensure the discreteness of all intermediate points. A mathematical idea for this problem is presented in [ES] where a trust-region based SQP approach is used and a discretization of the intermediate points of the SQP approach in each step is suggested. However, this method assumes equidistant grid points and, what is more, does not guarantee and can not be easily enlarged such that each intermediate solution is c-feasible, as it is required for the analog sizing task.

In this chapter, a new method is introduced for the discrete analog sizing task, which belongs to the class of model-based optimization approaches. The structure of the new approach is shown in Figure 5.1. It consists of two main parts: one to generate and to update the model of the sizing task (Section 5.2) and another one to find a discrete solution on the model (Section 5.3).

A quadratic approximation of each constraint and each performance is used for the model in the new approach. The model is computed based on gradients which can be defined for the discrete problem according to (2.32) on page 26. The quadratic terms of the model are computed using an SR1 update which allows an accurate approximation of convex and non-convex performances and constraints.

The new algorithm to solve the analog sizing task (Algorithm 12 in Section 5.3) computes an improved discrete intermediate solution based on the provided model in each iteration. At this point, the model is refined. The steps to improve the intermediate solution and to refine the model are repeated until the sizing task is solved. This procedure is similar to SQP. However, for an SQP algorithm the performances are merged to a single-objective function and the – mostly convex – quadratic model is computed for the Lagrangian function which is influenced only by the single-objective function and by constraints which are considered as active (cf. Chapter 3.4.2). In contrast, the new approach models each active and inactive constraint and each performance separately. As a consequence, the model in the new approach is typically more accurate – without additional simulation cost – but a nonlinear discrete program must be solved on the model to find a new intermediate point. In contrast, the intermediate points in an SQP approach are computed on a linearly constrained quadratic program.

To find a discrete intermediate solution on the nonlinear model, a modified version of the inner Branch-and-Bound algorithm of the analog sizing approach in Chapter 4 (Section 4.3.3) is used. The modifications can be identified by comparing the structure of the inner Branch-and-Bound approach in Figure 4.2 (page 83) with the structure of the modified algorithm in Figure 5.1. One difference is the use of an FSQP approach in Figure 5.1 instead of the active set approach in Figure 4.2 to compute a continuous solution for the model. The FSQP algorithm is required for the new approach in this chapter because constraints and performances are modeled by possibly non-convex quadratic functions.

In addition, the rounded solution of the FSQP and of the active set approach, respectively, is considered differently in the two structures: In both structures, such a solution candidate is returned as a result if it solves the sizing task. Otherwise, the rounded solution in Chapter 4 is only required if it is equal to the result of the active set algorithm. In contrast, the rounded solution is always used in this chapter if it is identified as a candidate for an intermediate solution. Such a candidate for an intermediate solution is defined as a discrete point where an improvement for the sizing task can be expected. In case the simulation of this points validates that the candidate point improves the

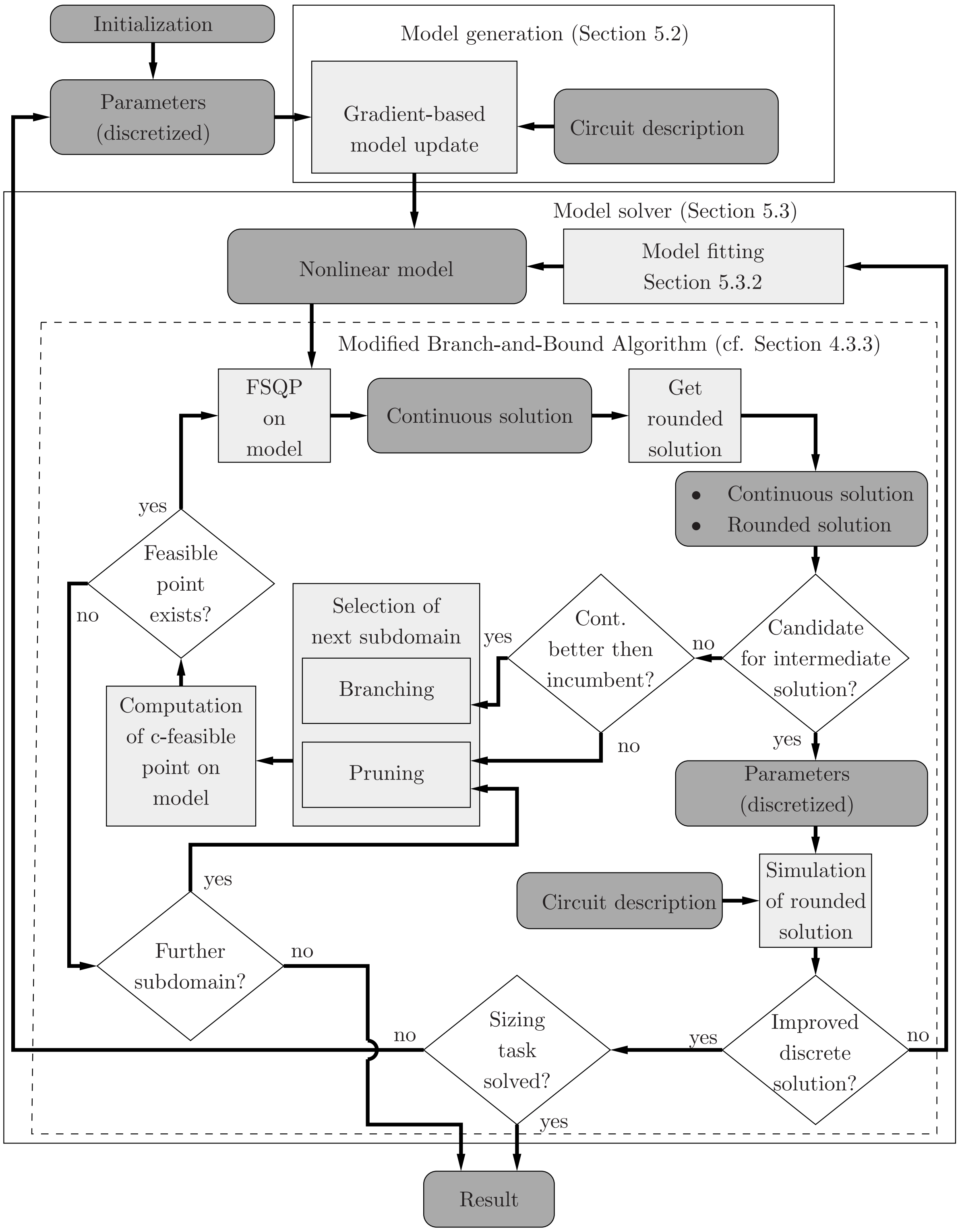

Initialization
Parameters (discretized)
Model generation (Section 5.2)
Gradient-based model update
Circuit description
Model solver (Section 5.3)
Nonlinear model
Model fitting Section 5.3.2
Modified Branch-and-Bound Algorithm (cf. Section 4.3.3)
FSQP on model
Continuous solution
Get rounded solution
• Continuous solution
• Rounded solution
yes
Feasible point exists?
no
Selection of next subdomain
Branching
Pruning
yes
Cont. better then incumbent?
no
Candidate for intermediate solution?
no
yes
Computation of c-feasible point on model
Parameters (discretized)
Simulation of rounded solution
Circuit description
yes
Further subdomain?
no
no
Sizing task solved?
yes
Improved discrete solution?
yes
no
Result

Figure 5.1: Structure of the model-based approach for discrete analog sizing

solution, it can be accepted as new discrete intermediate solution and the nonlinear model can be updated at this point. Otherwise, insufficient accuracy of the model has been detected and a model fitting step (Section 5.3.2) uses the discrete point and the simulation result at this point to improve the model. It can be noted – cf. Section 5.3.1 – that each discrete solution computed by the FSQP approach is a candidate for an improved solution. As a consequence, no update of the incumbent solution is required and the corresponding decision path – which can be found in Figure 4.2 – is left out in this chapter.

No convergence proof can be presented for the new model-based approach for analog sizing. However, the approach has reliably computed a discrete solution for the sizing task in all experiments. Furthermore, the convergence can be made plausible under mild assumptions and enhancements are suggested in case the assumptions are not valid (Section 5.3.3).

## 5.2 Model Generation for the Analog Sizing Task

The choice of a reasonable model is crucial with respect to the performance of the presented algorithm. Several models can be used for approximation in the analog sizing task. However, in many practical cases the performances of a circuit can be sufficiently approximated by a quadratic function. Therefore, a quadratic approximation is used in this thesis. More complex models, e.g., based on radial basis functions, could be used in the presented approach instead of the quadratic function and might result in higher accuracy of the model at the cost of increased runtime. The accuracy of less complex, e.g., linear models – which would lead to an outer approximation approach [ALL] – is typically not sufficient.

SQP approaches, which are related to this new approach, compute one quadratic model of the Lagrangian function, wherein – for an analog sizing task – all performances are considered in a single-objective function and only active constraints are modeled. In contrast, the quadratic model in this new approach is built separately for each constraint and for each performance to achieve high model quality, i.e., $N_c + N_\varepsilon$ different quadratic models are used where $N_c$ is the number of constraints of the sizing task and $N_\varepsilon$ is the number of performance-to-specification gaps (cf. Chapter 2.3.3).

The quadratic approximation computed for the size of a performance gap $\varepsilon_i(\mathbf{d})$ and for a constraint $c_j(\mathbf{d})$ at a point $\mathbf{d}_0$ can be given as

$$\begin{aligned} \varepsilon_i(\mathbf{d}) \approx \widehat{\varepsilon}_i(\mathbf{d}) &= \tfrac{1}{2}(\mathbf{d}-\mathbf{d}_0)^T \mathbf{H}_{\varepsilon,i}(\mathbf{d}-\mathbf{d}_0) + \mathbf{j}_{\varepsilon,i}^T(\mathbf{d}-\mathbf{d}_0) + \varepsilon_{i,0} \\ c_j(\mathbf{d}) \approx \widehat{c}_j(\mathbf{d}) &= \tfrac{1}{2}(\mathbf{d}-\mathbf{d}_0)^T \mathbf{H}_{c,j}(\mathbf{d}-\mathbf{d}_0) + \mathbf{j}_{c,j}^T(\mathbf{d}-\mathbf{d}_0) + c_{j,0} \end{aligned} \tag{5.1}$$

where $\mathbf{H}_{\varepsilon,i} = \nabla^2_{\mathbf{dd}}\varepsilon_i(\mathbf{d}_0)$ and $\mathbf{H}_{c,j} = \nabla^2_{\mathbf{dd}}c_j(\mathbf{d}_0)$ are the Hessian matrices, $\mathbf{j}_{\varepsilon,i} = \nabla_{\mathbf{d}}\varepsilon_i(\mathbf{d}_0)$ and $\mathbf{j}_{c,j} = \nabla_{\mathbf{d}}c_j(\mathbf{d}_0)$ are the sensitivities, and $\varepsilon_{i,0}$ and $c_{j,0}$ are the size of performance gap $i$ and constraint $j$ computed at $\mathbf{d}_0$.

The sensitivities for performances and constraints are approximated using the approach in Chapter 2.3.2 and can be derived based on simulations at discrete points. Also, the performance gap sizes and constraint values can be obtained directly by simulations at a discrete point. However, the computational effort to compute the quadratic term of the approximation of a multi-dimensional function typically is too high (cf. Chapter 3.4.2). Hence, the Hessian matrices in this model-based approach are approximated based on the gradients computed in each iteration step of the optimization process. For this purpose, an SR1 update formula (cf. Chapter 3.4.3) is used which allows the computation of an accurate model for each convex or non-convex function describing a constraint or the size of a performance gap[1].

To refine the model, the sensitivities of the performances are computed each time a new feasible discrete intermediate solution has been computed by the approach in Section 5.3.1. At this point the SR1 update is executed and a new, improved model of the quadratic function can be provided which can be used to compute another improved discrete point.

A critical point in the approximation of the Hessian matrix is its initialization. In literature this problem is typically solved heuristically by initializing the Hessian matrix by an identity matrix. However, the drawback of this choice is that the initialization of the Hessian matrix is convex, although the function which should be approximated might be non-convex. In addition, the initialization of the Hessian matrix by an identity matrix does not consider whether some performance gap sizes and some constraints can be described linearly.

An alternative could be initializing the Hessian matrix with zero and starting the approximation after one step. But in this case the first step, which is computed on a linear model, typically is too large and can lead far away from the initial solution, which has possibly been chosen carefully. To avoid these problems, the curvatures of the functions describing the constraints and the sizes of performance gaps with respect to the parameters are considered:

As mentioned in Chapter 2.3.2, the curvature of each performance – and as a consequence of each function describing a performance gap size or a constraint – with respect to the parameters can be computed requiring little additional effort by using the data already available from the computation of the sensitivities. The computed curvature information can be used directly to initialize the diagonal of the Hessian matrix. However, experiments have shown that it is preferable to limit the initial diagonal values $h_{ii}$ of the Hessian matrices to $-1 \leq h_{ii} \leq 1$ and to cut off smaller and higher values to avoid numerical problems and insufficiently small steps in the first iterations of the algorithm presented in Section 5.3.

[1]The SR1 update can be used to generate indefinite Hessian matrices, which can be considered as a disadvantage in line-search approaches but is an advantage in modern trust region approaches [NW99].

## 5.3 Algorithm for Model-Based Analog Sizing

The new algorithm for model-based analog sizing computes improved discrete intermediate solutions in each iteration using a Branch-and-Bound approach (cf. Section 5.3.1) in combination with a trust region like approach (cf. Section 5.3.2). At the computed intermediate point, the model is refined as discussed in Section 5.2. The two steps are repeated until the algorithm converges.

### 5.3.1 Branch-and-Bound Based Model-Solver

The newly developed algorithm is presented in this section for operating conditions and process parameters fixed to given values. Using a min-max formulation, the optimization problem which must be solved for the sizing task can be defined as a goal attainment optimization problem (cf. (2.51)):

$$\min_{k,\mathbf{d}} k \text{ s.t.: } \quad \begin{aligned} &\mathbf{c}(\mathbf{d}) \geq \mathbf{0} \\ &k - \varepsilon_i(\mathbf{d}) \geq 0;\ i = 1, ..., N_\varepsilon \end{aligned} \tag{5.2}$$

The model-based algorithm which is used to solve this task is shown in Algorithm 12 and discussed below. It is assumed for the algorithm that a discrete initial point can be provided which is feasible with respect to the design constraints.

In each iteration, the quadratic model for each performance gap and for each constraint is computed by the approach in Section 5.2 and a surrogate problem for (5.2) is defined as (Algorithm 12 lines 3 to 5):

$$\min_{\widehat{k},\mathbf{d}} \widehat{k} \text{ s.t. } \quad \begin{aligned} &\widehat{c}_j(\mathbf{d}) \geq 0;\ j = 1, ..., N_c \\ &\widehat{k} - \widehat{\varepsilon}_i(\mathbf{d}) \geq 0;\ i = 1, ..., N_\varepsilon \end{aligned} \tag{5.3}$$

$\widehat{\varepsilon}_i(\mathbf{d})$ and $\widehat{c}_j(\mathbf{d})$ are the quadratic models of the functions describing the performance gap sizes and constraints as defined in (5.1). $\widehat{k}$ is some real valued scalar. (5.3) is still a nonlinear discrete optimization problem with $\mathbf{d} \in \mathbb{D}^N$. However, the problem is explicitly known and, as a consequence, the model can be evaluated for each continuous point and the cost for function evaluations is low.

In this thesis, the surrogate problem is solved (line 6 of Algorithm 12) by a modification of the Branch-and-Bound algorithm in Algorithm 10 on page 94[2]. To adapt the Branch-and-Bound algorithm to the task in (5.3), lines 1 to 2 of Algorithm 10 – i.e., the simulation-based test if a discrete point fulfills the specifications and the corresponding stop criterion – are deleted. In addition, the active set method in line 12 of Algorithm 10 is replaced by the FSQP approach from Section 3.4.5 to consider the stronger non-linearity of the surrogate optimization problem (5.3).

[2]Other solution methods to solve the nonlinear programming task can be used at this point, e.g., to adapt the algorithm to more complex models.

**Algorithm 12:** Model-Based Discrete Sizing

```
Input: discrete domain $\mathbb{D}^N$, discrete feasible point $\mathbf{d}^{(0)}$
set $\mu = 0$; $\epsilon$ = numerical accuracy; $k^{(0)} = \max_{i=1,...,N_\varepsilon} \varepsilon_i\left(\mathbf{d}^{(0)}\right)$    1
while $\left\|\mathbf{d}^{(\mu+1)} - \mathbf{d}^{(\mu)}\right\| \geq \epsilon \wedge k^{(\mu)} > 0$ do    2
    compute values $\varepsilon_i\left(\mathbf{d}^{(\mu)}\right)$, $c_j\left(\mathbf{d}^{(\mu)}\right)$ and sensitivities $\mathbf{j}_{\varepsilon,i}\left(\mathbf{d}^{(\mu)}\right)$ and $\mathbf{j}_{c,j}\left(\mathbf{d}^{(\mu)}\right)$    3
            for $i = 1, ..., N_\varepsilon$, $j = 1, ..., N_c$
    compute quadratic terms $\mathbf{H}_{\varepsilon,i}$ and $\mathbf{H}_{c,j}$ for $i = 1, ..., N_\varepsilon$, $j = 1, ..., N_c$    4
            using an SR1 update
    set optimization problem of the form (5.3)    5
    compute discrete point $\mathbf{d}^{(\mu+1)}$ using (5.3) such that    6
            $\underset{\substack{i=1,...,N_\varepsilon \\ j=1,...,N_c}}{\forall} \widehat{\varepsilon}_i\left(\mathbf{d}^{(\mu+1)}\right) < k^{(\mu)} \wedge \widehat{c}_j\left(\mathbf{d}^{(\mu+1)}\right) \geq 0$
    set $\kappa = 0$    7
    set $m_{\varepsilon,i} = 1$ for $i = 1, ..., N_\varepsilon$ and $m_{c,j} = 1$ for $j = 1, ..., N_c$    8
    set index sets of modified models to $\mathbb{I}^{(\kappa)}_{\varepsilon,\mathrm{mod}} = \emptyset$ and $\mathbb{I}^{(\kappa)}_{c,\mathrm{mod}} = \emptyset$    9
    while $\left\|\mathbf{d}^{(\mu+1)} - \mathbf{d}^{(\mu)}\right\| \geq \epsilon \wedge \underset{\substack{i=1,...,N_\varepsilon \\ j=1,...,N_c}}{\exists} \varepsilon_i\left(\mathbf{d}^{(\mu+1)}\right) \geq k^{(\mu)} \vee c_j\left(\mathbf{d}^{(\mu+1)}\right) < 0$ do    10
        set index sets $\mathbb{I}^{(\kappa+1)}_{\varepsilon,\mathrm{mod}}$ and $\mathbb{I}^{(\kappa+1)}_{c,\mathrm{mod}}$ for modified models according to (5.14)    11
        for each $\varepsilon_i\left(\mathbf{d}^{(\mu+1)}\right) \geq k^{(\mu)}$ set $K_{i,1} = 1.1 \cdot m_{\varepsilon,i}$ (5.17); compute $s_i$ (5.15)    12
        for each $c_j\left(\mathbf{d}^{(\mu+1)}\right) < 0$ set $K_{j,2} = 1.1 \cdot m_{c,j}$ (5.18); compute $s_j$ (5.16)    13
        set modified optimization problem according to (5.19), (5.20)    14
        compute discrete point $\mathbf{d}^{(\mu+1)}$ using (5.20) such that    15
                $\underset{\substack{i=1,...,N_{\varepsilon+} \\ j=1,...,N_{c+}}}{\forall} \widehat{\varepsilon}_{i,+}\left(\mathbf{d}^{(\mu+1)}\right) < k^{(\mu)} \wedge \widehat{c}_{j,+}\left(\mathbf{d}^{(\mu+1)}\right) \geq 0$
        for all models modified in iteration $\kappa$ set $m_{\varepsilon,i} = 2m_{\varepsilon,i}$ or $m_{c,j} = 2m_{c,j}$    16
        set $\kappa = \kappa + 1$    17
    end    18
    set $k^{(\mu+1)} = \max_{i=1,...,N_\varepsilon} \varepsilon_i\left(\mathbf{d}^{(\mu+1)}\right)$    19
    set $\mu = \mu + 1$    20
end    21
return $\mathbf{d}^{(\mu)}$    22
```

However, the discrete point which is computed by Algorithm 10 on the quadratic model does not have to be an optimal solution: The task is finding a discrete solution which improves the objective function of (5.2) rather than solving problem (5.3) optimally[3]. Given the currently best discrete solution $\mathbf{d}^{(\mu)}$ where the model has been built and where the performances have been evaluated by simulations, a candidate for such a discrete point $\mathbf{d}^{(\mu+1)}$ must fulfill (cf. line 6 of Algorithm 12)

$$\underset{\substack{i=1,...,N_\varepsilon\\ j=1,...,N_c}}{\forall} \quad \widehat{\varepsilon}_i\left(\mathbf{d}^{(\mu+1)}\right) < k^{(\mu)} \;\wedge \widehat{c}_j\left(\mathbf{d}^{(\mu+1)}\right) \geq 0 \tag{5.4}$$

$$\text{with } k^{(\mu)} = \max_{i=1,\ldots,N_\varepsilon} \varepsilon_i\left(\mathbf{d}^{(\mu)}\right) = \max_{i=1,\ldots,N_\varepsilon} \widehat{\varepsilon}_i\left(\mathbf{d}^{(\mu)}\right)$$

I.e., $\mathbf{d}^{(\mu+1)}$ must be a better discrete feasible solution for the model than the discrete solution $\mathbf{d}^{(\mu)}$. For this purpose, the Branch-and-Bound algorithm in Algorithm 10 can be initialized with $\mathbf{d}_{\text{inc}} = \mathbf{d}^{(\mu)}$. In this case, the first solution which fulfills the condition for pruning by optimality is a candidate for an improved discrete intermediate solution and for a point where the model should be updated. As a consequence, Algorithm 10 can be stopped to reduce the computational effort for solving the optimization problem on the model if such a point has been found.

The solution candidate which is computed in line 6 of Algorithm 12 is accepted only if it fulfills the constraints and improves the solution of the underlying optimization task (5.2). This task is equivalent to the search for a point where each performance gap size $\varepsilon_i\big(\mathbf{d}^{(\mu+1)}\big)$ in the current iteration is smaller than an upper bound $k^{(\mu)}$ for this value and where all constraint values are positive:

$$\underset{\substack{i=1,...,N_\varepsilon\\ j=1,...,N_c}}{\forall} \quad \varepsilon_i\big(\mathbf{d}^{(\mu+1)}\big) < k^{(\mu)} \;\wedge c_j\left(\mathbf{d}^{(\mu+1)}\right) \geq 0 \tag{5.5}$$

$k^{(\mu)}$ can be considered as an upper bound for the performance gap size, as its value is set to the highest value of a performance gap size $\varepsilon_i\big(\mathbf{d}^{(\mu+1)}\big)$ at the beginning of the algorithm (line 1) and at the end of each iteration step (line 19). Condition (5.5) is tested by the inner while loop (line 10).

For a point which fulfills (5.4), condition (5.5) is fulfilled only if the quadratic model is accurate enough. Otherwise, the inner while loop is entered and the model of the optimization problem is modified by a new trust region like approach which is described in Section 5.3.2. The optimization problem in the inner while loop is alternately solved by the variant of the Branch-and-Bound algorithm described above, and modified until the stop criterion of the inner while loop (line 10 of Algorithm 12) is fulfilled.

[3] An analogy to this procedure can be found in line search approaches where for a given search direction a step of sufficient improvement is computed rather than solving the optimization problem for this direction exactly.

## 5.3.2 Model Fitting

If by the approach in Section 5.3.1 a discrete solution $\mathbf{d}^{(\mu+1)} = \mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$ is computed on the nonlinear model and it does not improve the underlying analog sizing task, it can be concluded that the nonlinear model is not accurate enough. In this case, the model is fitted by the new approach in this subsection.

For the computation of the quadratic model in Algorithm 12, the sensitivities and function values for all performance gaps $\varepsilon_i\big(\mathbf{d}^{(\mu+1)}\big)$ and for all constraints $c_j\left(\mathbf{d}^{(\mu+1)}\right)$ are computed based on simulations. As a consequence, it can be assumed that these values are sufficiently accurate. It can be concluded that the difference which can be found between a value computed on the model and computed by simulation for the same performance gap or constraint at the same point $\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$ can be assigned to the quadratic term of the model. To realize the assignment, $\mathbf{d}_0 = \mathbf{d}^{(\mu)}$ and $\mathbf{d} = \mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$ can be applied to (5.1). With this definition, matrices $\mathbf{S}_i$ can be selected and added to the quadratic models such that

$$\frac{1}{2}\Delta\mathbf{d}^{(\mu)^T}\left(\mathbf{H}_{\varepsilon,i} + \mathbf{S}_i\right)\Delta\mathbf{d}^{(\mu)} + \mathbf{j}_{\varepsilon,i}^T\Delta\mathbf{d}^{(\mu)} + \varepsilon_{i,0} \overset{!}{=} \varepsilon_i\left(\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}\right) \tag{5.6}$$

for each performance and

$$\frac{1}{2}\Delta\mathbf{d}^{(\mu)^T}\left(\mathbf{H}_{c,i} + \mathbf{S}_i\right)\Delta\mathbf{d}^{(\mu)} + \mathbf{j}_{c,i}^T\Delta\mathbf{d}^{(\mu)} + c_{i,0} \overset{!}{=} c_i\left(\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}\right) \tag{5.7}$$

for each constraint. The matrices $\mathbf{S}_i$ are not uniquely defined and no suggestions to define such a matrix can be found in literature. Assuming that $\mathbf{S}_i$ is a symmetric matrix, different methods were tested for the new approach to realize this matrix, e.g., to use a vector $\mathbf{s}_i$ and to construct the matrix following the idea of an SR1 update by the outer product $\mathbf{S}_i = \mathbf{s}_i\mathbf{s}_i^T$. Although the SR1-like method results in good approximations in mathematical test cases, in the practical experiments of this thesis the approximation of $\mathbf{S}_i$ by $\mathbf{S}_i = s_i \cdot \mathbf{I}$ yielded better results and is used, where $s_i$ is a scalar value and $\mathbf{I}$ is the identity matrix.

Setting $\mathbf{S}_i = s_i \cdot \mathbf{I}$ and assuming that $\mathbf{d}^{(\mu+1)} = \mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$ with $\Delta\mathbf{d}^{(\mu)} \neq \mathbf{0}$, $s_i$ can be computed from (5.6) and (5.7) by

$$\tfrac{1}{2}\Delta\mathbf{d}^{(\mu)^T} s_i \cdot \mathbf{I}\Delta\mathbf{d}^{(\mu)} + \tfrac{1}{2}\Delta\mathbf{d}^{(\mu)^T}\mathbf{H}_{\varepsilon,i}\Delta\mathbf{d}^{(\mu)} + \mathbf{j}_{\varepsilon,i}^T\Delta\mathbf{d}^{(\mu)} + \varepsilon_{i,0} \overset{!}{=} \varepsilon_i\left(\mathbf{d}^{(\mu+1)}\right) \tag{5.8}$$

$$\Rightarrow \tfrac{s_i}{2}\Delta\mathbf{d}^{(\mu)^T}\mathbf{I}\Delta\mathbf{d}^{(\mu)} + \widehat{\varepsilon}_i\left(\mathbf{d}^{(\mu+1)}\right) \overset{!}{=} \varepsilon_i\left(\mathbf{d}^{(\mu+1)}\right) \tag{5.9}$$

$$\Rightarrow s_i = \frac{2\left(\varepsilon_i\left(\mathbf{d}^{(\mu+1)}\right) - \widehat{\varepsilon}_i\left(\mathbf{d}^{(\mu+1)}\right)\right)}{\Delta\mathbf{d}^{(\mu)^T}\Delta\mathbf{d}^{(\mu)}} \tag{5.10}$$

for performance gaps and

$$s_i = \frac{2\left(c_i\left(\mathbf{d}^{(\mu+1)}\right) - \widehat{c}_i\left(\mathbf{d}^{(\mu+1)}\right)\right)}{\Delta\mathbf{d}^{(\mu)^T}\Delta\mathbf{d}^{(\mu)}} \tag{5.11}$$

for constraints.

The fitting method is applied in Algorithm 12 only for performance gaps which would cause an increase of the objective function value in (5.2) and for constraints which would be violated if the point $\mathbf{d}^{(\mu+1)} = \mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$ were accepted as new intermediate solution, i.e., for performance gaps and constraints with

$$\varepsilon_i\left(\mathbf{d}^{(\mu+1)}\right) \geq k^{(\mu)};\ i = 1, ..., N_\varepsilon \tag{5.12}$$

$$c_j\left(\mathbf{d}^{(\mu+1)}\right) < 0;\ j = 1, ..., N_c \tag{5.13}$$

All other models are considered sufficiently accurate. The indexes of the models in (5.3) which should be modified are stored in an index set for performance gaps $\mathbb{I}_{\varepsilon,\text{mod}}^{(\kappa+1)}$ and an index set for constraints $\mathbb{I}_{c,\text{mod}}^{(\kappa+1)}$, respectively (line 11 of Algorithm 12). I.e., the index sets in iteration $\kappa + 1$ are defined as

$$\begin{aligned} \mathbb{I}_{\varepsilon,\text{mod}}^{(\kappa+1)} &= \mathbb{I}_{\varepsilon,\text{mod}}^{(\kappa)} \cup \left\{ i \left| \varepsilon_i\left(\mathbf{d}^{(\mu+1)}\right) \geq k^{(\mu)} \right.\right\} \\ \mathbb{I}_{c,\text{mod}}^{(\kappa+1)} &= \mathbb{I}_{c,\text{mod}}^{(\kappa)} \cup \left\{ j \left| c_j\left(\mathbf{d}^{(\mu+1)}\right) < 0 \right.\right\} \end{aligned} \tag{5.14}$$

where $\mathbf{d}^{(\mu+1)}$ changes in each iteration $\kappa$ of the inner while loop and, hence, $\mathbb{I}_{\varepsilon,\text{mod}}^{(\kappa+1)}$ and $\mathbb{I}_{c,\text{mod}}^{(\kappa+1)}$ may change in each iteration $\kappa$.

Practical experiments performed in this thesis have shown that the discrete optimum of the quadratically approximated optimization problem converges against a discrete feasible solution which causes an improvement of the underlying sizing task if the described fitting method is applied iteratively. However, the number of iterations can be high. To overcome this problem, in this thesis additional penalty values were introduced in (5.10) and (5.11) and are used to compute a correction for the Hessian matrix. The revised computation formulas for Algorithm 12 are defined for performance gaps and constraints as (lines 12 and 13)

$$s_i = \frac{2\left(K_{i,1} \cdot \varepsilon_i\left(\mathbf{d}^{(\mu+1)}\right) - \widehat{\varepsilon}_i\left(\mathbf{d}^{(\mu+1)}\right)\right)}{\Delta\mathbf{d}^{(\mu)^T}\Delta\mathbf{d}^{(\mu)}} \tag{5.15}$$

and

$$s_j = \frac{2\left(K_{j,2} \cdot c_j\left(\mathbf{d}^{(\mu+1)}\right) - \widehat{c}_j\left(\mathbf{d}^{(\mu+1)}\right)\right)}{\Delta\mathbf{d}^{(\mu)^T}\Delta\mathbf{d}^{(\mu)}} \tag{5.16}$$

The values of $K_{i,1}$ and $K_{j,2}$ are chosen in such a way that the absolute values of the modified models for performance gaps and constraints at $\mathbf{d}^{(\mu)} + \Delta\mathbf{d}^{(\mu)}$ are increased compared to the values computed by simulation. I.e., considering that the model is only modified if $\varepsilon_i\left(\mathbf{d}^{(\mu+1)}\right) > 0$ – otherwise all performances fulfill the specifications and the algorithm can be stopped – and if $c_j\left(\mathbf{d}^{(\mu+1)}\right) < 0$, the values of $K_{i,1}$ and $K_{j,2}$ must be greater or equal to one. In this thesis the values are initialized from experience with $K_{i,1} = K_{j,2} = 1.1$.

The penalty value for each individual model is increased by a factor $m_{\varepsilon,i}$ and $m_{c,i}$, respectively, if the same model must be fitted multiple times according to condition (5.12), (5.13). I.e., in the inner loop of Algorithm 12 $K_{i,1}$ and $K_{j,2}$ become (lines 12 and 13)

$$K_{i,1} = 1.1 \cdot m_{\varepsilon,i} \tag{5.17}$$

$$K_{j,2} = 1.1 \cdot m_{c,i} \tag{5.18}$$

where the value of each $m_{\varepsilon,i}$ and of each $m_{c,i}$ is initialized by one in each iteration $\mu$ (line 8) and doubled each time when the corresponding model is modified (line 16). This procedure has shown fast convergence, i.e., a low number of required modification steps, for the inner loop (line 10).

In the presence of nonlinear constraints, it is also suitable to modify the constraints so that a certain safety margin can be achieved. For this purpose, the constraint value $c_{j,0}$ of the modified constraints at the point $\Delta\mathbf{d}^{(\mu)} = \mathbf{0}$ is set to $\frac{c_{j,0}}{K_{j,2}}$. The resulting modified optimization problem which must be solved in the inner loop (line 15) can be given with these modifications, $\Delta\mathbf{d}^{(\mu)} = \mathbf{d}^{(\mu+1)} - \mathbf{d}^{(\mu)}$, (5.1), (5.3), (5.14), (5.15), and (5.16) as:

$$\begin{aligned}
\min_{\widehat{k},\Delta\mathbf{d}^{(\mu)}} \quad & \widehat{k} \\
\text{s.t.} \quad & \tfrac{1}{2}\Delta\mathbf{d}^{(\mu)^T}\mathbf{H}_{c,j}\Delta\mathbf{d}^{(\mu)} + \mathbf{j}_{c,j}^T\Delta\mathbf{d}^{(\mu)} + c_{j,0} \geq 0;\ j = 1,...,N_c \\
& \tfrac{1}{2}\Delta\mathbf{d}^{(\mu)^T}\left(\mathbf{H}_{c,j} + s_j\mathbf{I}\right)\Delta\mathbf{d}^{(\mu)} + \mathbf{j}_{c,j}^T\Delta\mathbf{d}^{(\mu)} + \tfrac{c_{j,0}}{K_{j,2}} \geq 0;\ j \in \mathbb{I}_{c,\text{mod}}^{(\kappa+1)} \\
& \widehat{k} - \tfrac{1}{2}\Delta\mathbf{d}^{(\mu)^T}\mathbf{H}_{\varepsilon,i}\Delta\mathbf{d}^{(\mu)} - \mathbf{j}_{\varepsilon,i}^T\Delta\mathbf{d}^{(\mu)} - \varepsilon_{i,0} \geq 0;\ i = 1,...,N_\varepsilon \\
& \widehat{k} - \tfrac{1}{2}\Delta\mathbf{d}^{(\mu)^T}\left(\mathbf{H}_{\varepsilon,i} + s_i\mathbf{I}\right)\Delta\mathbf{d}^{(\mu)} - \mathbf{j}_{\varepsilon,i}^T\Delta\mathbf{d}^{(\mu)} - \varepsilon_{i,0} \geq 0;\ i \in \mathbb{I}_{\varepsilon,\text{mod}}^{(\kappa+1)}
\end{aligned} \tag{5.19}$$

Assigning all models and modified models for performance gaps in a certain iteration of the inner loop of Algorithm 12 as $\widehat{\varepsilon}_{i,+}\left(\mathbf{d}^{(\mu+1)}\right)$, with $i = 1,...,N_\varepsilon + \left|\mathbb{I}_{\varepsilon,\text{mod}}^{(\kappa+1)}\right| = 1,...,N_{\varepsilon+}$, and all models and modified models of constraints in a certain iteration of the inner loop of Algorithm 12 as $\widehat{c}_{j,+}\left(\mathbf{d}^{(\mu+1)}\right)$, with $j = 1,...,N_c + \left|\mathbb{I}_{c,\text{mod}}^{(\kappa+1)}\right| = 1,...,N_{c+}$, (5.19) can be rewritten as

$$\begin{aligned}
\min_{\widehat{k},\mathbf{d}} \widehat{k} \text{ s.t. } \quad & \widehat{c}_{j,+}\left(\mathbf{d}\right) \geq 0;\ j = 1,...,N_{c+} \\
& \widehat{k} - \widehat{\varepsilon}_{i,+}\left(\mathbf{d}\right) \geq 0;\ i = 1,...,N_{\varepsilon+}
\end{aligned} \tag{5.20}$$

It can be noted that for each $\Delta\mathbf{d}^{(\mu)}$ the modified model of the constraints is smaller than the model given initially as $K_{j,2} > 1$ and as $s_j$ is negative if it is computed by (5.16) and if (5.13) is fulfilled. The modified models for all performance gap sizes are for each $\Delta\mathbf{d}^{(\mu)}$ greater or equal to the value of the model given initially as $s_i$ is positive if it is computed by (5.15) and if (5.12) is fulfilled. I.e., each model given initially is dominated by its modification. As a consequence, a model given initially can be dropped from the optimization problem if a modification of this model is considered in the optimization task. Therefore, the number of models for constraints and performance gaps in (5.19) can be reduced to $N_c$ and $N_\varepsilon$, respectively, by deleting each model given initially as soon as it has been modified.

The modified models are only used to determine a feasible intermediate solution $\mathbf{d}^{(\mu+1)}$ which improves the performances of the underlying analog sizing task. I.e., all modified models are discarded after the inner while loop, and the SR1 update in line 4 is computed based on the models computed by the SR1 update in the previous iteration. This procedure guarantees that the convergence properties of the SR1 update are not affected.

### 5.3.3 Discussion of the Algorithm

The new approach which was presented in Algorithm 12 belongs to the class of model-based algorithms and uses a trust region like approach to guarantee feasible intermediate solutions which improve the solution of the underlying problem. Simulations are required to test the computed intermediate solutions in line 10 and to compute a sensitivity in line 3. The simulations for the computation of a sensitivity can be fully parallelized.

Although no convergence proof can be given for the approach, the convergence in practical cases can be made plausible under the assumption that the use of quadratic models and of local optimization on the nonlinear model is sufficient: It has been proved for Quasi-Newton and SQP approaches that the quadratic model computed by the SR1 update converges (locally) against the underlying problem. This must also hold true for the new approach if the number of computed intermediate solutions is sufficiently high and if the quadratic model can be used. As a consequence, the discrete optimum of the underlying task must be the discrete optimum of the model as soon as the quadratic model is accurate enough. However, such a solution of the optimization problem defined by the models can be found by the approach presented if the local optimum of the relaxed optimization problem – which is computed by an FSQP approach – is sufficient.

The feasibility and the improvement of each intermediate step is guaranteed by the trust region like approach in the inner while loop of Algorithm 12. It could be assumed that the trust region like approach prevents the algorithm from converging against a solution of the analog sizing task if step size gets too small, particularly as – in contrast to continuous approaches – the step can be non-zero but smaller than the distance to the next discrete point. Such a case could occur if a strongly nonlinear constraint prevents the algorithm from further improvement or if no improvement can be found for any performance. However, the presented approach tries to improve the quadratic model in this case. Therefore, it can be assumed that the algorithm can only get stuck at a constraint if the quadratic model is not sufficient, i.e., if one of the assumptions made for the convergence is violated. In case the algorithm cannot find a solution for the problem, any step to a feasible discrete point can be used so that the model can be further improved. A method to realize such a minimum step was implemented in this thesis. However, in practical experiments no case was observed where it was required.

Practical experiments have shown that – due to the non-convex model – a new intermediate point may be far away from the previous point. This may slow down the convergence as the quadratic model might not be valid at such a new point. As a consequence, a limitation of the maximum step size is realized in this thesis, which limits the maximum step length to at most ten percent of the parameter range. Parameters which can take only a low number of discrete points are excluded from this limitation.

If the quadratic model or the local solution for the relaxed model is not sufficient, the modularity of the algorithm allows replacing the corresponding parts easily. Especially the FSQP approach can be replaced easily, e.g., by global stochastic solvers which typically require many function evaluations and can benefit most from the low computational cost of the model evaluation. However, for all results in Chapter 6 quadratic models and local optimizations on the models using FSQP were sufficient.

## 5.4 Model-Based Tolerance Design

The model-based approach described in Section 5.3 computes a discrete solution for nominal operating and process conditions. In this section, it is enlarged to consider variations in operating and process parameters and for the computation of a tolerance design as defined in Chapter 2.5.4. This task is equivalent to the task of finding a design parameter point where the performance specifications are fulfilled at the worst case points.

The first additional requirement for computing a tolerance design by the model-based approach is considering operating and process parameters in the model. Assuming that an approximation of the worst case points is available for each performance, the quadratic models for the performances introduced in Section 5.2 are extended to

$$\begin{aligned}\varepsilon_i(\mathbf{p}) \approx \widehat{\varepsilon}_i(\mathbf{p}) = \tfrac{1}{2}\left(\mathbf{p}-\mathbf{p}_{wc,i}\right)^T \mathbf{H}_{\varepsilon,i}\left(\mathbf{p}-\mathbf{p}_{wc,i}\right) + \mathbf{j}_{\varepsilon,i}^T\left(\mathbf{p}-\mathbf{p}_{wc,i}\right) + \varepsilon_{i,wc}\\ \text{with } \mathbf{p}^T = \left[\mathbf{d}^T, \mathbf{o}^T, \mathbf{s}^T\right]^T\\ \text{and } \mathbf{p}_{wc_i}^T = \left[\mathbf{d}_0^T, {\mathbf{o}_{wc_i}}^T, {\mathbf{s}_{wc_i}}^T\right]\end{aligned} \tag{5.21}$$

This quadratic model to compute a tolerance design can be constructed iteratively in lines 3 and 4 of Algorithm 12 if the sensitivities in line 3 are computed with respect to design, operating, and process parameters.

Using the quadratic model, the worst case points with respect to operating and process parameters can be approximated in iteration $\mu+1$ by

$${\mathbf{o}_{wc_i}}^{(\mu+1)} = \arg\max_{\mathbf{o}\in\mathbb{T}_o^{N_o}} \widehat{\varepsilon}_i\left(\mathbf{d}^{(\mu)}, \mathbf{o}, {\mathbf{s}_{wc_i}}^{(\mu)}\right) \tag{5.22}$$

for operating parameters and by

$${\mathbf{s}_{wc_i}}^{(\mu+1)} = \arg\max_{\mathbf{s}\in\mathbb{T}_s^{N_s}} \widehat{\varepsilon}_i\left(\mathbf{d}^{(\mu)}, {\mathbf{o}_{wc_i}}^{(\mu)}, \mathbf{s}\right) \tag{5.23}$$

**Algorithm 13:** Computation of Worst Case Points on Model

```
Input: $\mathbf{d}^{(\mu)}$, $\mathbf{o}_{wc1}^{(\mu)}$, ...,$\mathbf{o}_{wcN_\varepsilon}^{(\mu)}$, $\mathbf{s}_{wc1}^{(\mu)}$, ...,$\mathbf{s}_{wcN_\varepsilon}^{(\mu)}$
set $\nu = 0$, $\epsilon$ =numerical accuracy                                              1
set $\mathbf{o}_{wci}^{(\mu,\nu)} = \mathbf{o}_{wci}^{(\mu)}$, $\mathbf{s}_{wci}^{(\mu,\nu)} = \mathbf{s}_{wci}^{(\mu)}$ for $i = 1, ..., N_\varepsilon$      2
for $i = 1, ..., N_\varepsilon$ do                                                          3
    repeat                                                                                  4
        compute Sensitivity $\mathbf{j}_{o,i}^T = \nabla_{\mathbf{o}} \widehat{\varepsilon}_i \left(\mathbf{d}^{(\mu)}, \mathbf{o}, \mathbf{s}_{wci}^{(\mu,\nu)}\right)\Big|_{\mathbf{o}=\mathbf{o}_{wci}^{(\mu,\nu)}}$      5
        compute Sensitivity $\mathbf{j}_{s,i}^T = \nabla_{\mathbf{s}} \widehat{\varepsilon}_i \left(\mathbf{d}^{(\mu)}, \mathbf{o}_{wci}^{(\mu,\nu)}, \mathbf{s}\right)\Big|_{\mathbf{s}=\mathbf{s}_{wci}^{(\mu,\nu)}}$      6
        compute $\mathbf{o}_{wci,lin} = \arg \max_{\mathbf{o} \in \mathbb{T}_o^{N_o}} \mathbf{j}_{o,i}^T \cdot \mathbf{o}$      7
        compute $\mathbf{s}_{wci,lin} = \arg \max_{\mathbf{s} \in \mathbb{T}_s^{N_s}} \mathbf{j}_{s,i}^T \cdot \mathbf{s}$      8
        compute sufficient large step size $t_i$ using line search such that               9
            $\widehat{\varepsilon}_i \left(\mathbf{d}^{(\mu)}, \mathbf{o}_{wci}^{(\mu,\nu)}, \mathbf{s}_{wci}^{(\mu,\nu)}\right) < \widehat{\varepsilon}_i \left(\mathbf{d}^{(\mu)}, \mathbf{o}_{wci}^{(\mu,\nu+1)}, \mathbf{s}_{wci}^{(\mu,\nu+1)}\right)$ with
            $\mathbf{o}_{wci}^{(\mu,\nu+1)} = \mathbf{o}_{wci}^{(\mu,\nu)} + t_i \left(\mathbf{o}_{wci,lin} - \mathbf{o}_{wci}^{(\mu,\nu)}\right)$
            $\mathbf{s}_{wci}^{(\mu,\nu+1)} = \mathbf{s}_{wci}^{(\mu,\nu)} + t_i \left(\mathbf{s}_{wci,lin} - \mathbf{s}_{wci}^{(\mu,\nu)}\right)$
        set $\nu = \nu + 1$                                                                 10
    until $\left\|\mathbf{o}_{wci}^{(\mu,\nu)} - \mathbf{o}_{wci}^{(\mu,\nu-1)}\right\| < \epsilon \wedge \left\|\mathbf{s}_{wci}^{(\mu,\nu)} - \mathbf{s}_{wci}^{(\mu,\nu-1)}\right\| < \epsilon$      11
    set $\mathbf{o}_{wci}^{(\mu+1)} = \mathbf{o}_{wci}^{(\mu,\nu)}$, $\mathbf{s}_{wci}^{(\mu+1)} = \mathbf{s}_{wci}^{(\mu,\nu)}$ for $i = 1, ..., N_\varepsilon$      12
end                                                                                         13
return $\mathbf{o}_{wc1}^{(\mu+1)}$, ..., $\mathbf{o}_{wcN_\varepsilon}^{(\mu+1)}$, $\mathbf{s}_{wc1}^{(\mu+1)}$, ..., $\mathbf{s}_{wcN_\varepsilon}^{(\mu+1)}$      14
```

for process parameters.

As the Hessian matrix for each performance constructed by the SR1 update formula may be non-convex, the solution of this optimization problem must be computed by nonlinear optimization. It is assumed within this thesis that a local optimization is sufficient to solve this task. For this purpose, Algorithm 13 is applied.

Linear worst case analyses are used to compute the worst case points for operating and process parameters with respect to each performance in lines 5 to 8 (cf. (4.14), (4.15)). However, the linearly approximated worst case points are only accepted if they result in model values which are worse than the values at the previous approximations of these points. Otherwise a line search is used in line 9 to compute a new approximation of the worst case points. At the new approximation of the worst case point again a linear worst case analysis is used and the approximation is improved until the algorithm converges.

It is worth noting that – due to the quadratic approximation and in contrast to the approach in Section 4.4 – the worst case points might not lie on the bounds of the operating region and of the tolerance domain for process parameters. Therefore, the approach can be expected to be more general than the linear approximation in Section 4.4. However, in the presence of multiple local maxima in the tolerance region or if discrete operating parameters should be considered, a more complex search algorithm than the one in Algorithm 13 might be required.

To extend Algorithm 12 for the computation of a tolerance design, the approximation of the worst case points is computed if the model has been updated by the SR1 approach (line 5 of Algorithm 12). Afterward, the worst case parameters are fixed to the new computed values and the parameters of the quadratic models can be reduced to the design parameters. The Branch-and-Bound approach is used without modifications to find a discrete solution on the model, i.e., lines 6 to 18 of Algorithm 12 remain unchanged and operate only on design parameters. Especially, the modification of the quadratic model in the inner while loop is computed with exclusive consideration of design parameters, i.e., only the sub-matrix of the Hessian matrix belonging to the design parameters is modified.

After the computation of a new design point $\mathbf{d}^{(\mu+1)}$ (line 19 of Algorithm 12), the worst case points are again updated using Algorithm 13 at the new design point $\mathbf{d}^{(\mu+1)}$ but using the initial model for the iteration. This second update ensures that the new sensitivities against variations in operating and process conditions are computed at the expected worst case points after the step in the design parameters. In addition, it reduces the risk of a design parameter point being accepted as a solution although the computed sizing does not provide the predefined yield.

During the algorithm, the worst case points are computed before and after the simulation-based computation of the sensitivities. Alternatively, the worst case points for the quadratic model could be computed exactly for each design parameter point which is considered during the FSQP approach. Also, the modification of the FSQP approach presented in Section 4.4 could be used to approximate the worst case points successively with the computation of a new design parameter point on the model. However, experiments during this thesis showed that both alternative approaches involve a significantly increased runtime to solve the numerical model. At the same time, no reduction in the number of required iterations and in the number of required simulations was observed.

To ensure a sufficient accuracy of the approximated worst case points, a small number of additional steps is used in this thesis at the end of the algorithm if the worst case points did not converge during the sizing approach. In these steps, no modification of the design parameters is computed but the sensitivities of the performances against variations in operating and process parameters are computed by simulations, the quadratic models are refined, and the worst case points are recalculated using Algorithm 12.

The model-based approach presented in this section can be considered as the first deterministic analog sizing approach which requires simulations only at discrete points and allows the computation of a sizing with discrete parameters at the side with variations in operating and process parameters. The experimental results in the next chapter show that – predominantly due to the time required to solve the nonlinear model by Branch-and-Bound – the runtime of the algorithm is longer than the runtime of the approach presented in Section 4. However, the runtime is still acceptable for practical problems.

# Chapter 6

# Simulation Results

Two new deterministic approaches to size analog circuits considering discrete parameters were presented in Chapter 4 and 5. These new approaches were implemented in Python 2.X [PYT12] and are applied to simulation-based experiments in this chapter. Four different amplifiers, two differential amplifiers – the Miller amplifier in Section 6.1.2 and the low voltage amplifier in Section 6.1.3 – a sense amplifier (Section 6.1.4), and a low noise amplifier (Section 6.1.5) are used to demonstrate the functionality and effectiveness of the new approaches.

The setup for the experiments is shown in Section 6.1 together with the circuits to be sized. In Sections 6.2 and 6.3 the results for the computation of a nominal and of a tolerance sizing are analyzed with respect to solution quality and runtime, and the efficiency and efficacy of the new approaches is clearly shown.

## 6.1 Experimental Setup

In the first part of this section, the general setup of the experiments is shown. Afterward, the considered amplifiers are introduced and the corresponding performance specifications, constraints, and design parameters are outlined as well as the operating conditions and process variations considered.

### 6.1.1 General Setup

The circuits in this chapter are designed in a $180nm$ CMOS technology. For all lengths and widths of the transistors a $10nm$ manufacturing grid is assumed. The transistor lengths are selected in the range from $300nm$ to $1\mu m$, transistor widths can be scaled

Table 6.1: Overview of design parameter definitions in the experiments

| Device Type | Parameter Type | Minimum Value | Maximum Value | Grid |
|---|---|---|---|---|
| transistor | length | $300nm$ | $1\mu m$ | $10nm$ |
| | width | $300nm$ | $10\mu m$ | $10nm$ |
| | multiplier | 1 | 100 | 1 |
| inductor | inner diameter | $20\mu m$ | $200\mu m$ | $0.25\mu m$ |
| | turn width | $2\mu m$ | $50\mu m$ | $0.25\mu m$ |
| | space between turns | $1\mu m$ | $5\mu m$ | $0.25\mu m$ |
| | number of turns | 1.5 | 20 | 0.5 |
| | inductor sides | 4 | 8 | 2 |
| capacitor | capacitance | $100fF$ | $100pF$ | continuous |
| resistor | resistance | $100\Omega$ | $10k\Omega$ | continuous |
| voltage source | voltage | 0 | $V_{DD}$ | continuous |

from $300nm$ to $10\mu m$. In addition, transistor multipliers were used and considered in the range from 1 to 100. Considering multipliers and a manufacturing grid for lengths and widths allows a layout-friendly sizing of the transistors because rounding can be avoided if the edges of the transistor gates are to be snapped onto a grid or if the transistors are to be realized as multi-finger or common centroid structures.

The capacitors, resistors, and voltage sources, whose values were considered as design parameters in the sizing task, were assumed to be continuously scalable ideal devices. The parameter range for capacitors and resistors was defined from $0.1pF$ to $100pF$ and from $100\Omega$ to $10k\Omega$, respectively. Voltage sources were scaled depending on the supply voltage $V_{DD}$ from 0% to 100% of $V_{DD}$.

For the inductors in Section 6.1.3 a realistic inductor model according to [PFC09] was used. This inductor model allows a design of the inductor with respect to inner diameter, width of the winding turns, spacing between the turns, number of turns, and with respect to the number of sides of the inductor in the later layout. The first three parameters were defined in a range from $20\mu m$ to $200\mu m$ for the inner diameter, from $2\mu m$ to $50\mu m$ for the width of the turns, and from $1\mu m$ to $5\mu m$ for the space between the turns. For each of the three parameters a $0.25\mu m$ grid was used, as suggested by [PFC09]. The winding turns were scaled in half turns from 1.5 to 20. For the number of sides in the layout step squared, hexagonal, and octagonal designs were considered, i.e., the inductor was allowed to be implemented with 4, 6, or 8 sides. The influence of the inductor layout configuration can be computed using a look-up table. As a consequence, no intermediate solutions can be simulated and the number of sides can only be considered by the approach in Chapter 5.

All design parameters for the different devices are listed in Table 6.1.

For the sizing of all circuits in this chapter, a three-step approach was used to compute the sizing:

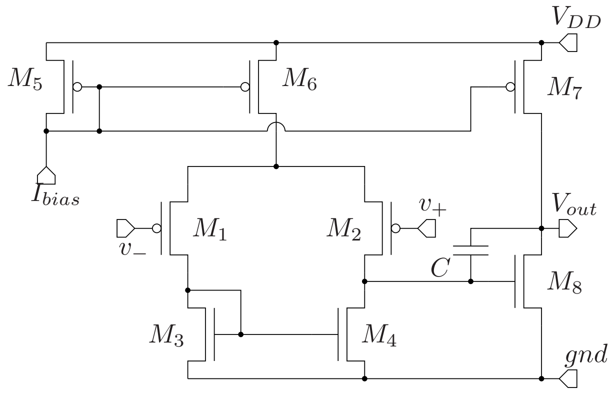


Figure 6.1: Miller amplifier

Table 6.2: Design parameters for the sizing of the Miller amplifier

| Device | $M_1$ | $M_2$ | $M_5$ | $M_6$ | $M_7$ | $M_3$ | $M_4$ | $M_8$ | $C$ |
|---|---|---|---|---|---|---|---|---|---|
| Lengths | $l_p$ | | | | | $l_n$ | | | — |
| Widths | $w_1$ | | $w_5$ | | | $w_3$ | | $w_8$ | — |
| Multipliers | $m_1$ | | $m_5$ | $m_6$ | $m_7$ | $m_3$ | | $m_8$ | — |
| Capacitances | — | | | | | | | | $C_{Miller}$ |

1. In the first step a pre-sizing was computed which guaranteed that the initial point for the subsequent steps was a discrete feasible point. However, each algorithm considered found such an initial point within a few seconds such that the results have no significance and this step is neglected in the remainder of this chapter.
2. A nominal design was computed, i.e., a discrete solution was computed where the performance specifications and constraints were fulfilled for nominal operating and process parameters.
3. Finally, a tolerance design was computed for each circuit starting from a sizing computed during the nominal design step. For this purpose, a predefined yield requirement at the worst case operating points was set to a minimum of 99.87% ($3\sigma$) for each performance.

The algorithms were parallelized up to 16 times and the experiments were run on an 8 Core Intel R Xeon R X5500 2.67 GHz CPU with hyper-threading.

### 6.1.2 Miller Amplifier

In this section the setup of the Miller amplifier in Figure 6.1 is shown. To define reasonable design parameters, the structure of the amplifier should be considered initially.

The first structure considered is the differential pair built by transistors $M_1$ and $M_2$. To guarantee the functionality of the circuit, the lengths, widths, and multipliers of

Table 6.3: Operating parameters for the sizing of the Miller amplifier

| Parameter | Minimum | Nominal | Maximum |
|---|---|---|---|
| Temperature $T$ in $°C$ | -40 | 27 | 125 |
| Supply Voltage $V_{DD}$ in $V$ | 2.3 | 2.5 | 2.7 |
| Bias Current $I_{bias}$ in $\mu A$ | 9 | 10 | 11 |
| Load Capacitance $C_L$ in $pF$ | 15 | 20 | 25 |

these two transistors were equalized. For the same reason, the drain currents of the two transistors in the current mirror $M_3$-$M_4$ should be equal and the lengths, widths, and multipliers of $M_3$ and $M_4$ were matched. For the current mirror bank $M_6$-$M_7$-$M_8$ the transistor lengths must be equal to guarantee functionality. In addition, the widths of the three transistors were matched to allow a multi-finger or common centroid structure of the current mirror bank in a subsequent layout step without using additional rounding operations. Thus, the relation of the drain currents was determined only by using three different multipliers. In addition to this structure-based reduction of the parameters, the lengths of all PMOS transistors and the lengths of all NMOS transistors were set equal, such that the total number of parameters summed up to 13 if the width and multiplier of transistor $M_8$ and the Miller capacitance $C_{Miller}$ were considered as further parameters. The $N_c = 1$ continuous and $N_d = 12$ discrete design parameters are summarized in Table 6.2.

Variations of the operating temperature $T$, the supply voltage $V_{DD}$, the bias current $I_{bias}$, and the load capacitance $C_L$ were considered operating parameters for the sizing task. The specified nominal values, lower and upper bounds for the operating parameters are shown in Table 6.3. In addition, normally distributed global variations of the oxide thickness, of the electron mobility for NMOS and PMOS transistors, and of the threshold voltage for NMOS and PMOS transistors were considered as well as normally distributed local variations of electron mobility and threshold voltage for each transistor. Thus, 5 global process parameters and 16 local process parameters (also referred to as mismatch parameters) were considered in this sizing task.

Besides the equality constraints used above to reduce the number of free parameters, 57 sizing constraints were found for the circuit, using the approach in [MGS08]. In addition, the eight performances listed in Table 6.4 were specified for the sizing of the amplifier.

### 6.1.3 Low Voltage Amplifier

The second amplifier considered in this section is the low voltage amplifier introduced by [Mar98] in Figure 6.2. For this experiment, one length was considered for all transistors. As with the Miller amplifier, the structure of the amplifier was considered to define the design parameters.

Table 6.4: Performance specifications for Miller amplifier; $f_T$ is the transit frequency, $\varphi_R$ is the phase margin, $SR+$ and $SR-$ are slew rate rising and falling

| Performance | Specification |
|---|---|
| Power $[mW]$ | $< 5$ |
| PSRR $[dB]$ | $> 140$ |
| Gain $[dB]$ | $> 75$ |
| CMRR $[dB]$ | $> 120$ |
| $f_T$ $[MHz]$ | $> 10$ |
| $\varphi_R$ $[°]$ | $> 60$ |
| $\lvert$SR$-\rvert$ $[\frac{V}{\mu s}]$ | $> 8$ |
| SR+ $[\frac{V}{\mu s}]$ | $> 8$ |

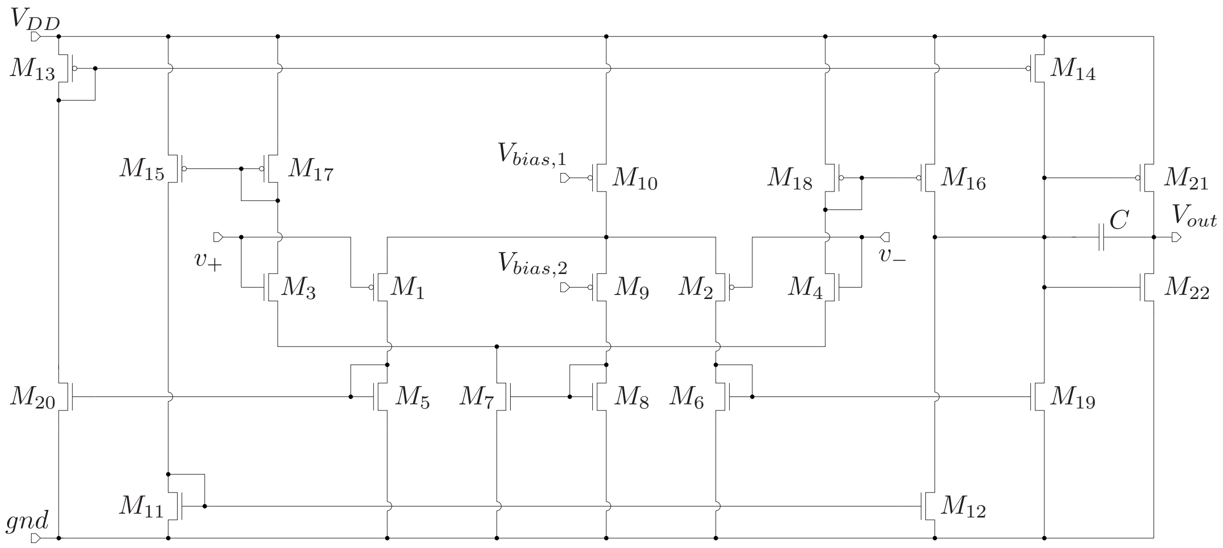


Figure 6.2: Low voltage amplifier [Mar98]

Table 6.5: Design parameters for the sizing of the low voltage amplifier

| Device | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_9$ | $M_{10}$ | $M_{13}$ | $M_{14}$ | $M_{15}$ | $M_{16}$ | $M_{17}$ | $M_{18}$ | $M_{21}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Lengths | $l_0$ | | | | | | | | | | | | |
| Widths | $w_1$ | | $w_3$ | | $w_9$ | $w_{10}$ | $w_{13}$ | | $w_{15}$ | | | | $w_{21}$ |
| Multipliers | $m_1$ | | $m_3$ | | $m_9$ | $m_{10}$ | $m_{13}$ | | $m_{15}$ | | $m_{17}$ | | $m_{21}$ |

| Device | $M_5$ | $M_6$ | $M_{19}$ | $M_{20}$ | $M_7$ | $M_8$ | $M_{11}$ | $M_{12}$ | $M_{22}$ | $C$ | $V_{bias,1}$ | $V_{bias,2}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Lengths | $l_0$ (same as in first part of table) | | | | | | | | | — | | |
| Widths | $w_5$ | | | | $w_7$ | | $w_{11}$ | | $w_{22}$ | — | | |
| Multipliers | $m_5$ | | $m_{19}$ | | $m_7$ | $m_8$ | $m_{11}$ | | $m_{22}$ | — | | |
| Capacitances | — | | | | | | | | | $C_c$ | — | |
| Voltages | — | | | | | | | | | | $v_1$ | $v_2$ |

Table 6.6: Operating parameters for the sizing of the low voltage amplifier

| Parameter | Minimum | Nominal | Maximum |
|---|---|---|---|
| Temperature $T$ in $°C$ | -40 | 27 | 125 |
| Supply Voltage $V_{DD}$ in $V$ | 1.9 | 2.0 | 2.1 |
| Load Capacitance $C_L$ in $pF$ | 19 | 20 | 21 |

Table 6.7: Performance specifications for low voltage amplifier; $f_T$ is the transit frequency, $\varphi_R$ is the phase margin, $SR+$ and $SR-$ are slew rate rising and falling

| Performance | Specification |
|---|---|
| Power $[mW]$ | $< 7$ |
| PSRR $[dB]$ | $> 90$ |
| Gain $[dB]$ | $> 75$ |
| CMRR $[dB]$ | $> 90$ |
| $f_T$ $[MHz]$ | $> 20$ |
| $\varphi_R$ $[°]$ | $> 60$ |
| $\lvert$SR$-\rvert$ $[\frac{V}{\mu s}]$ | $> 12$ |
| SR+ $[\frac{V}{\mu s}]$ | $> 12$ |

The amplifier contains a PMOS and an NMOS differential pair built by the transistors $M_1$-$M_2$ and $M_3$-$M_4$, respectively, so that only one multiplier and one transistor width was considered for each differential pair. As discussed for the Miller amplifier, the widths of the transistors in a current mirror were set equal to allow a layout of the transistors, e.g., as multi-finger structures, without using additional rounding operations. In addition, due to the symmetry of the amplifier, some of the transistors were sized equally. Considering this symmetry, the widths of the transistors $M_5$-$M_6$-$M_{19}$-$M_{20}$, $M_{15}$-$M_{16}$-$M_{17}$-$M_{18}$, $M_7$-$M_8$, $M_{11}$-$M_{12}$, and $M_{13}$-$M_{14}$ and the multipliers of the transistors $M_5$-$M_6$, $M_{19}$-$M_{20}$, $M_{15}$-$M_{16}$, $M_{17}$-$M_{18}$, $M_{11}$-$M_{12}$, and $M_{13}$-$M_{14}$ were matched. Thus, 11 widths and 14 multipliers were used as discrete design parameters for the sizing task. In addition, the bias voltages $V_{bias,1}$ and $V_{bias,2}$, and the capacitance $C$ were considered as continuous parameters such that the number of parameters used for the sizing of the low voltage amplifier summed up to the 3 continuous and 26 discrete design parameters listed in Table 6.5.

For the tolerance design of the low voltage amplifier, the operating temperature $T$, the supply voltage $V_{DD}$, and the load capacitance $C_L$ were considered as operating parameters. The specified values of nominal conditions and lower and upper bounds are shown in Table 6.6. In addition, five normally distributed global variations were considered as process variations: the variation of the oxide thickness, of the electron mobility for NMOS and PMOS transistors, and of the threshold voltage for NMOS and PMOS transistors.

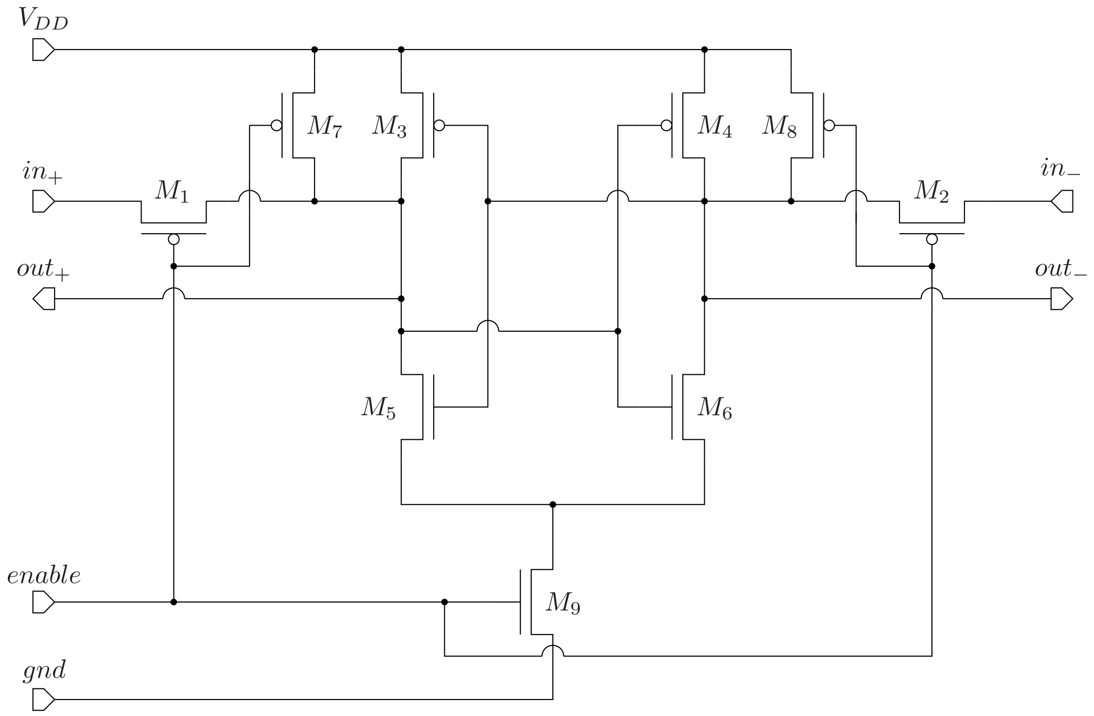


Figure 6.3: Sense amplifier [YM06]

Table 6.8: Design parameters for the sizing of the sense amplifier

| Device | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_7$ | $M_8$ | $M_5$ | $M_6$ | $M_9$ |
|---|---|---|---|---|---|---|---|---|---|
| Lengths | $l_p$ | | | | | | $l_n$ | | |
| Widths | $w_p$ | | | | | | $w_n$ | | |
| Multipliers | $m_1$ | | $m_3$ | | $m_7$ | | $m_5$ | | $m_9$ |

Besides the constraints used above to match the design parameters, 93 design constraints were computed by the approach in [MGS08]. Again eight performances were specified for the sizing of the amplifier. These performance specifications are outlined in Table 6.7.

### 6.1.4 Sense Amplifier

The sense amplifier in Figure 6.3 represents an example from the field of mixed-signal circuits and can be used, e.g., to read values from an SRAM cell. The amplifier contains two cross coupled inverters ($M_3$-$M_5$, $M_4$-$M_6$) which force the signal at the positive output to be the inverse of the negative output and which cause two stable states and one unstable state of the circuit. The transistors $M_1$ and $M_2$ disconnect the bit lines at $in_+$ and $in_-$ from the output nodes if the enable signal is set to $V_{DD}$ – i.e., if a signal at the input should be amplified – and the output signal depends on the discharge of the corresponding nodes due to the leakage current over $M_1$ and $M_2$. Transistor $M_9$ disconnects the circuit from ground if the enable signal is on ground level, i.e., if no

Table 6.9: Operating parameters for the sizing of the sense amplifier

| Parameter | Minimum | Nominal | Maximum |
|---|---|---|---|
| Temperature $T$ in $°C$ | -40 | 27 | 85 |
| Supply Voltage $V_{DD}$ in $V$ | 2.4 | 2.5 | 2.6 |
| Bit-Line Capacitance $C_L$ in $fF$ | 90 | 100 | 110 |
| Load Capacitance $C_L$ in $fF$ | 14 | 15 | 16 |

Table 6.10: Performance specifications for the sense amplifier; $SR+$ and $SR-$ are slew rate rising and falling, $t_s$ is the settling time

| Performance | Specification |
|---|---|
| DC-Power $[nW]$ | $< 1$ |
| $t_s[ps]$ | $< 500$ |
| Area $[\mu^2 m^2]$ | $< 10$ |
| $\vert$SR$-\vert$ $[\frac{V}{ns}]$ | $> 7$ |
| SR+ $[\frac{V}{ns}]$ | $> 7$ |

input signal should be amplified. At the same time, the transistors $M_7$ and $M_8$ connect the output and – as $M_1$ and $M_2$ are open if the enable signal is at ground level – the input to the supply voltage and the bit-lines at the input are preloaded.

To allow a layout which is insensitive to local variations, e.g., using a common centroid structure, without using additional rounding operations, all lengths and widths of PMOS and all lengths and widths of NMOS transistors were set equal. In addition, the symmetry of the circuit was considered and the multipliers of the transistors $M_1$-$M_2$, $M_3$-$M_4$, $M_5$-$M_6$, and $M_7$-$M_8$ were set equal. Thus, the sizing task had the nine discrete design parameters listed in Table 6.8

Besides variations in the supply voltage $V_{DD}$ and in the temperature, the capacitance values of the input capacitance and of the output capacitance were considered as operating parameters for the tolerance design (cf. Table 6.9). The capacitance value at the input, which is typically formed by the parasitic capacitors of the bit-lines and – if the sense amplifier is used in an SRAM design – of the SRAM cells connected to these bit-lines, was assumed to be significantly higher than the capacitive load at the output, which is typically dominated by the parasitic capacitance of one single subsequent gate. For the variation of the process parameters, normally distributed global variations of the oxide thickness, of the electron mobility for NMOS and PMOS transistors, and of the threshold voltage for NMOS and PMOS transistors were assumed such that five statistical parameters are considered in this example.

For the sense amplifier, the five performances in Table 6.10 were specified. Using the parameters in Table 6.8, the area was approximated by:

$$\text{area} \approx (m_1 + m_3 + m_7) \cdot w_p \cdot l_p + (m_5 + m_9) \cdot w_n \cdot l_n \tag{6.1}$$

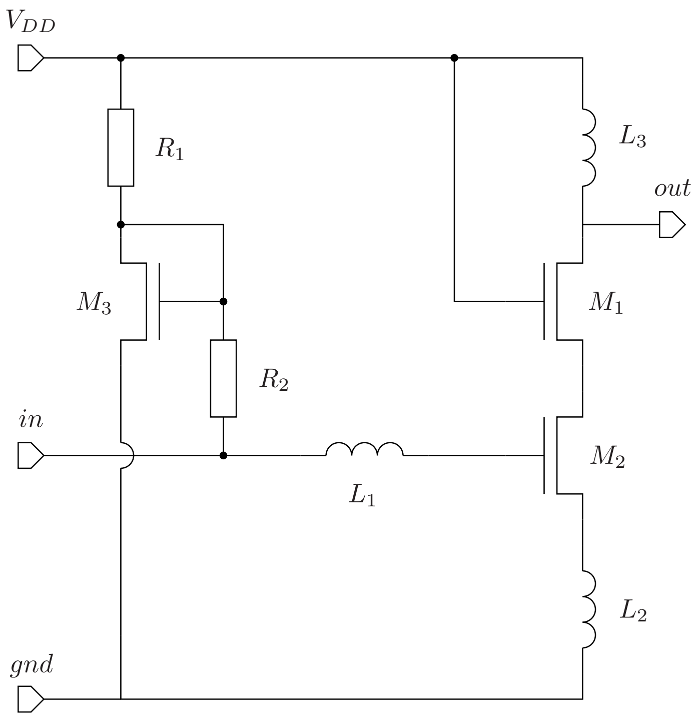


Figure 6.4: Low noise amplifier [Lee04]

The DC-Power was measured for the enabled signal set to $V_{DD}$, i.e., assuming that transistor $M_9$ connects the circuit to ground. This power consumption can be assumed to be always higher than the power consumption for enable set to ground level.

For the sense amplifier, no DC-constraints were required. However, six transient constraints were used which ensured that the output signal was in the expected range. These constraints prevented the output signals from switching into the wrong direction and ensured that the output signals reached the required voltage levels in time.

### 6.1.5 Low Noise Amplifier

The last amplifier investigated is the low noise amplifier (LNA) in Figure 6.4. In contrast to the previous circuits, no symmetry can be considered to reduce the number of parameters. Hence, only the lengths of all transistors were matched and the number of parameters for the sizing task shown in Table 6.11 summed up to 2 continuous and 22 discrete parameters if the sides of the inductors were considered. However, the influence of the sides can only be considered by the approach in Chapter 5 because this influence is modeled based on a look-up table (cf. [PFC09]) and no intermediate points can be simulated. For comparability, the number of sides was fixed to four and the number of parameters considered in the experiments was reduced to 21 if the consideration of the number of sides is not mentioned explicitly.

Table 6.11: Design parameters for the sizing of the low noise amplifier

| Device | $M_1$ | $M_2$ | $M_3$ |
|---|---|---|---|
| Lengths | $l_0$ | | |
| Widths | $w_1$ | $w_2$ | $w_3$ |
| Multipliers | $m_1$ | $m_2$ | $m_3$ |

| Device | $R_1$ | $R_2$ |
|---|---|---|
| Resistances | $r_1$ | $r_2$ |

| Device | $L_1$ | $L_2$ | $L_3$ |
|---|---|---|---|
| Turns | $n_1$ | $n_2$ | $n_3$ |
| Line Width | $w_{L1}$ | $w_{L2}$ | $w_{L3}$ |
| Line Spacing | $s_1$ | $s_2$ | $s_3$ |
| Inner Diameter | $d_1$ | $d_2$ | $d_3$ |
| Sides | $N_1$ | $N_2$ | $N_3$ |

Table 6.12: Operating parameters for the sizing of the LNA

| Parameter | Minimum | Nominal | Maximum |
|---|---|---|---|
| Temperature $T$ in $°C$ | -40 | 27 | 85 |
| Supply Voltage $V_{DD}$ in $V$ | 2.4 | 2.5 | 2.6 |
| Load Capacitance $C_L$ in $fF$ | 450 | 500 | 550 |

For the tolerance design of the circuit, the values of supply voltage, temperature, and load capacitance were considered as operating parameters as shown in Table 6.12. As in the previous examples, normally distributed global variations of oxide thickness, electron mobility, and threshold voltage were considered for the NMOS transistors. In addition, normally distributed local variations of electron mobility and threshold voltage were assumed for each transistor such that nine process parameters were considered in this example.

To provide for basic functionality, nine constraints were formulated to ensure that all transistors operate in the saturation region. Moreover, six performances were specified as shown in Table 6.13. The specifications were defined for a center frequency of $2.4GHz$ and describe the circuit properties using the s-parameters. In addition, the noise factor and the third-order input intercept point (IIP3) were considered.

Table 6.13: Performance specifications for the low noise amplifier for a center frequency of $2.4GHz$

| Performance | Specification |
|---|---|
| $Gain(S21)[dB]$ | $> 15$ |
| $S11[dB]$ | $< -9$ |
| $S22[dB]$ | $< -5$ |
| $S12[dB]$ | $< -30$ |
| $NF[dB]$ | $< 2.5$ |
| $IIP3[dBm]$ | $> -5$ |

## 6.2 Nominal Design

To compute a nominal design, the sizing task is formulated as an optimization problem with fixed operating and process parameters as described in Chapter 2.5.3. The objective of the nominal design is to compute a discrete sizing which fulfills all specifications and constraints. For this purpose, a feasible point was computed for each of the circuits in Section 6.1, and the approach in Chapter 4 (referred to as **B**ranch-**A**nd-**B**ound for **A**nalog sizing or **BABA**) as well as the approach in Chapter 5 (referred to as **Mo**del-based **P**rogramming for analog **S**izing or **MoPS**) were applied. In addition, continuous sizing with subsequent rounding was applied as a commonly used state-of-the-art method and was compared to the new methods. To show the benefit of using the prediction method in Chapter 4.3 in the Branch-and-Bound approach for analog sizing, a comparison with the outer Branch-and-Bound approach in Chapter 4.2 is shown. For this comparison, the outer Branch-and-Bound is enlarged so that the continuous result of the simulation-based FSQP approach in each recursion is rounded to the next discrete point and so that this rounded point is considered as a solution candidate for the sizing task.

A fair comparison to commonly used statistical approaches is not possible – and is therefore not discussed in detail in this thesis – because these approaches do typically not consider continuous parameters. However, if all continuous parameters are discretized, comparisons for the Miller amplifier and for a BiCMOS operational transconductance amplifier have shown that the runtime of statistical approaches is higher than the runtime of the BABA approach by a factor 8 to 36 (cf. [PZG10]). Further experiments with the Miller amplifier and with the low voltage amplifier have shown that the commercially used statistical approaches applied did not find an existing solution for the sizing task in some cases. However, a solution for the same tasks can be found by the new approaches in this thesis. The statistical approaches failed to find such a solution in the experiments, especially when the number of discrete points in the design space was large and when the number of possible solutions was small. The first case appears, e.g., if the grid size for some parameters is small. The second case can be observed, e.g., if the specifications for the circuit are restrictive.

### 6.2.1 Nominal Design Using Discretely Simulated Parameters

Most of the parameters used to compute a sizing of the circuits in Section 6.1 can be scaled continuously although the final result should be in a discrete domain. However, the influence of the number of sides of an integrated inductor on the inductance value is computed by using correction factors which can be read from a look-up table (cf. [PFC09]). As a consequence, the inductance value is not defined and the low noise amplifier in Section 6.1.5 can not be evaluated for intermediate values for the number of sides.

For the experiment in this subsection, the upper bound for the inner diameter and for the number of winding turns of the inductors are restricted to $d_i \leq 40\mu m$ and $n_i \leq 5$ with $i = 1, 2, 3$, respectively, and the line spacings and line widths are set to the fixed values $s_i = 1.5\mu m$ and $w_{L,i} = 5\mu m$[1]. The most important influence of this modification to the sizing task is that the maximum available inductance is reduced.

A solution for the sizing task with the specifications as given in Section 6.1.5 but with the modified parameter ranges should be computed by the four approaches mentioned above. However, a sizing in the continuous domain can only be computed if the circuit can be evaluated for each point in the relaxed design space (2.14). As a consequence, the three parameters which describe the numbers of sides of the integrated inductors in the low noise amplifier cannot be considered for continuous sizing and subsequent rounding, for the BABA approach, and for the outer Branch-and-Bound approach. Only the MoPS approach can consider these three additional degrees of freedom for the sizing task. For the other approaches the number of sides of each inductor is set to eight.

Applying the FSQP approach to this sizing problem, the best result which can be computed in the continuous domain gains performance values for the s-parameters $S11$ and $S21$ of $S11 = -8.7dB$ and $S21 = 14.3dB$, respectively. I.e., the specifications cannot be fulfilled by the continuous sizing approach. However, the FSQP approach is used for continuous sizing and rounding as well as for the Branch-and-Bound approaches in Chapter 4. As a consequence, no discrete solution for the sizing task – which can exist only if a continuous solution exists – was computed by rounding, BABA, and the outer Branch-and-Bound approach.

In contrast, the new MoPS approach did find a solution for the sizing task if the additional three parameters for numbers of sides of the inductors were considered. Figure 6.5 shows that the runtime to find such a solution was 32 minutes and thus only 10 minutes higher than the runtime required by the continuous approach which converged after 20 minutes at a point which did not solve the problem. The runtime consists of a part required for the simulations and of a part required for other numerical operations (referred to as numerical runtime). It can be seen that the numerical runtime of the MoPS approach contributes significantly to the overall runtime. This phenomenon is discussed in detail in Section 6.2.2

It can be observed that the number of sides for the inductors $L_1$ and $L_3$ is changed to four, i.e., to a squared design, if the MoPS approach is used. The number of sides for inductor $L_2$ is kept at eight, i.e., a octagonal design is selected. This result corresponds to the expectation as – if all other parameters are equal – the inductance can take the highest value for the squared design, and as the inductance of $L_1$ and $L_3$ is typically larger than the inductance of $L_2$ in the considered design of a low noise amplifier.

With the results in this section and considering the discussion of the integrated inductor in [PFC09], the numbers of sides were set to four for the experiments in Sections 6.2.2 and 6.3. Using this setting and the parameter ranges as defined in Section 6.1.5, a discrete solution for the sizing task exists.

[1]Such a setup could be used, e.g., to limit the area required by the inductors.

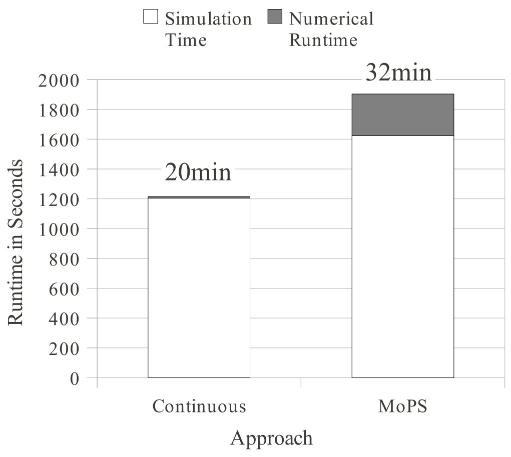


Figure 6.5: Runtime of MoPS (Chapter 5) for the sizing of the low noise amplifier considering the inductor sides. The runtime of the continuous approach is given as a reference although it did not find a valid solution for the task.

## 6.2.2 Nominal Design Using Continuously Simulated Parameters

The previous section has shown that from the four algorithms presented only the MoPS approach can be used to compute the sizing of a circuit reliably if the circuit cannot be evaluated for each continuous point in the relaxed design space (2.14). However, in many practical cases an evaluation of each continuous point is possible. Such cases are discussed in this section. For the comparability of the approaches in case of the low noise amplifier, the numbers of sides were set to a fixed value and only the remaining 21 design parameters – which can be relaxed and evaluated for continuous intermediate points – were considered for the experiments below.

### Solution Quality

The first approach which is used to find a solution for the sizing tasks in Chapter 6.1 is the computation of a sizing in the continuous domain and rounding the result to the next point. This state-of-the-art approach is frequently used in practice due to the high efficiency of continuous approaches if the sizing is to be computed in a continuous domain. However, the results in Table 6.14 show that the discrete point computed by this method violates one or more specifications in three of the four experiments. In the fourth experiment – the sizing of the low voltage amplifier – two sizing constraints were violated. I.e., no discrete point which was computed by continuous optimization and subsequent rounding is a valid solution for the corresponding sizing task.

However, such a valid discrete solution for the sizing task exists. This can be seen in Table 6.14 if the results of the BABA, the outer Branch-and-Bound, or the MoPS

Table 6.14: Number of violated specifications and constraints after the computation of a nominal sizing with continuous sizing and rounding (C+R), BABA, outer Branch-and-Bound (outer BAB), and MoPS

| Circuit | Specifications violated/overall | | | | Constraints violated/overall | | | |
|---|---|---|---|---|---|---|---|---|
| | C+R | outer BAB | BABA | MoPS | C+R | outer BAB | BABA | MoPS |
| Miller Amplifier | $\boxed{\mathbf{3/8}}$ | 0/8 | 0/8 | 0/8 | 0/57 | 0/57 | 0/57 | 0/57 |
| Low Voltage Amplifier | 0/8 | 0/8 | 0/8 | 0/8 | $\boxed{\mathbf{2/93}}$ | 0/93 | 0/93 | 0/93 |
| Sense Amplifier | $\boxed{\mathbf{1/5}}$ | 0/5 | 0/5 | 0/5 | 0/8 | 0/8 | 0/8 | 0/8 |
| Low Noise Amplifier | $\boxed{\mathbf{1/6}}$ | 0/6 | 0/6 | 0/6 | 0/9 | 0/9 | 0/9 | 0/9 |

approach are considered. Each of these new approaches was able to find a solution for the sizing task which fulfills all specifications and constraints[2].

The outer Branch-and-Bound approach did not find a solution for the discrete sizing task in the first recursion because the procedure in this recursion is identical to continuous sizing and subsequent rounding. However, using additional rounding constraints, a solution was found after a low number of recursions. In the search process, considering the rounded solution did reduce the number of considered subdomains. Nevertheless, in each example more than one simulation-based FSQP run was required to find a solution, which leads to a significantly increased runtime of the approach compared to continuous sizing and subsequent rounding as discussed below.

The BABA approach enlarges the outer Branch-and-Bound approach by the prediction method for a discrete solution described in Chapter 4.3. As a consequence, it can be guaranteed that it finds a solution for the sizing task if the outer Branch-and-Bound approach converges against such a solution. However, in each of the experiments the first prediction found a discrete solution for the sizing task. Therefore, only one FSQP run was required and the discrete solution was computed on the linearly constrained quadratic model.

The MoPS approach also converged in each example to a solution which solves the sizing task. In contrast to outer Branch-and-Bound and BABA, it does not require the computation of discrete intermediate points to find such a solution. Therefore, it is the

[2]A comparison of the performance values is not provided because each of the algorithms is stopped as soon as a solution has been found. As a consequence the performance values achieved at a point which fulfills all specifications and constraints are not significant.

only approach which can still be used and which will still solve the sizing problem if some of the parameters can only be evaluated at discrete points (cf. Section 6.2.1).

From the fact that a discrete solution exists for each sizing task and that this solution cannot be found by continuous optimization and subsequent rounding, it can be concluded that this state-of-the-art approach should not be used for discrete analog sizing. In contrast, in all experiments the new approaches converged reliably against a discrete solution for the sizing task.

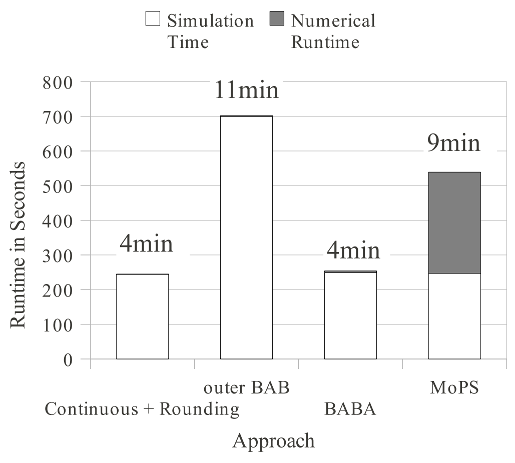
## Runtime Comparison

A discrete solution for the sizing task must be found reliably. However, the second important criterion for the quality of an algorithm for analog sizing is the required runtime. To evaluate this criterion for the new algorithms, the time required to solve the four sizing problems mentioned above is considered. To make the comparison of the algorithms as fair as possible, all algorithms are started from the same initial point. Although the continuous approach with subsequent rounding did not find a solution for any of the nominal sizing tasks defined in Section 6.1, the required computation time for this state-of-the-art approach is given as a reference. The runtime to compute a nominal design for all four sizing tasks is presented in Figure 6.6. The Figure shows the overall runtime of each approach and the amount of time which was required for simulations – i.e., to evaluate the circuit – as well as the amount of time required for other numerical operations (referred to as numerical runtime in the following). It is worth noting that the time required for the simulations can be greatly reduced if more CPUs are used and if the algorithm can therefore be parallelized to a higher degree.

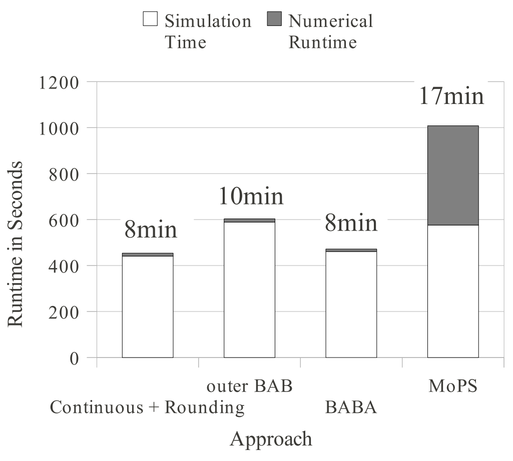
At first, the runtime for the nominal sizing using the BABA approach should be discussed. Figure 6.6 shows that the additional runtime required by this new approach is – compared to the state-of-the-art approach of continuous sizing and subsequent rounding – always below 1 minute and can be neglected in comparison to the complete runtime of the algorithm. In all cases, the solution was computed on the quadratic model in Chapter 4.3 and the simulation-based FSQP run was executed only once.

The results also show that using the quadratic model the runtime is considerably lower than the runtime required by the outer Branch-and-Bound approach in Chapter 4.2, which is stopped as soon as the rounded solution of a subproblem fulfills the specifications. The reason for the longer runtime is that typically more than one subproblem must be solved by the simulation-based FSQP approach if the discrete solution is not computed on the quadratic model.

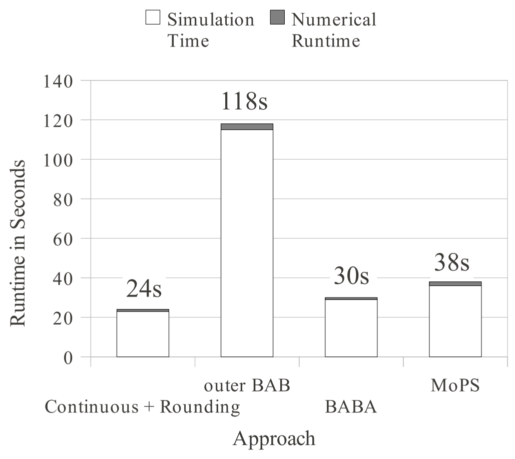
In contrast to the Branch-and-Bound approach, the runtime required to solve the sizing task using the MoPS approach from Chapter 5 is always – in the results approximately by a factor two – longer than the runtime of the BABA approach. However, for the MoPS approach no simulations between the discrete grid points are required such that the additional runtime is acceptable if the circuit can only be evaluated at discrete points, i.e., if the BABA approach cannot be used.



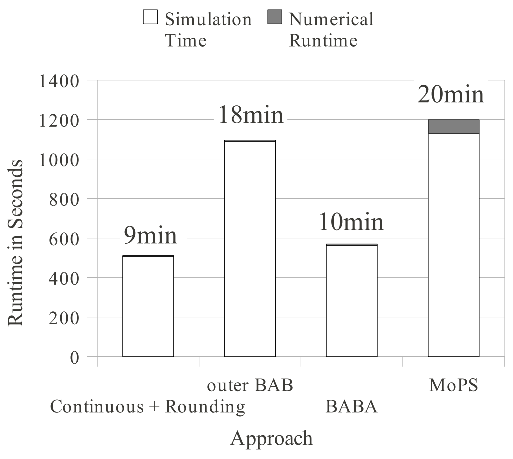
(a) Miller Amplifier



(b) Low Voltage Amplifier



(c) Sense Amplifier



(d) Low Noise Amplifier (without consideration of inductor sides)

Figure 6.6: Runtime of the state-of-the-art approach (Continuous + Rounding), the approach in Chapter 4 (BABA), outer Branch-and-Bound (outer BAB), and the approach in Chapter 5 (MoPS) for a nominal design of the circuits in Section 6.1

In the results two reasons can be observed which cause a longer runtime to compute a nominal design by using the new MoPS approach: The first reason which can be clearly identified in Figures 6.6a and 6.6b is due to an increased runtime of the algorithm excluding the simulation-based circuit evaluations. The increased runtime in these cases is due to the high number of nonlinear models of the underlying optimization problem which must be solved over the discrete domain (cf. Chapter 5.3). E.g., for the Miller amplifier, the MoPS approach requires the computation of eight sensitivities – one fewer than the FSQP approach – for each performance and for each constraint, which make up the greatest part of the simulation effort. However, during this run 24 non-linear models of the problem were solved where each of these numerical problems had a high number of performances and constraints which made the numerical problem complex. As a consequence, the effort to solve these subproblems is typically the reason for the increased runtime if the non-linear model of the problem is hard to solve. Besides the high number of performances and constraints, one of the major reasons which can be observed for an increased numerical runtime is the correlation of design parameters: E.g., if the multiplier and the width of one transistor are used as design parameters for the sizing task, the effect of scaling multiplier and width is not equal but similar. As a consequence, if one of the parameters is forced by branching to be discrete, the other – possibly currently discrete – parameter has a high chance of becoming non-discrete, and the number of subproblems to solve the discrete optimization problem increases.

The second reason for the increased runtime in the examples is an increased simulation effort and can be observed in the result in Figure 6.6d. Also in this case, the numerical effort of the MoPS approach cannot be neglected. However, the main reason for the increased runtime is the higher number of sensitivities which must be computed to model the performances (10 for each performance and each constraint in the MoPS approach compared to 6 for the BABA approach). This reason for an increased runtime can be observed if the effort to compute a good quadratic model for each performance is high, i.e., if the considered performances and constraints are stronger non-linear.

It can be concluded that both new approaches can be used to solve the nominal sizing task, but – due to the lower runtime – the BABA approach is preferable if a simulation of the circuit at continuous intermediate points is possible.

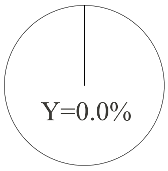
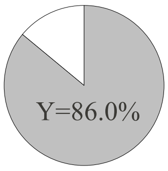
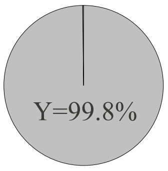
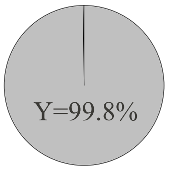
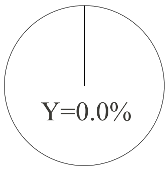
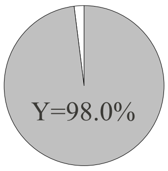
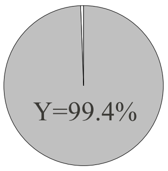
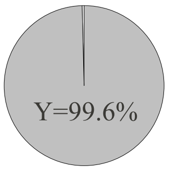
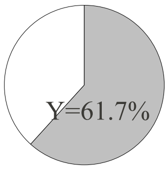
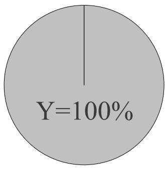
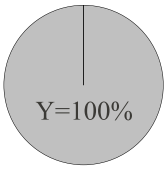
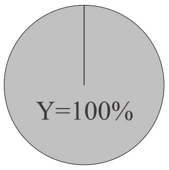
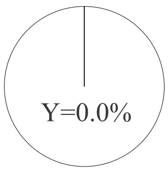
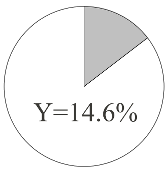
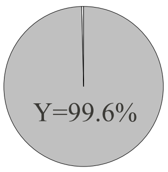
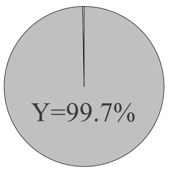
## 6.3 Tolerance Design

For the computation of a tolerance design, the mathematical formulation in Chapter 2.5.4 was used. The problem was solved by the new approaches in Chapters 4 and 5 and is compared to continuous sizing with subsequent rounding, which can be considered as a state-of-the-art approach used to compute a tolerance design along with the consideration of discrete design parameters. For each sizing task, the algorithms were started with the same sizing at a design parameter point where the specified performances were fulfilled for nominal operating and process conditions. The outer Branch-and-Bound approach without prediction of the discrete solution is not considered in this section due to the observation in Section 6.2 that it is always slower than the Branch-and-Bound approach for analog sizing using the prediction.

### Solution Quality

As mentioned above, the required yield for the problem is set to 99.87% per performance ($3\sigma$ or $\beta_W = 3$ in (2.59)), i.e., the analog sizing task is solved for the Miller amplifier and for the low voltage amplifier if the overall yield measured at the worst case operating points is higher than $99.87^8 \approx 99.0\%$, where 8 is the number of performances.

Table 6.15: Yield Y after nominal design, state-of-the-art tolerance design (Continuous + Rounding), and tolerance design with the approaches in Chapter 4 (BABA) and Chapter 5 (MoPS) for the circuits in Section 6.1

| Circuit | Nominal | Continuous + Rounding | BABA | MoPS |
|---|---|---|---|---|
| Miller Amplifier |  Y=0.0% |  Y=86.0% |  Y=99.8% |  Y=99.8% |
| Low Voltage Amplifier |  Y=0.0% |  Y=98.0% |  Y=99.4% |  Y=99.6% |
| Sense Amplifier |  Y=61.7% |  Y=100% |  Y=100% |  Y=100% |
| Low Noise Amplifier |  Y=0.0% |  Y=14.6% |  Y=99.6% |  Y=99.7% |

Analogously, the tolerance sizing task is solved for the sense amplifier and for the low noise amplifier if a yield of at least[3] $99.87^4 \approx 99.5\%$ and $99.87^6 \approx 99.2\%$, respectively, is reached. Higher yield than the minimum yield defined in this example can be achieved if the value of $\beta_W$ in (2.59) is increased.

The results of the sizing approach listed in Table 6.15 were evaluated by a Monte Carlo analysis with 2500 samples executed at the worst case operating points and using the tool WiCkeD [WIC11]. Hence, if a yield of 100% was measured and is given in the table, the yield is – with a probability of 95% – higher than 99.85%.

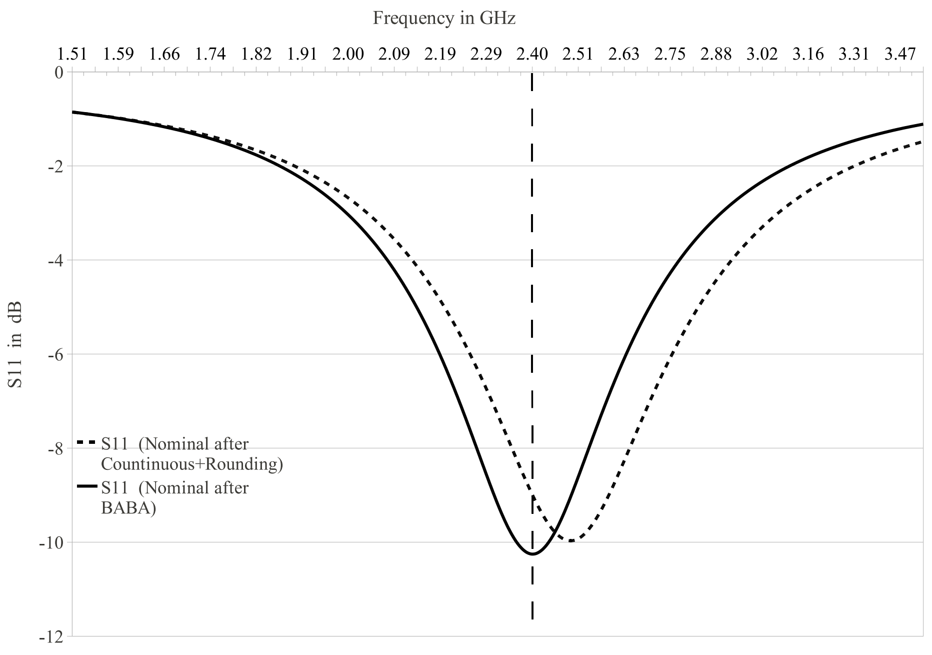
Starting from an initial solution which does not fulfill the yield requirements (shown in the left column of Table 6.15), the results clearly show that both new sizing approaches

[3]None of the considered process parameters influences the area of the circuit. Thus, the worst case point for the area is equal to the nominal point for the area and only four performances can cause a loss of yield for the sense amplifier.

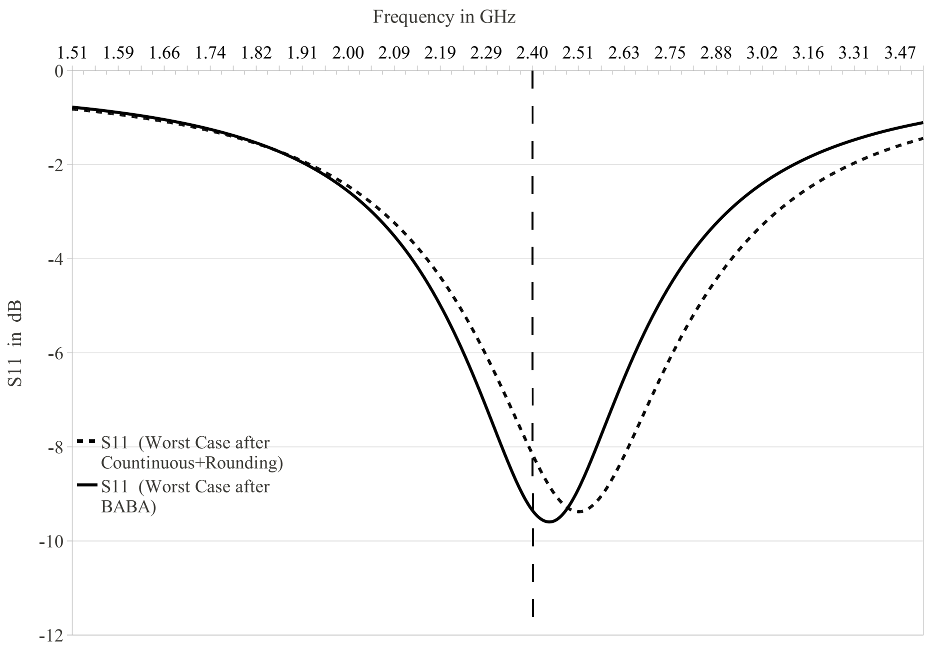
(listed in the two columns on the right) can find a solution where the specified yield can be reached. At the same time it can be seen that continuous sizing with subsequent rounding does not solve the task in three of the four cases. Especially, the sizing of the Miller amplifier and of the low noise amplifier shows that a more sophisticated approach than the state-of-the-art approach is required to compute a tolerance design for analog circuits.

The reason for the much higher yield in the design computed by BABA and MoPS compared to the low yield of the sizing computed by continuous optimization and subsequent rounding is further analyzed for the low noise amplifier. For this purpose, the s-parameter $S11$ – which is predominantly responsible for the low yield of the low noise amplifier if it is sized by the rounding approach – is shown in Figure 6.7. This s-parameter can be considered as a measure for the input matching of the amplifier where a good matching corresponds to a low value of $S11$. However, due to the rounding, e.g., the number of winding turns is changed from an arbitrary float number to a discrete value. As a consequence, the matching gets worse and the minimum for $S11$ is no longer at the center frequency of $2.4GHz$ but is shifted to higher frequencies. In addition, the value of the minimum of $S11$ is increased. These effects can be observed for the nominal value of operating and process parameters (Figure 6.7a) and are increased considering the worst case conditions (Figure 6.7b). In contrast, $S11$ has its minimum exactly at the center frequency if nominal conditions are considered for the solution computed by the new algorithms (in the figure the result of the BABA approach is shown). Furthermore, the minimum value is lower than after continuous optimization with subsequent rounding. As a consequence, the minimum for the worst case points is also closer to the center frequency and the specification is fulfilled. It can be noted, that cases can be observed also where, after continuous sizing with subsequent rounding, the specifications are violated at the nominal point although the specifications were fulfilled at the worst case points after the sizing in the continuous domain. Thus, it can be concluded that considering discrete design parameters is crucial if the matching of the circuit is important.

The only circuit in the examples where the yield requirements are fulfilled after continuous optimization and subsequent rounding was the sense amplifier from Section 6.1.4. The reason for the low influence of the rounding operation can be found in the problem size and in the structure of the problem: As stated in Chapter 4.3, the rounding operation often yields good results if the number of parameters which should be discretized is low. However, for the sense amplifier the number of discrete parameters is only nine. In addition, the sizing of a sense amplifier is typically good if all lengths, widths, and multipliers are as low as possible. Thus, after the continuous sizing three parameters are already close to the lower bound and can be considered as discrete such that only 6 parameters must be discretized by the rounding operation. However, in this case as well it cannot be guaranteed in general that this rounded solution solves the sizing task such that the fact that the rounded solution fulfills the task can be considered as random behavior.

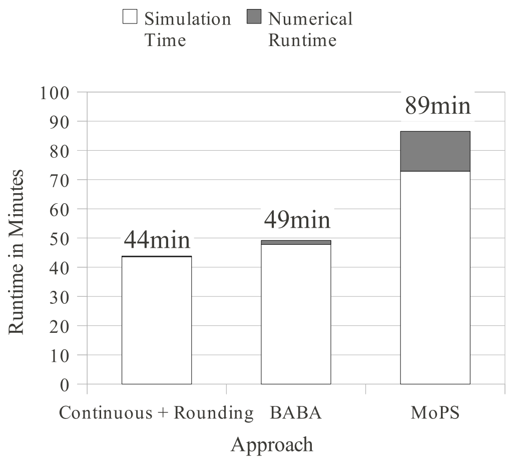


(a) S-parameter $S11$ at nominal operating and process parameter values

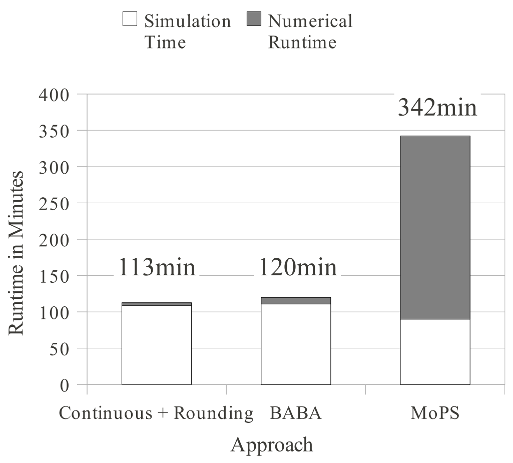


(b) S-parameter $S11$ at worst case operating and process parameter values

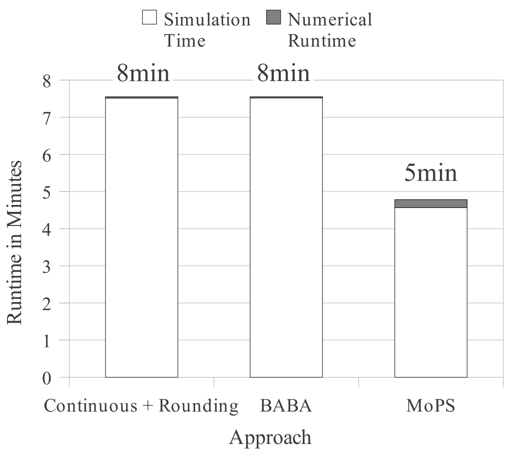
Figure 6.7: S-parameter S11 of the low noise amplifier plotted over frequency for sizing computed by BABA and by continuous optimization and rounding



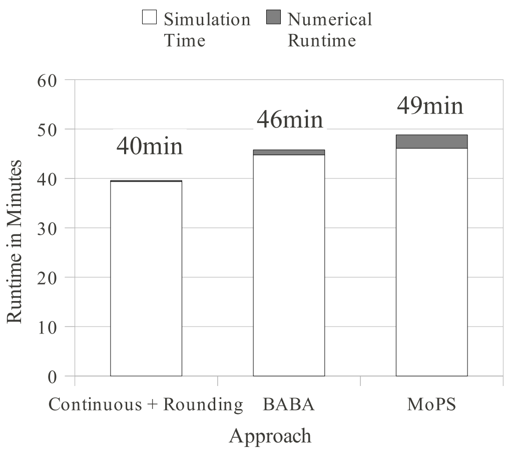
(a) Miller Amplifier



(b) Low Voltage Amplifier



(c) Sense Amplifier



(d) Low Noise Amplifier

Figure 6.8: Runtime comparison of the state-of-the-art approach (Continuous + Rounding) with the approach in Chapter 4 (BABA), and the approach in Chapter 5 (MoPS) for a tolerance design of the circuits in Section 6.1

### Runtime Comparison

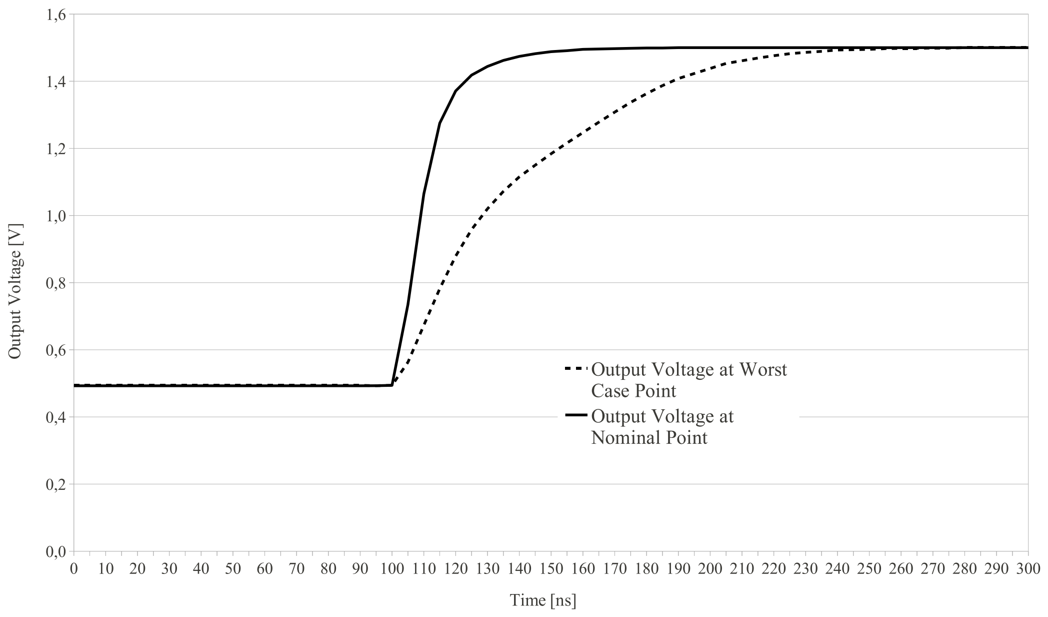
Both new approaches can compute a tolerance design in the analog sizing step. Thus, the efficiency of the new approaches with respect to the computational time can be evaluated. As in the nominal case, continuous sizing and subsequent rounding is given as a reference, although it has been shown that the state-of-the-art approach cannot find a solution for the tolerance design task in general.

The runtime to compute a tolerance design for the different circuits is given in Figure 6.8. The results show that the additional runtime of the BABA approach compared to the state-of-the-art approach of continuous optimization and subsequent rounding is, with seven minutes at most, still small and can be neglected in all examples, especially if the benefit of the increased yield is taken into account. Comparing the runtime to compute a tolerance design with the runtime to compute a nominal design, predominantly the time required for simulating the circuit is increased for the Branch-and-Bound approach for analog sizing. The reason for this behavior is that for the tolerance design the sensitivities of the performances are not only computed against variations of the design parameters but also against variations of statistical and operating parameters.

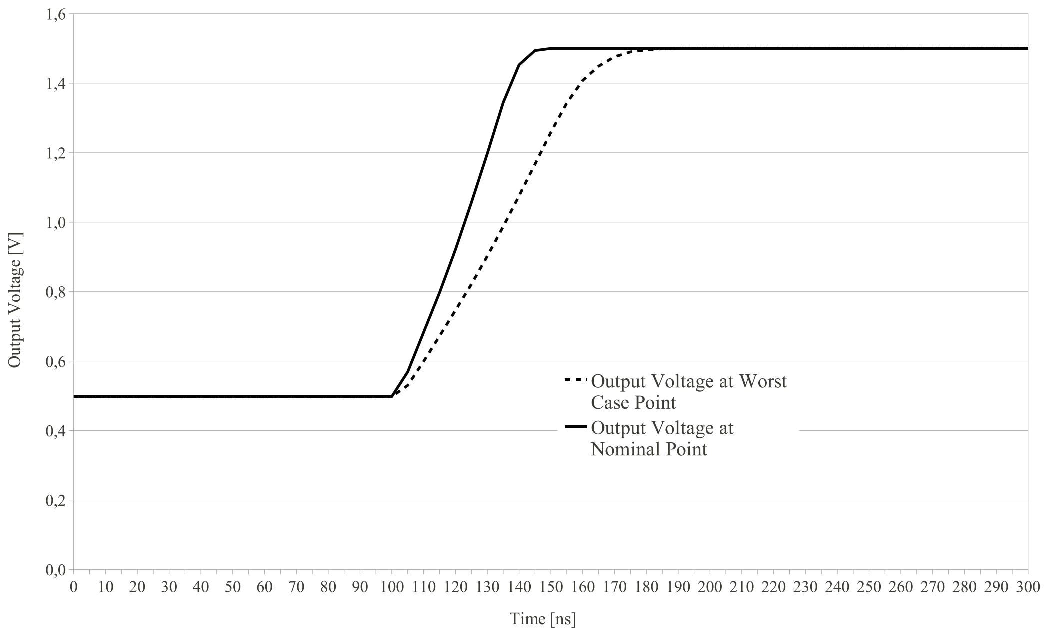
The runtime comparison of BABA and MoPS shows that – although the runtime of the MoPS approach can be lower than the runtime of the BABA approach for small size problems – the runtime of MoPS is up to a factor three higher than the runtime of the BABA approach. The increase is typically due to a large number of nonlinear models of the discrete optimization problem which must be solved. This can be observed especially for the low voltage amplifier where the effort for simulation-based evaluations is only a forth of the overall runtime. The high runtime without circuit evaluations can be explained by the fact that on the one hand, the time to solve the nonlinear model of the sizing task in the continuous domain is high (several seconds) if the number of performances, constraints, and parameters is large. On the other hand, the discrete intermediate solutions computed on many nonlinear models considered during this sizing task were close to the discrete optimum such that a large number of Branch-and-Bound recursions was required to solve the model of the underlying problem in the MoPS approach. As a consequence, a large number of continuous optimization problems must be solved with a high runtime each.

It follows that the application of the MoPS approach is reasonable only in case the designed circuit can be evaluated only at discrete design parameter points and not at continuous intermediate points, i.e., if the BABA approach cannot be used. The most important reason for this conclusion is that – as in the nominal design case – the time required to solve the problem without considering the simulation time strongly contributes to the runtime of the MoPS approach. As in the nominal case, this time is predominantly required to solve the discrete non-linear model of the optimization problem and cannot be avoided, although it might be reduced by certain modifications of the MoPS algorithm.

As the results show that the runtime for the tolerance design is always significantly longer than the computational time for a nominal design, it should finally be demonstrated why a tolerance design is required and why it is not sufficient to increase the performances at the nominal point with respect to operation and process parameters until the required yield is reached. The reason can be observed, e.g., if the positive slew rate of the low voltage amplifier in Section 6.1.3 is considered. The slew rate is measured from the transient response of the output shown in Figure 6.9. In Figure 6.9a, the output curve for a sizing is given where the specification for the slew rate is fulfilled for nominal operating and process parameters. Figure 6.9b shows the transient response for a tolerance design. In both figures, the transient response is plotted for nominal values for operating and process parameters and for the worst case values which are computed for the positive slew rate. A comparison of the curves shows that the amplifier is much faster for the nominal design than for the tolerance design if operating and process parameters are



(a) Transient response for nominal and worst case point of the positive slew rate with design parameters chosen such that the slew rate specification is fulfilled at the nominal point



(b) Transient response for nominal and worst case point of the positive slew rate after computation of a tolerance design

Figure 6.9: Transient response of the low voltage amplifier in voltage follower configuration for a unit step from 0.5V to 1.5V at the input

considered at the nominal point (the measured values of the slew rate are more than $40\frac{V}{\mu s}$ for the nominal design compared to approximately $24\frac{V}{\mu s}$ for the tolerance design). However, at the computed worst case point for a $3\sigma$ design, the tolerance design of the circuit still fulfills the specification with a slew rate of $16\frac{V}{\mu s}$ whereas the nominal design can only achieve a slew rate of less than $10\frac{V}{\mu s}$ and does not solve the task. Thus, a nominal design with more restrictive specifications might not gain the required yield either because the computation of the tolerance design does not only improve the performances at the nominal point but it also makes the circuit less sensitive against variations of operating and process conditions.

It can be concluded from the results that the Branch-and-Bound approach for analog sizing is the first highly efficient approach to solve the analog sizing task for nominal and tolerance design. As it is restricted to problems where the circuit can be evaluated at each continuous point between lower and upper bound of the design parameters, an additional effort must be made if the circuit can only be evaluated for discrete points. For this case the model-based programming approach for analog sizing is presented in this thesis, which is less efficient than the Branch-and-Bound approach but still fast enough to solve both sizing tasks in practice.

# Chapter 7

# Conclusion

The design of new circuits is one of the most challenging and most complex tasks in the IC industry. To make the task easier to handle, the system to be designed is partitioned into several functional blocks which can be designed separately and which can be classified into analog, digital, and mixed-signal blocks. Nevertheless, tools to automate the design of these functional blocks are crucial to support the designer, to increase productivity, and to avoid the more error-prone manual design.

In contrast to the widely automated digital design flow, the design of analog blocks is still predominantly manual work. The first step in the analog design flow, the synthesis of the design schematic, is not automated at all, whereas the subsequent steps, the sizing and the layout of a circuit, are partly automated in practice.

This thesis is focused on the sizing of analog circuits. Although this step is partly automated, available commercial tools are of limited usability as none of these tools can consider variations in operating and process conditions along with discrete scalable parameters. However, discrete parameters are unavoidable if layout information, e.g., the number of winding turns of an integrated inductor, a parameter grid, or the transistor multiplier, should be considered in the sizing step to support the subsequent layout step. At the same time, considering variations in process and operating conditions in the sizing step is crucial to ensure the functionality of a circuit for the specified operation conditions after manufacturing. This problem is addressed in this thesis by means of two new deterministic approaches for analog sizing.

For this purpose, the sizing task was initially formulated as a discrete minimization problem. In this context, the computation of a gradient by finite differences was enlarged for a discrete sizing task and available objective functions were discussed and selected which are required to map the multi-objective sizing task to a single objective optimization problem. Furthermore, the required state-of-the-art optimization algorithms were introduced and the required enhancement for analog sizing was shown.

Two new algorithms, which are constructed for two different applications, were developed in this thesis:

- A fast Branch-and-Bound algorithm was realized for the sizing of analog circuits, assuming that the circuits can be evaluated for each continuous design point. This case can be found in practical problems if multipliers or manufacturing grids are to be considered during sizing.
- A model-based algorithm was implemented for the sizing of analog circuits where some parameters must be considered which can only be evaluated at discrete points. Such problems typically arise if the influence of a parameter on a device – in the examples the sides of an integrated inductor – is measured or computed for a certain discrete point and stored in a look-up table rather than modeled in a continuous scalable model as, e.g., BSIM.

Both approaches were enlarged by a successive approximation of the worst case points which allow the computation of a tolerance design, i.e., of a design which is robust against variations in operating and process conditions. At the same time, special care has been taken to keep the number of computationally expensive circuit simulations required to evaluate the circuit performances low.

The simulation results of the sizing of two operational amplifiers, one sense amplifier, and one low noise amplifier show that the new Branch-and-Bound based approach is highly efficient and was able to compute a nominal design – i.e., a design with all operating and process parameters set to fixed (nominal) values – as well as a tolerance design within approximately the same time as it is required for a sizing in the continuous domain. The results also show that the runtime required by the model-based approach to solve the nominal and tolerance design task is up to three times the time of the Branch-and-Bound based approach. However, this additional runtime is sustainable for the case that the circuit can be evaluated at discrete points only.

Comparing the newly developed algorithms with the state-of-the-art approach of continuous optimization and subsequent rounding, it can be observed that in contrast to the state-of-the art approach, which did not solve the sizing task in seven of the eight nominal and tolerance design cases shown, the result computed by the new approaches did always solve the specified design tasks.

Besides their relevance in practice, the approaches presented in this thesis can be seen as a basis for optimization and for the computation of Pareto fronts for analog circuits with discrete parameters. The application of the newly developed algorithms in these fields would require only a small number of modifications and enhancements. However, a considerable increase in computational time can be expected compared to the results shown in this thesis.

From a practical point of view, the results of this thesis can be used to enlarge current deterministic state-of-the-art tools in such a way that in the sizing step of the analog design flow discrete parameters, process variations, and operating conditions can be considered at the same time. The relevance of this ability can be expected to rise further in future because it can be expected that with decreasing structure sizes and new technologies more and more discrete parameters must be considered along with an increasing importance of the consideration of process variations. At the same time, an increasing need for analog design automation, which can only be satisfied by approaches as presented in this thesis, is to be anticipated to keep productivity in the IC industry at a high level.

# Appendix A

# Numerical Examples

## A.1 Continuous Optimization and Rounding vs Discrete Optimization

Figure A.1 illustrates that rounding of a continuous point may not solve a discrete optimization problem in general. The following simple linear optimization problem is considered:

$$\min -(x+y) \qquad \text{s.t.:} \quad \begin{aligned} x &\in \{0,1,2,3\} \\ 0 \le\ &y \le 3 \\ x + 2y &\le 6 \\ 20x + 4y &\le 45 \end{aligned} \tag{A.1}$$

The discrete solution of this optimization problem is point F in Figure A.1 with $(x, y) = (1, 2.5)$ and objective function value $-(x + y) = -3.5$. This value is obviously

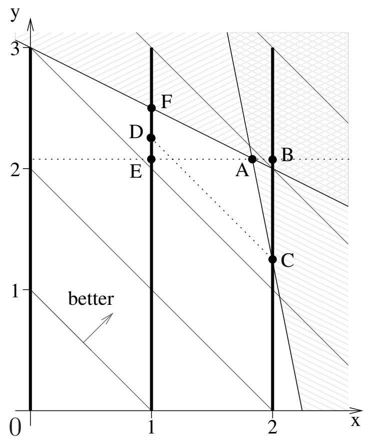


Figure A.1: Linear discrete optimization example

larger than the value at the continuous solution $(x, y) \approx (1.83, 2.08)$ (point A in Figure A.1) which is derived after replacing $x \in \{0, 1, 2, 3\}$ by the relaxation $0 \leq x \leq 3$ and which has the objective function value $-3.91$.

If the discretization of the continuous solution is to be done by rounding to the next discrete point (point B in Figure A.1 with $(x, y) \approx (2, 2.08)$), two constraints are violated. Also rounding the continuous solution to the next discrete point and a subsequent optimization over the continuous parameters to find the best point which fulfills the constraints might not solve the task. The point which can be found in this way for the example (point C in Figure A.1 with $(x, y) = (2, 1.25)$) has an objective function value of $-3.25$ and is significantly worse than the discrete optimum. For graphical comparison, point D is added to Figure A.1 which has the same objective function value as point C but has the same value for $x$ as the discrete optimum.

Point E with $(x, y) \approx (1, 2.08)$ shows that a discretization of the continuous point by rounding to the next lower value does not solve the discrete optimization problem either. In this example, the objective function value of this point with $-3.08$ is even worse than the objective function value of every other solution which was considered in the example.

The example shows that – although a simple linear discrete optimization task was considered – discrete optimization cannot be replaced by continuous optimization and rounding.

## A.2 Examples for Objective Functions

A multi-objective problem with two real-valued continuous parameters $p_1, p_2 \in \mathbb{R}^2$ and four performances $f_1(\mathbf{p}), ..., f_4(\mathbf{p})$ is given. Constraints are not considered in this example as they are not included into the objective function within this thesis and are not required for the following observations. A specification is defined for each performance:

| Performance | Specification |
|---|---|
| $f_1 = p_1^2 + p_2^2$ | $f_1 \leq 1$ |
| $f_2 = p_1 + p_2$ | $f_2 \leq -1$ |
| $f_3 = p_1 + p_2$ | $f_3 \geq -2$ |
| $f_4 = p_1$ | $f_4 \geq -1$ |

(A.2)

Using (2.37), the performance-to-specification gap can be defined:

| Performance-to-Specification Gap | Specification |
|---|---|
| $\varepsilon_1 = p_1^2 + p_2^2 - 1$ | $\varepsilon_1 \leq 0$ |
| $\varepsilon_2 = p_1 + p_2 + 1$ | $\varepsilon_2 \leq 0$ |
| $\varepsilon_3 = -\frac{1}{2}p_1 - \frac{1}{2}p_2 - 1$ | $\varepsilon_3 \leq 0$ |
| $\varepsilon_4 = -p_1 - 1$ | $\varepsilon_4 \leq 0$ |

(A.3)

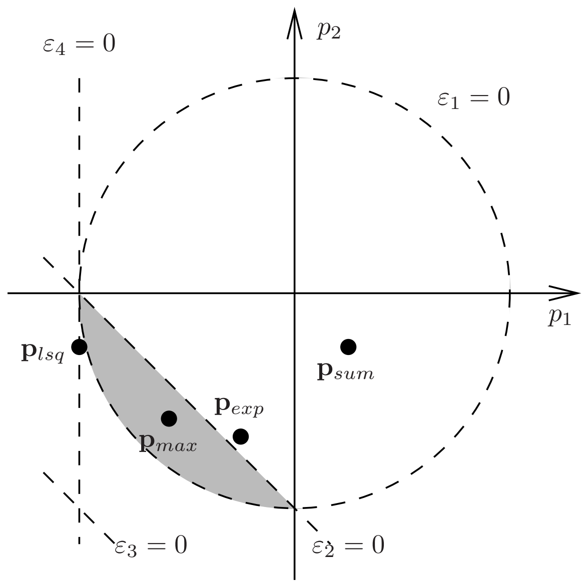


Figure A.2: Example for a multi-objective problem

The problem is visualized in Figure A.2 where the task is to find a point in the shaded area. The problem should be solved by a minimization problem of the form

$$\min \varphi(\mathbf{p}) \tag{A.4}$$

To realize the mapping from a multi-objective problem to a scalar, a sum of errors

$$\varphi_{sum}(\mathbf{p}) = \sum_{i=1}^{4} \varepsilon_i(\mathbf{p}) \tag{A.5}$$

a least-squares approach

$$\varphi_{lsq}(\mathbf{p}) = \sum_{i=1}^{4} \varepsilon_i^2(\mathbf{p}) \tag{A.6}$$

an exponentially weighted sum

$$\varphi_{exp}(\mathbf{p}) = \sum_{i=1}^{4} \exp\left(\varepsilon_i(\mathbf{p})\right) \tag{A.7}$$

and a max-norm formulation

$$\varphi_{max}(\mathbf{p}) = \max_{i=1,\ldots,4} \left(\varepsilon_i(\mathbf{p})\right) \tag{A.8}$$

are used.

Following the claims in Chapter 2.4.1, the objective function should attach greater weight to the alteration of the worse of two objectives if both alterations are comparably large in size. As each term in the sum of errors contributes linearly to the objective function, the sum of errors violates this claim. As a consequence, and as also claim 4 in Chapter 2.4.1 is violated, the optimum of this function (point $\mathbf{p}_{sum}$ in Figure A.2) is far away from the specified (shaded) region and the specification for $\varepsilon_2$ is violated. In contrast, the claim is fulfilled for all other objective functions in this section because each size of a performance gap in the least-squares approach and in the exponentially weighted sum is weighted quadratically and exponentially, respectively, and only the worst value is considered by the max-norm.

The second claim in Chapter 2.4.1 requires the objective function not to penalize the over-fulfillment of specifications. This claim is not fulfilled for the least-squares objective function: Considering that a performance fulfills the specification if $\varepsilon_i \leq 0$ and putting such a value into the four objective functions above, it can be observed that only the least-squares approach is increased if a negative size of the performance gap is further decreased. As a consequence, the least-squares distance to all bounds is minimized for objective function (A.6) and an over-fulfillment of the specifications is not supported. In the example this leads to a solution $\mathbf{p}_{lsq}$ (cf. Figure A.2) which does not fulfill the specification for $\varepsilon_1$.

The last claim in Chapter 2.4.1 is to use an objective function ensuring that a point which fulfills the specifications has a better objective function value than any point where a specification is violated. The results for the least-squares approach – which is penalized if some specifications are over-fulfilled – and the sum of errors – which attaches the same weight to the improvement of each performance – show that this claim is violated for these two objective functions. However, the exponential sum also violates this claim because the exponential weighting of each size of a performance gap can result in a case where the influence of many over-fulfilled specifications prevents some other performances from achieving the specified value. Such a case can be observed in the example where the optimum of the exponential sum ($\mathbf{p}_{exp}$ in Figure A.2) does not fulfill all specifications.

It can be seen that in the example only for the max-norm formulation all performance specifications are fulfilled. Due to the formulation of the performance-to-specification gap – i.e., as the value of the gap is always negative if the specifications are fulfilled and positive if not – a global minimum of the max-norm is always a point which fulfills the specifications if such a point exists.

The truncated forms of an objective function – especially the truncated least squares approach (cf. Chapter 2.4.2) – can also be used to solve the problem. In this case, any point in the shaded area in Figure A.2 can be found because each of these points has an objective function value of zero. The result found if a truncated function is used depends on the starting point and on the algorithm used to solve the optimization program.

## A.3 Optimization Example

In this section the following non-linear discrete optimization problem is considered:

$$\begin{aligned}
\min \quad & \varphi(d_1, d_2) := \tfrac{1}{2} \cdot (d_1^2 + d_2^2) - 6 \cdot (d_1 + d_2) \\
\text{s.t.} \quad & c_1(d_1, d_2) := 36 - d_1^2 - d_2^2 \geq 0 \\
& c_2(d_1, d_2) := 0.5 - d_1 + d_2 \geq 0 \\
& d_1, d_2 \in \{0, 1, 2, 3, 4, 5, 6\}
\end{aligned} \tag{A.9}$$

The discrete solution of the problem can be found at $(d_1^*, d_2^*) = (4, 4)$.

### A.3.1 Example for Feasible Sequential Quadratic Programming

If the problem is to be solved by a standard Branch-and-Bound algorithm, a solution for the relaxed optimization problem is required. As constraint $c_2$ does not influence the result of the relaxed problem, it is neglected in this section. I.e., the following relaxation of (A.9) must be solved:

$$\begin{aligned}
\min \quad & \varphi(d_1, d_2) := \tfrac{1}{2} \cdot (d_1^2 + d_2^2) - 6 \cdot (d_1 + d_2) \\
\text{s.t.} \quad & c_1(d_1, d_2) := 36 - d_1^2 - d_2^2 \geq 0 \\
& 0 \leq d_1 \leq 6 \\
& 0 \leq d_2 \leq 6
\end{aligned} \tag{A.10}$$

The optimum of the problem is $(d_1^*, d_2^*) \approx (4.24, 4.24)$

To solve the optimization problem, an FSQP algorithm (cf. Chapter 3.4.5) can be used. For this purpose an initial feasible point must be given. Let the initial point be $\mathbf{d}^{(0)} = [d_1^{(0)}, d_2^{(0)}]^T = [0, 6]^T$ such that the gradient for the objective function and the non-linear constraint at this point can be computed as

$$\begin{aligned}
\nabla_{\mathbf{d}} \varphi(d_1, d_2)\big|_{d_1=0, d_2=6} &= [-6, 0]^T \\
\nabla_{\mathbf{d}} c_1(d_1, d_2)\big|_{d_1=0, d_2=6} &= [0, -12]^T
\end{aligned} \tag{A.11}$$

Typically, the identity matrix is used to initialize the Hessian matrix of the Lagrangian function. Thus, the quadratic subproblem for a standard SQP approach which is solved in the first iteration step is

$$\begin{aligned}
\min_{\Delta \mathbf{d}^{(0)}} \quad & \tfrac{1}{2} \cdot \left( \Delta d_1^{(0)^2} + \Delta d_2^{(0)^2} \right) - 6 \cdot \Delta d_1^{(0)} \\
\text{s.t.} \quad & -12 \cdot \Delta d_2^{(0)} \geq 0 \\
& 0 \leq \Delta d_1^{(0)} \leq 6 \\
& -6 \leq \Delta d_2^{(0)} \leq 0
\end{aligned} \tag{A.12}$$

with optimum $\Delta\mathbf{d}^{(0)^T} = [\Delta d_1^{(0)}, \Delta d_2^{(0)}]^T = [6, 0]^T$. It can be seen in Figure 3.4 on Page 62 that no step length $\delta \neq 0$ along $\mathbf{d}^{(0)} + \delta\Delta\mathbf{d}^{(0)}$ leads to a feasible point.

To overcome this problem, a tilted direction is computed in each step of an FSQP algorithm by solving subproblem (3.68). As a consequence – instead of subproblem (A.12) – the following quadratic problem is solved in the first step of the FSQP algorithm for the given problem:

$$\begin{aligned} \min_{\Delta\mathbf{d}^{(0)}} \quad & \tfrac{1}{2} \cdot \left(\Delta d_1^{(0)^2} + \Delta d_2^{(0)^2}\right) + \gamma \\ s.t. \quad & \gamma + 6 \cdot \Delta d_1^{(0)} \geq 0 \\ & \gamma \cdot \eta - 12 \cdot \Delta d_2^{(0)} \geq 0 \\ & 0 \leq \Delta d_1^{(0)} \leq 6 \\ & -6 \leq \Delta d_2^{(0)} \leq 0 \end{aligned} \tag{A.13}$$

for some $\eta > 0$. The step can be computed depending on $\eta$ as:

$$\Delta\mathbf{d}^{(0)^T} = \begin{bmatrix} \frac{6}{1+\frac{1}{4}\eta^2} & , & \frac{-3\cdot\eta}{1+\frac{1}{4}\eta^2} \end{bmatrix}^T \tag{A.14}$$

It can be seen that for $\eta \to 0$ the computed step $\Delta\mathbf{d}^{(0)}$ is equal to the result of (A.12) and for $\eta \to \infty$ the step goes to zero. Some examples for different values of $\eta$ are shown in Figure 3.5 on Page 64. It can be seen that for typical values $0 < \eta \ll 1$ only a small step length $\delta$ can be chosen along $\mathbf{d}^{(0)} + \delta\Delta\mathbf{d}^{(0)}$ if the resulting point is to be feasible. Thus, a correction is computed for the step in the FSQP algorithm.

Setting the value of $\eta$ to $\eta = 0.1$ results in a step $\Delta\mathbf{d}^{(0)^T} \approx [5.99, -0.30]^T$. For the full step $\Delta\mathbf{d}^{(0)}$ the non-linear constraint is violated and a correction step is computed in the FSQP algorithm. Optimization problem (3.70) must be solved to compute the correction step in the FSQP algorithm. For the example, the explicit optimization problem can be given as:

$$\begin{aligned} \min_{\widehat{\Delta\mathbf{d}}} \quad & \tfrac{1}{2} \cdot \left(\Delta\widehat{d}_1^2 + \Delta\widehat{d}_2^2\right)^T - 0.01 \cdot \Delta\widehat{d}_1 - 0.30\Delta\widehat{d}_2 \\ \text{s.t.} \quad & -12 \cdot \Delta\widehat{d}_2 - 32.43 \geq 0 \end{aligned} \tag{A.15}$$

This results in a correction step $\widehat{\Delta\mathbf{d}}^T = [\Delta\widehat{d}_1, \Delta\widehat{d}_2]^T = [0.01, -2.70]^T$. Although the step $\Delta\mathbf{d}^{(0)} + \widehat{\Delta\mathbf{d}}$ is closer to the bound, it does not fulfill the constraint. The correction step is shown in Figure 3.6 on page 65.

After the computation of tilted direction and correction step, an arc search along this direction is executed. This is shown in Figure 3.7 on page 66. It can be seen that a feasible step can be computed along the arc, and that the step is larger and yields more improvement of the objective function than the longest feasible step along the tilted direction.

### A.3.2 Example for Standard Branch-and-Bound

Now the complete optimization problem (A.9) is considered. Using the Branch-and-Bound algorithm from Chapter 3.5.4, the steps to solve the given problem are as follows:

1. Initially, the relaxed problem is solved, which results in the relaxed solution $(d_1^*, d_2^*) \approx (4.24, 4.24)$ with objective function value $\varphi(4.24, 4.24) = -32.90$ (cf. Figure 3.10a on Page 77). This result is a lower bound for the discrete solution.

2. Using the solution of the relaxed problem, a rounding constraint is added which ensures that $d_1 \leq 4$ (cf. Figure 3.10b on Page 77). The solution in this subproblem is $(d_1^*, d_2^*) \approx (4, 4.47)$ with $\varphi(4, 4.47) = -32.82$. The result is a lower bound for the discrete solution in the subdomain with $d_1 \leq 4$, but not for the original domain.

3. The next rounding constraint which is added is $d_2 \leq 4$. The discrete solution of this subproblem is $(d_1^*, d_2^*) \approx (4, 4)$ with $\varphi(4, 4) = -32$ ((cf. Figure 3.10c on Page 77). This result is not only a lower bound for the discrete solution in the subdomain, but also – as it is discrete and no better discrete solution has been found yet – an upper bound for the discrete optimum.

4. As a discrete solution has been found, the next subdomain considered is defined by the rounding constraints $d_1 \leq 4$ and $d_2 \geq 5$ (cf. Figure 3.10d on Page 77). The result for this subdomain is $(d_1^*, d_2^*) \approx (3.32, 5)$ with $\varphi(3.32, 5) = -31.91$. As the result is worse than the result of the incumbent solution, it can be concluded that the discrete optimum cannot be in this subdomain.

5. The last subdomain which must be considered is defined by the rounding constraint $d_1 \geq 5$ (cf. Figure 3.10e on Page 77). This subdomain does not include any feasible solution. However, after considering this subdomain, all required subproblems have been examined. It follows that the best solution yet, i.e., $(d_1^*, d_2^*) \approx (4, 4)$, is the discrete optimum for the problem.

# Appendix B

# Derivations

## B.1 Gradient Computation using Quadratic Performance Approximation

The performances which are evaluated in this thesis are assumed to be differentiable if the discrete problem is relaxed. Thus, at each point in the discrete domain a derivative of every performance $f_k$ with respect to each parameter $p_i$ can be defined.

For continuous parameters and discrete parameters on a uniform grid the approximation of the gradient can be computed by the central form of finite differences. However, if a parameter can only take non-equidistant discrete values, the symmetrically defined central form of finite differences can not be computed.

To overcome this problem, each performance $f_k$ is modeled at a point $\mathbf{p}$ and in direction of parameter $p_i$ by a function

$$f_k(p_i + \Delta p_i) \approx a_k \cdot \Delta {p_i}^2 + b_k \cdot \Delta p_i + c_k \tag{B.1}$$

The parameters $a_k$, $b_k$, and $c_k$ can be computed using three sampling points. Without loss of generality, one sampling point can be set to $\Delta p_i = 0$ with performance value $f_{k,0}$. I.e., the performance is evaluated at the point where the gradient should be computed. Two additional points can be set to $\Delta p_1 \neq 0$ and $\Delta p_2 \neq 0$ with performance value $f_{k,1}$ and $f_{k,2}$, respectively, assuming that $\Delta p_1$ and $\Delta p_2$ lead to points which can be evaluated.

Using $\Delta p_1$,$\Delta p_2 \neq \Delta p_1$, $f_{k,0}$, $f_{k,1}$, and $f_{k,2}$, and assuming a quadratic form (B.1) for the underlying function, a linear equation system in $a_k$, $b_k$, and $c_k$ can be established:

$$\begin{array}{ccccccccccc} 0 & \cdot & a_k & + & 0 & \cdot & b_k & + & c_k & = & f_{k,0} \\ \Delta p_1^2 & \cdot & a_k & + & \Delta p_1 & \cdot & b_k & + & c_k & = & f_{k,1} \\ \Delta p_2^2 & \cdot & a_k & + & \Delta p_2 & \cdot & b_k & + & c_k & = & f_{k,2} \end{array} \tag{B.2}$$

This linear equation system has a unique solution for $a_k$, $b_k$, and $c_k$:

$$\begin{aligned} c_k &= f_{k,0} \\ b_k &= \tfrac{\Delta p_2}{\Delta p_2 - \Delta p_1} \cdot \tfrac{f_{k,1} - f_{k,0}}{\Delta p_1} + \tfrac{\Delta p_1}{\Delta p_1 - \Delta p_2} \cdot \tfrac{f_{k,2} - f_{k,0}}{\Delta p_2} \\ a_k &= \tfrac{f_{k,1} - f_{k,0}}{\Delta p_1^2} - \tfrac{b}{\Delta p_1} \end{aligned} \tag{B.3}$$

The sensitivity of a performance $f_{k,1}$ with respect to a parameter $p_i$ at a point $\mathbf{p}$ can be defined as:

$$j_{k,i} = \left.\frac{\partial f_k}{\partial p_i}\right|_{\mathbf{p}} \approx \frac{\partial}{\partial \Delta p_i} \left( a_k \cdot \Delta {p_i}^2 + b_k \cdot \Delta p_i + c_k \right) \Big|_{\Delta p_i = 0} = b_k \tag{B.4}$$

It can be observed that for a symmetric deflection $\Delta p = \Delta p_1 = -\Delta p_2 > 0$ the value of $b_k$ is equal to

$$b_k = \frac{f_{k,1} - f_{k,2}}{2 \cdot \Delta p} \tag{B.5}$$

i.e., the approximation is equal to the central form of finite differences.

## B.2 Yield Approximation

This section shows the derivation of the yield approximation for $\mathcal{N}(\mathbf{s}_0, \mathbf{C})$-distributed process parameters $\mathbf{s}$ according to [Gra07].

Let $\mathbf{s}_{wci}$ be the worst point – i.e., the point with the highest value of $\varepsilon_i(\mathbf{s})$ – in an ellipsoid defined by setting $\beta^2(\mathbf{s})$ in (2.24) to $\beta^2(\mathbf{s}) = \beta_W^2$. Then $\mathbf{s}_{wci}$ solves

$$\max_{\mathbf{s}} \varepsilon_i(\mathbf{s}) \text{ s.t. } (\mathbf{s} - \mathbf{s}_0)^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{s} - \mathbf{s}_0) \leq \beta_W^2 \tag{B.6}$$

Establishing the Lagrangian function for (B.6) and applying the first-order optimality conditions (cf. Section 3.1), $\mathbf{s}_{wci}$ can be expressed as [Gra07]:

$$(\mathbf{s}_{wci} - \mathbf{s}_0) = \frac{\beta_W}{\sqrt{\mathbf{j}_{wc,i}^T \cdot \mathbf{C} \cdot \mathbf{j}_{wc,i}}} \mathbf{C} \cdot \mathbf{j}_{wc,i} \tag{B.7}$$

with

$$\mathbf{j}_{wc,i} = \left.\frac{\partial \varepsilon_i(\mathbf{s})}{\partial \mathbf{s}^T}\right|_{\mathbf{s} = \mathbf{s}_{wci}} . \tag{B.8}$$

A linearized model for the performance-to-specification gap $\varepsilon_i$ with respect to the process parameters at $\mathbf{s}_{wci}$ can be given as:

$$\varepsilon_i(\mathbf{s}) \approx \bar{\varepsilon}_i(\mathbf{s}) = \varepsilon_i(\mathbf{s}_{wci}) + \mathbf{j}_{wc,i}^T \cdot (\mathbf{s} - \mathbf{s}_{wci}) \tag{B.9}$$

For the variance $V\left\{\bar{\varepsilon}_i(\mathbf{s})\right\}$,

$$\sigma_{\bar{\varepsilon}_i}^2 := V\left\{\bar{\varepsilon}_i(\mathbf{s})\right\} = V\left\{\varepsilon_i(\mathbf{s}_{wci}) + \mathbf{j}_{wc,i} \cdot (\mathbf{s} - \mathbf{s}_{wci})\right\} = \mathbf{j}_{wc,i}^T V\left\{\mathbf{s}\right\} \mathbf{j}_{wc,i} = \mathbf{j}_{wc,i}^T \cdot \mathbf{C} \cdot \mathbf{j}_{wc,i} \tag{B.10}$$

must hold true [Gra07, Koc99]. From (B.7) and (B.9) follows

$$\begin{aligned} \bar{\varepsilon}_i(\mathbf{s}_0) &= \varepsilon_i(\mathbf{s}_{wci}) + \mathbf{j}_{wc,i}^T \cdot (\mathbf{s}_0 - \mathbf{s}_{wci}) \\ \Leftrightarrow \varepsilon_i(\mathbf{s}_{wci}) &= \bar{\varepsilon}_i(\mathbf{s}_0) + \mathbf{j}_{wc,i}^T \cdot (\mathbf{s}_{wci} - \mathbf{s}_0) \\ &\overset{(B.7)}{=} \bar{\varepsilon}_i(\mathbf{s}_0) + \beta_W \sqrt{\mathbf{j}_{wc,i}^T \cdot \mathbf{C} \cdot \mathbf{j}_{wc,i}} \overset{(B.10)}{=} \bar{\varepsilon}_i(\mathbf{s}_0) + \beta_W \sigma_{\bar{\varepsilon}_i} \\ \Leftrightarrow \beta_W &= \tfrac{\varepsilon_i(\mathbf{s}_{wci}) - \bar{\varepsilon}_i(\mathbf{s}_0)}{\sigma_{\bar{\varepsilon}_i}} \end{aligned} \tag{B.11}$$

The values of the linearly modeled performance gap sizes according to (B.9) are – due to the linearity of $\bar{\varepsilon}_i(\mathbf{s})$ and the normal distribution of $\mathbf{s}$ – $\mathcal{N}\left(\bar{\varepsilon}_i(\mathbf{s}_0), \sigma_{\bar{\varepsilon}_i}^2\right)$-distributed [Gra07]. Therefore, the probability of the value of the performance gap size being smaller than a certain value $\varepsilon_i(\mathbf{s}_{wci})$ can be approximated as

$$\bar{Y}_i = \int\limits_{-\infty}^{\varepsilon_i(\mathbf{s}_{wci})} \frac{1}{\sqrt{2\pi} \cdot \sigma_{\bar{\varepsilon}_i}} \exp\left(-\frac{1}{2}\left(\frac{\bar{\varepsilon}_i - \bar{\varepsilon}_i(\mathbf{s}_0)}{\sigma_{\bar{\varepsilon}_i}}\right)^2\right) d\bar{\varepsilon}_i \tag{B.12}$$

or, substituting $\frac{\bar{\varepsilon}_i - \bar{\varepsilon}_i(\mathbf{s}_0)}{\sigma_{\bar{\varepsilon}_i}}$ by $\beta$,

$$\bar{Y}_i = \int\limits_{-\infty}^{\frac{\varepsilon_i(\mathbf{s}_{wci}) - \bar{\varepsilon}_i(\mathbf{s}_0)}{\sigma_{\bar{\varepsilon}_i}}} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\beta^2\right) d\beta \overset{(B.11)}{=} \int\limits_{-\infty}^{\beta_W} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\beta^2\right) d\beta \tag{B.13}$$

In the special case that $\varepsilon_i(\mathbf{s}_{wci}) = 0$ – i.e., if performance specification $i$ is fulfilled exactly at $\mathbf{s}_{wci}$ – $\bar{Y}_i$ is an approximation for the yield of the circuit with respect to performance $i$ and for a given specification bound.

# Bibliography

[ABD03] G. Alpaydin, S. Balkir, G. Dündar: *An Evolutionary Approach to Automatic Synthesis of High-Performance Analog Integrated Circuits*, IEEE Transactions on Evolutionary Computation, volume 7(3), pages 240–252, June 2003.

[AEG$^+$00] K. Antreich, J. Eckmueller, H. Graeb, M. Pronath, F. Schenkel, R. Schwencker, S. Zizala: *WiCkeD: Analog Circuit Synthesis Incorporating Mismatch*, in: *IEEE Custom Integrated Circuits Conference*, pages 511–514, May 2000.

[AG] K. J. Antreich, H. Graeb: *Circuit Optimization Driven by Worst-Case Distances*, in: A. Kuehlmann (ed.), *The Best of ICCAD - 20 Years of Excellence in Computer-Aided Design*, pages 585–595, Springer.

[AKM05] T. Achterberg, T. Koch, A. Martin: *Branching Rules Revisited*, Operations Research Letters, volume 33(1), pages 42–54, January 2005.

[ALL] K. Abhishek, S. Leyffer, J. Linderoth: *FilMINT: An Outer Approximation-Based Solver for Convex Mixed-Integer Nonlinear Programs.*

[AS94] S. A. Aftab, M. A. Styblinski: *IC Variability Minimization Using a New Cp and Cpk Based Variability/Performance Measure*, in: *IEEE International Symposium on Circuits and Systems*, pages 793–805, June 1994.

[BKVH07] S. Boyd, S.-J. Kim, L. Vadenberghe, A. Hassibi: *A Tutorial on Geometric Programming*, Optimization and Engineering, volume 8(1), pages 67–127, March 2007.

[BSI11] *BSIM (Berkeley Short-channel IGFET Mode)*, www-device.eecs.berkeley.edu/˜bsim3/, 2011.

[CAD11] *Cadence Design Framework* , www.cadence.com, 2011.

[CCH$^+$96] A. Conn, P. Coulman, R. Haring, G. Morill, C. Visweswariah: *Optimization of Custom MOS Circuits by Transistor Sizing*, in: *IEEE/ACM International Conference on Computer-Aided Design*, pages 174–180, November 1996.

[CM10] T. Chen, J. Ma: *Advances in Bipolar Junction Transistor Modeling*, in: 10*th* *IEEE International Conference on Solid-State and Integrated Circuit Technology*, pages 1749–1752, November 2010.

[Dan66] G. B. Dantzig: *Lineare Programmierung und Erweiterung (Ökonometrie und Unternehmensforschung)*, Springer, 1966.

[DCR05] T. R. Dastidar, P. P. Chakrabarti, P. Ray: *A Synthesis System for Analog Circuits Based on Evolutionary Search and Topological Reuse*, IEEE Transactions on Evolutionary Computation, volume 9(2), pages 211–224, April 2005.

[DGS03] W. Daems, G. Gielen, W. Sansen: *Simulation-Based Automatic Generation of Signomial and Posynomial Performance Models for Analog Integrated Circuit Sizing*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume 22(5), pages 517–534, May 2003.

[DK95] A. Dharchoudhury, S. M. Kang: *Worst-Case Analysis and Optimization of VLSI Circuit Performances*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume 14(4), pages 481–492, April 1995.

[dMH04] M. d. M. Hershenson: *CMOS Analog Circuit Design via Geometric Programming*, in: *American Control Conference*, pages 3266–3271, June 2004.

[Eck98] J. Eckmüller: *Zur rechnergestützten Dimensionierung analoger integrierter Schaltungen unter besonderer Berücksichtigung von Struktureigenschaften*, Ph.D. thesis, Technische Universität München, 1998.

[EG02] M. Ehrgott, E. A. Galperin: *Min-Max Formulation of the Balance Number in Multiobjective Global Optimization*, Computers & Mathematics with Applications, volume 44(7), pages 899–907, October 2002.

[ES] O. Exler, K. Schittkowksi: *A Trust Region SQP Algorithm for Mixed-Integer Nonlinear Programming.*

[Esh92] K. S. Eshbaugh: *Generation of Correlated Parameters for Statistical Circuit Simulation*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume 11(10), pages 1198–1206, October 1992.

[GDL$^+$95] G. Gielen, G. Debyser, K. Lampaert, F. Leyn, K. Swings, G. D. V. Plas, W. Sansen, D. Leenaerts, P. Veselinovic, W. van Bokhoven: *An Analogue Module Generator for Mixed Analogue/Digital ASIC Design*, International Journal of Circuit Theory and Applications, volume 23(4), pages 296–283, August 1995.

[GL97] F. Glover, M. Laguna: *Tabu Search*, Springer, 1997.

[GM79] P. E. Gill, W. Murray: *The Computation of Lagrange-Multiplier Estimates for Constrained Minimization*, Mathematical Programming, volume 17(1), pages 32–60, December 1979.

[GR00] G. G. E. Gielen, R. A. Rutenbar: *Computer-Aided Design of Analog and Mixed-Signal Integrated Circuits*, Proceedings of the IEEE, volume 88(12), pages 1825–1854, December 2000.

[Gra07] H. Graeb: *Analog Design Centering and Sizing*, Springer, 2007.

[Gre70] J. Greenstadt: *Variations on Variable-Metric Methods. (With Discussion)*, Mathematics of Computation, volume 24(109), pages 1–22, January 1970.

[GWS90] G. G. E. Gielen, H. C. C. Walscharts, W. M. C. Sansen: *Analog Circuit Design Optimization Based on Symbolic Simulation and Simulated Annealing*, IEEE Journal of Solid-State Circuits, volume 25(3), pages 707–713, June 1990.

[Has01] A. Hastings: *The Art of Analog Layout*, Prentice-Hall, 2001.

[HC04] D. Han, A. Chatterjee: *Simulation-in-the-Loop Analog Circuit Sizing Method Using Adaptive Model-Based Simulated Annealing*, in: *IEEE International Workshop on System-on-Chip for Real-Time Applications*, pages 127–130, July 2004.

[ITR09] ITRS: *International Technology Roadmap for Semiconductors 2009 Edition Design*, www.itrs.net/links/2009itrs/2009Chapters_2009Tables/2009_Design.pdf, 2009.

[ITR10] ITRS: *International Technology Roadmap for Semiconductors Design Chapter 2010 Update*, www.itrs.net/links/2010itrs/2010Update/ToPost/2010_DesignUpdate_ITRS.pdf, 2010.

[KD95] K. Krishna, S. W. Director: *The Linearized Performance Penalty (LPP) Method for Optimization of Parametric Yield and its Reliability*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume 14(12), pages 1557–1568, December 1995.

[KGV83] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi: *Optimization by Simulated Annealing*, Science, volume 220(4598), pages 671–680, May 1983.

[KH06] Z. Kaiping, S. A. Huss: *Structure Synthesis of Analog and Mixed-Signal Circuits Using Partition Techniques*, in: *IEEE International Symposium on Quality Electronic Design*, pages 230–235, March 2006.

[KKC$^+$08] H.-S. Kim, J. Kim, C. Chung, J. Lim, J. Jeong, J. H. Joe, J. Park, K.-W. Park, H. Oh, J. S. Yoon: *Effects of Parasitic Capacitance, External Resistance, and Local Stress on the RF Performance of the Transistors Fabricated by Standard 65-nm CMOS Technologies*, IEEE Transactions on Electron Devices, volume 55(10), pages 2712–2717, October 2008.

[Koc99] K. R. Koch: *Parameter Estimation and Hypothesis Testing in Linear Models*, Springer, 1999.

[LAG$^+$03] T. Ludwig, I. Aller, V. Gernhoefer, J. Keinert, E. Nowak, R. Joshini, A. Mueller, S. Tomaschko: *FinFET Technology for Future Microprocessors*, in: *IEEE Silicon-on-Insulator Conference*, pages 33–34, November 2003.

[Law98] C. Lawrence: *A Computationally Efficient Feasible Sequential Quadratic Programming Algorithm*, Ph.D. thesis, University of Maryland, 1998.

[Lee04] T. H. Lee: *The Design of CMOS Radio-Frequency Integrated Circuits*, Cambridge University Press, 2004.

[LFG11] B. Liu, F. V. Fernàndez, G. G. E. Gielen: *Efficient and Accurate Statistical Analog Yield Optimization and Variation Aware Circuit Sizing Based on Computational Intelligence Techniques*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume 30(6), pages 793–805, June 2011.

[Lin05] J. G. Lin: *On Min-Norm and Min-Max Methods of Multi-Objective Optimization*, Mathematical Programming, volume 103(1), pages 1–33, May 2005.

[LS06] D. Li, X. Sun: *Nonlinear Integer Programming*, Springer, 2006.

[LT01] C. Lawrence, A. Tits: *A computationally efficient feasible quadratic programming algorithm*, SIAM Journal on Optimization, volume 11(4), pages 1092–1118, March/May 2001.

[Mar98] R. Martins: *On the Design of Very Low Power Integrated Circuits*, Ph.D. thesis, Vienna University of Technology, 1998.

[Mas10] T. Massier: *On the Structural Analysis of CMOS and Bipolar Analog Integrated Circuits*, Ph.D. thesis, Technische Universität München, 2010.

[MCR92] P. C. Maulik, L. R. Carley, R. A. Rutenbar: *A Mixed-Integer Nonlinear Programming Approach to Analog Circuit Synthesis*, in: $29^{th}$ *ACM/IEEE Design Automation Conference*, pages 698–703, June 1992.

[MCR95] P. C. Maulik, L. R. Carley, R. A. Rutenbar: *Integer Programming Based Topology Selection of Cell-Level Analog Circuits*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume 14(4), pages 401–412, June 1995.

[MG09] D. Müller-Gritschneder: *Deterministic Performance Space Exploration of Analog Integrated Circuits considering Process Variations and Operating Conditions*, Ph.D. thesis, Technische Universität München, 2009.

[MGG10] D. Mueller-Gritschneder, H. Graeb: *Computation of Yield-Optimized Pareto Fronts for Analog Integrated Circuit Specifications*, in: *Design, Automation and Test in Europe*, pages 1088–1093, March 2010.

[MGS08] T. Massier, H. Graeb, U. Schlichtmann: *The Sizing Rules Method for CMOS and Bipolar Analog Integrated Circuit Synthesis*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume 27(12), pages 2209–2222, December 2008.

[Moo65] G. E. Moorer: *Cramming More Components onto Integrated Circuits*, Electronics, volume 38(8), pages 114–117, April 1965.

[MV01] P. Mandal, V. Visvanathan: *CMOS Op-Amp Sizing Using a Geometric Programming Formulation*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume 20(1), pages 22–38, January 2001.

[NAL$^+$04] E. J. Nowak, I. Aller, T. Ludwig, K. Kim, R. V. Joshi, C.-T. Chuang, K. Bernstein, R. Puri: *Turning Silicon on its Edge [double gate CMOS/FinFET technology]*, IEEE Circuits and Devices Magazine, volume 20(1), pages 20–31, August 2004.

[NKZB94] K. G. Nichols, T. J. Kazmierski, M. Zwolinski, A. D. Brown: *Overview of SPICE-Like Circuit Simulation Algorithms*, in: *IEE Circuits, Devices and Systems*, pages 242–250, August 1994.

[NW88] G. L. Nemhauser, L. A. Wolsey: *Integer and Combinatorial Optimization*, Jon Wiley & Sons, Inc., 1988.

[NW99] J. Nocedal, S. J. Wright: *Numerical Optimization*, Springer, 1999.

[ORC96] E. S. Ochotta, R. A. Rutenbar, L. R. Calrley: *Synthesis of High-Performance Analog Circuits in ASTRX/OBLX*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume 15(3), pages 273–294, March 1996.

[PD00] P. J. Pahl, R. Damrath: *Mathematische Grundlagen der Ingenieurinformatik*, Springer, 2000.

[PDML94] J. Power, B. Donellan, A. Mathewson, W. Lane: *Relating Statistical MOSFET Model Parameter Variabilities to IC Manufacturing Process Fluctuations Enabling Realist Worst-Case Design*, IEEE Transactions on Semiconductor Manufacturing, volume 7(3), pages 306–318, August 1994.

[PFC09] P. Pereira, M. H. Fino, F. V. Coito: *Using Discrete-Variable Optimization for CMOS Spiral Inductor Design*, in: *International Conference on Microelectronics (ICM)*, pages 324–327, December 2009.

[PG09] M. Pehl, H. Graeb: **RaGAzi***: A* **r***andom* **a***nd* **g***radient-Based Approach to* **a***nalog Si***zi***ng for Mixed Discrete and Continuous Parameters*, in: *IEEE International Symposium on Integrated Circuits*, pages 113–116, December 2009.

[PG11] M. Pehl, H. Graeb: *An SQP and Branch-and-Bound Based Approach for Discrete Sizing of Analog Circuits*, pages 297–316, InTech, February 2011.

[PKR$^+$00] R. Phelps, M. Krasnicki, R. A. Rutenbar, L. R. Carley, J. R. Hellums: *Anaconda: Simulation-Based Synthesis of Analog Circuits via Stochastic Pattern Search*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume 19(6), pages 703–717, June 2000.

[PMG08] M. Pehl, T. Massier, H. Graeb: *A Random and Pseudo-Gradient Approach for Analog Circuit Sizing with Non-Uniformly Discretized Parameters*, in:

*IEEE International Conference on Computer Design*, pages 188–193, October 2008.

[Pow78] M. Powell: *A Fast Algorithm for Nonlinearly Constrained Optimization Calculations*, in: G. Watson (ed.), *Numerical Analysis*, volume 630 of *Lecture Notes in Mathematics*, pages 144–157, Springer, 1978.

[PYT12] *Python Programming Language*, www.python.org, 2012.

[PZG10] M. Pehl, M. Zwerger, H. Graeb: *Sizing Analog Circuits Using an SQP and Branch and Bound Based Approach*, in: *IEEE International Conference on Electronics, Circuits, and Systems*, pages 37–40, December 2010.

[PZG11] M. Pehl, M. Zwerger, H. Graeb: *Variability-Aware Automated Sizing of Analog Circuits Considering Discrete Design Parameters*, in: *IEEE International Symposium on Integrated Circuits*, pages 98–101, December 2011.

[RGR07] R. A. Rutenbar, G. G. E. Gielen, J. Roychowdhury: *Hierarchical Modeling, Optimization, and Synthesis for System-Level Analog and RF Designs*, Proceedings of the IEEE, volume 95(3), pages 640–669, March 2007.

[Rut] R. A. Rutenbar: *Analog Layout Synthesis: What's Missing?*

[Sch02] R. Schwencker: *Zur Dimensionierung analoger integrierter Schaltungen unter Berücksichtigung struktureller Nebenbedingungen*, Ph.D. thesis, Technische Universität München, 2002.

[Sch04] F. C. Schenkel: *Tolerance Analysis and Design Centering of Analog Circuits, with Consideration of Mismatch*, Ph.D. thesis, Technische Universität München, 2004.

[SCP07] A. Somani, P. P. Chakrabarti, A. Patra: *An Evolutionary Algorithm-Based Approach to Automated Design of Analog and RF Circuits Using Adaptive Normalized Cost Functions*, IEEE Transactions on Evolutionary Computation, volume 11(3), pages 336–353, June 2007.

[SH06] B. Swahn, S. Hassoun: *Gate Sizing: FinFETs vs 32nm Bulk MOSFETs*, in: 43$^{rd}$ *ACM/IEEE Design Automation Conference*, pages 528–531, September 2006.

[SHA$^+$05] B. Swahn, S. Hassoun, S. Alam, D. Botha, A. Vidyarthi: *Thermal Analysis of FinFETs and its Application to Gate Sizing*, in: *ACM/IEEE International Workshop on Timing Issues*, March 2005.

[SKK03] M. J. Streeter, M. A. Keane, J. R. Koza: *Automatic Synthesis Using Genetic Programming of Both the Topology and Sizing for Five Post-2000 Patented Analog and Mixed Analog-Digital Circuits*, in: *IEEE Southwest Symposium on Mixed-Signal Design*, pages 5–10, February 2003.

[SMF02] H. Shibata, S. Mori, N. Fujii: *Automated Design of Analog Circuits Using Cell-Based Structure*, in: *NASA/DoD Conference on Evolvable Hardware*, pages 85–92, November 2002.

[SSGA00] R. Schwencker, F. Schenkel, H. Graeb, K. Antreich: *The Generalized Boundary Curve – A Common Method for Automatic Nominal Design and Design Centering of Analog Circuits*, in: *Design, Automation and Test in Europe*, pages 42–47, March 2000.

[SSPG02] R. Schwencker, F. Schenkel, M. Pronath, H. Graeb: *Analog Circuit Sizing using Adaptive Worst-Case Parameter Sets*, in: *DATE*, pages 581–585, March 2002.

[Str11] M. Strasser: *Deterministische hierarchische Platzierung analoger integrierter Schaltungen*, Ph.D. thesis, Technische Universität München, 2011.

[TCF96] A. Torralba, J. Chávez, L. G. Franquelo: *Circuit Performance Modeling by Means of Fuzzy Logic*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume 15(11), pages 1391–1398, November 1996.

[TDG09] N. Thakoor, V. Devarajan, J. Gao: *Computation Complexity of Branch-and-Bound Model Selection*, in: *IEEE International Conference on Computer Vision*, pages 1895–1900, October 2009.

[Tho94] M. E. Thomadakis: *Implementation and Evaluaton of Primal and Dual Simplex Methods with Different Pivot-Selection Techniques in the LPBench Environment*, Technical report, Texas A & M University, May 1994.

[TTR06] S. K. Tiwary, P. K. Tiwary, R. A. Rutenbar: *Generation of Yield-Aware Pareto Surfaces for Hierarchical Circuit Design Space Exploration*, in: *ACM/IEEE Design Automation Conference*, pages 31–36, September 2006.

[VSBM04] P. F. Vieira, L. B. Sa, J. P. B. Botelho, A. Mesquita: *Evolutionary Synthesis of Analog Circuits Using only MOS Transistors*, in: *NASA/DoD Conference on Evolvable Hardware*, pages 38–45, June 2004.

[Wan75] Y.-H. Wan: *On Local Pareto Optima*, Journal of Mathematical Economics, volume 2(1), pages 35–42, March 1975.

[WFG$^+$95] P. Wambacq, F. V. Fernandez, G. Gielen, W. Sansen, A. Rodriguez-Vazquez: *Efficient Symbolic Computation of Approximated Small-Signal Characteristics of Analog Integrated Circuits*, IEEE Journal of Solid-State Circuits, volume 30(3), pages 327–330, March 1995.

[WIC11] *WiCkeD*, www.muneda.com, 2011.

[YD09] E. Yilmaz, G. Dündar: *Analog Layout Generator for CMOS Circuits*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume 28(1), pages 32–45, January 2009.

[YKC$^+$05] W.-K. Yeh, C.-C. Ku, S.-M. Chen, Y.-K. Fang, C. P. Chao: *Effect of Extrinsic Impedance and Parasitic Capacitance on Figure of Merit of RF MOSFET*, IEEE Transactions on Electron Devices, volume 52(9), pages 2054–2060, September 2005.

[YM06] J. Yeung, H. Mahmoodi: *Robust Sense Amplifier Design under Random Dopant Fluctuations in Nano-Scale CMOS Technologies*, in: *IEEE International System-on-Chip Conference 2006*, pages 261–264, September 2006.

[YS96] Q. Yu, C. Sechen: *A Unified Approach to the Approximate Symbolic Analysis of Large Analog Integrated Circuits*, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, volume 43(8), pages 656–669, August 1996.

[YWLF95] H. Z. Yang, H. Wang, R. S. Liu, C. Z. Fan: *Simultaneous Topology Selection and Sizing for Synthesis of Analog Cells*, in: *IEEE Region 10 International Conference on Microelectronics and VLSI*, pages 159–162, November 1995.

[Ziz01] S. Zizala: *Numerische Verhaltensmodellierung analoger CMOS-Schaltungen unter Berücksichtigung von Dimensionierungsregeln*, Ph.D. thesis, Technische Universität München, 2001.

[ZJBS08] L. Zhang, N. Jangkrajarng, S. Bhattacharya, C.-J. R. Shi.: *Parasitic-Aware Optimization and Retargeting of Analog Layouts: A Symbolic-Template Approach*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume 27(5), pages 791–802, May 2008.

# List of Figures

# List of Tables

# List of Symbols

| | | |
|---|---|---|
| $N_p$ | Number of all parameters | 17 |
| $\mathbf{p}$ | Vector of all parameters | 17 |
| $\mathbf{d}$ | Vector of design parameters | 18 |
| $d_i$ | Component $i$ of $\mathbf{d}$ | 18 |
| $\mathbf{d}_c$ | Continuous design parameters | 18 |
| $\mathbf{d}_d$ | Discrete design parameters | 18 |
| $N$ | Number of design parameters | 18 |
| $\mathbb{D}^N$ | Parameter domain for all design parameters (design space) | 18 |
| $\mathbb{D}_c^{N_c}$ | Parameter domain for continuous design parameters | 18 |
| $\mathbb{D}_d^{N_d}$ | Parameter domain for discrete design parameters | 18 |
| $N_c$ | Number of continuous design parameters | 18 |
| $N_d$ | Number of discrete design parameters | 18 |
| $\mathbf{d}_L$ | Lower bound for design parameters | 19 |
| $\mathbf{d}_U$ | Upper bound for design parameters | 19 |
| $\mathbb{D}_{\text{rel}}^N$ | Relaxation of $\mathbb{D}^N$ | 19 |
| $N_o$ | Number of operating parameters | 21 |
| $\mathbf{o}$ | Vector of operating parameters | 21 |
| $\mathbf{o}_l$ | Lower bound for operating parameters | 21 |
| $\mathbf{o}_u$ | Upper bound for operating parameters | 21 |
| $\mathbb{T}_o^{N_o}$ | Tolerance domain for operating parameters | 21 |
| $\mathbf{C}$ | Variance-covariance matrix | 22 |
| $N_s$ | Number of process parameters | 22 |
| $\mathbf{s}$ | Vector of process parameters | 22 |
| $\mathbf{c}\,(\mathbf{p})$ | Vector of constraints at $\mathbf{p}$ | 23 |
| $\mathbf{f}\,(\mathbf{p})$ | Vector of performances at $\mathbf{p}$ | 25 |
| $\varepsilon_i$ | Performance-to-specification gap (component of $\boldsymbol{\varepsilon}(\mathbf{p})$) | 27 |
| $\mathbf{j}$ | Sensitivity vector | 27 |
| $\mathbf{J}_c$ | Jacobian matrix for constraints | 27 |
| $\mathbf{J}_f$ | Jacobian matrix for performances | 27 |
| $N_f$ | Number of performances | 27 |
| $\boldsymbol{\varepsilon}(\mathbf{p})$ | Vector of performance-to-specification gaps at $\mathbf{p}$ | 28 |
| $\mathbf{J}_\varepsilon$ | Jacobian matrix for performance-to-specification gaps | 28 |
| $N_\varepsilon$ | Number of performance-to-specification gaps | 28 |
| $\varphi\,(\mathbf{p})$ | Objective function at $\mathbf{p}$ | 28 |
| $k$ | Real valued minimization objective in min-max formulation | 34 |
| $\mathbb{A}$ | Acceptance region | 36 |