Technical University of Munich
TUM School of Computation, Information and Technology
Department of Computer Engineering
Chair of Electronic Design Automation

Mathematical Methods of Circuit Design

Textbook

Helmut Graeb

Formerly „Optimization Methods for Circuit Design“
Since WS 19/20 „Mathematical Methods of Circuit Design“
Since SS 2020 as textbook

Version 3.0 - 3.3	(SS 20 - SS 24)	Helmut Graeb
Version 2.11 - 2.13	(WS 15/16 - WS 18/19)	Maximilian Neuner
Version 2.8 - 2.11	(WS 12/13 - WS 15/16)	Michael Zwerger
Version 2.0 - 2.7	(WS 08/09 - SS 12)	Michael Eick
Version 1.0 - 1.2	(SS 07 - SS 08)	Husni Habal

Presentation follows:
H. Graeb, Analog Design Centering and Sizing, Springer, 2007.
R. Fletcher, Practical Methods of Optimization, John Wiley & Sons, 2nd Edition, 2000.

Status: May 14, 2024

Mathematical Methods of Circuit Design
Textbook
Technical University of Munich
Chair of Electronic Design Automation
Arcisstr. 21
80333 Munich, Germany

Contents

1 Introduction **1**

1.1 Parameters, performance, simulation 2

1.2 Performance specification 4

1.3 Design flow 5

1.4 Organization of the presentation 6

2 Some basics of optimization **9**

2.1 Maximization, minimization, minimum, unconstrained optimization 9

2.2 Constrained optimization 10

2.3 Classification of optimization problems 12

2.4 Multivariate Taylor series 14

2.5 Structure of an iterative optimization process 16

3 Optimality conditions **19**

3.1 Optimality conditions – unconstrained optimization 19

3.1.1 First-order condition for a local minimum of an unconstrained optimization problem 21

3.1.2 Second-order condition for a local minimum of an unconstrained optimization problem 21

3.2 Optimality conditions – constrained optimization 24

3.2.1 Feasible descent direction $\mathbf{r}$ 24

3.2.2 First-order conditions for a local minimum of a constrained optimization problem (Karush-Kuhn-Tucker (KKT) conditions) 27

3.2.3 Second-order condition for a local minimum of a constrained optimization problem 29

3.2.4 Sensitivity of the optimum objective value with regard to a change in an active constraint 30

4 Worst-case analysis **31**

4.1 Task 31

4.2 Typical tolerance regions 33

4.3 Linear performance model 34

4.4 Classical worst-case analysis 36

4.4.1 Classical worst-case analysis, lower worst case 37

4.4.2 Classical worst-case analysis, upper worst case 38

4.5 Realistic worst-case analysis 39

4.5.1 Realistic worst-case analysis, lower worst case 41

4.5.2 Realistic worst-case analysis, upper worst case 43

4.6 General worst-case analysis 44

4.6.1 General worst-case analysis, lower worst case, tolerance hyperellipsoid 45

4.6.2 General worst-case analysis, upper worst case, tolerance hyperellisoid 47

4.6.3 General worst-case analysis, tolerance hyperrectangle 47

4.7 Input/output of worst-case analysis 48

5 Unconstrained optimization **51**

5.1 Line search 51

5.2 Golden sectioning 55

5.3 Line search by quadratic model 57

5.4 Backtracking line search 58

5.5 Coordinate search for multivariate unconstrained optimization without derivatives 59

5.6 Polytope method (Nelder-Mead simplex method) for multivariate unconstrained optimization without derivatives 61

5.7 Newton approach for multivariate unconstrained optimization with derivatives 64

5.8 Quasi-Newton approach for multivariate unconstrained optimization with derivatives 66

5.9 Levenberg-Marquardt approach (Newton direction plus trust region) for multivariate unconstrained optimization with derivatives 68

5.10 Least-squares (plus trust-region) approach for multivariate unconstrained optimization with derivatives 69

5.11 Conjugate-gradient (CG) approach for multivariate unconstrained optimization with derivatives 73

6 Constrained optimization **79**

6.1 Quadratic Programming (QP) – linear equality constraints 79

6.2 Quadratic Programming – inequality constraints 83

6.3 Sequential Quadratic Programming (SQP), Lagrange–Newton 89

7 Statistical parameter tolerances **93**

7.1 Univariate Gaussian distribution (univariate normal distribution) 95

7.2 Multivariate Gaussian distribution (multivariate normal distribution) . . . 97

7.3 Transformation of statistical distributions 100

8 Expectation values and their estimation **105**

8.1 Expectation values . . . 105

8.1.1 Definitions . . . 105

8.1.2 Expectation value of linear transformation . . . 107

8.1.3 Variance of linear transformation . . . 107

8.1.4 Translation law of variances . . . 108

8.1.5 Normalizing a random variable . . . 108

8.1.6 Linear transformation of a normal distribution . . . 109

8.2 Estimation of expectation values . . . 110

8.2.1 Variance of the expectation value estimator . . . 112

8.2.2 Estimated expectation value of linear transformation . . . 113

8.2.3 Estimated variance of linear transformation . . . 113

8.2.4 Translation law of estimated variance . . . 113

9 Yield analysis **115**

9.1 Problem formulation . . . 115

9.2 Statistical yield estimation/Monte-Carlo analysis . . . 116

9.3 Accuracy of statistical yield estimation . . . 118

9.4 Geometric yield analysis for linearized performance feature . . . 122

9.4.1 Upper performance bound $\overline{\varphi} \leq \varphi_U$ . . . 126

9.5 General geometric yield analysis . . . 127

9.6 Accuracy of geometric yield approximation . . . 129

9.7 Overall yield from geometric yield analysis . . . 131

9.8 Consideration of range parameters . . . 132

9.9 Another interpretation of the worst-case distance and preparing its gradient 134

10 Yield optimization/design centering/nominal design 137

10.1 Optimization objectives . . . 137

10.2 Derivatives of optimization objectives . . . 140

10.3 Problem formulations of analog optimization . . . 142

10.4 Input/output of yield optimization with worst-case distances . . . 143

11 Sizing rules for analog circuit optimization 145

11.1 Library of NMOS transistor groups . . . 146

11.2 Single (NMOS) transistor . . . 148

11.2.1 Sizing rules for a single transistor that acts as a voltage-controlled current source (vccs) . . . 149

11.2.2 Sizing rules for a single transistor that acts as a voltage-controlled resistor (vcres) . . . 149

11.3 Transistor pairs (NMOS) . . . 150

11.3.1 Simple current mirror . . . 150

11.3.2 Level shifter . . . 150

11.3.3 Differential pair . . . 151

11.3.4 Cross-coupled pair . . . 151

11.4 Transistor pair groups . . . 151

11.4.1 Cascode current mirror . . . 151

11.4.2 4-Transistor current mirror . . . 151

11.4.3 Wide swing cascode current mirror . . . 152

11.4.4 Wilson current mirror . . . 152

11.4.5 Improved Wilson current mirror . . . 152

A Matrix and vector notations **153**

A.1 Vector . . . 153

A.2 Matrix . . . 153

A.3 Addition . . . 154

A.4 Multiplication . . . 154

A.5 Special cases . . . 155

A.6 Determinant of a quadratic matrix . . . 156

A.7 Inverse of a quadratic non-singular matrix . . . 156

A.8 Some properties . . . 157

B Notations and abbreviations for first- and second-order derivatives **159**

C Partial derivatives of linear, quadratic terms in matrix/vector notation **161**

D Norms **163**

E Pseudo-inverse, singular value decomposition (SVD) **165**

E.1 Moore-Penrose conditions . . . 165

E.2 Singular value decomposition . . . 166

F Linear equation system, rectangular system matrix with full rank **167**

F.1 underdetermined system of equations . . . 167

F.2 overdetermined system of equations . . . 168

F.3 determined system of equations . . . 169

G Probability space **171**

H Convexity **173**

H.1 Convex set $K \in \mathbb{R}^n$ . . . 173

H.2 Convex function . . . 173

I Derivatives of the statistically estimated yield **175**

1 Introduction

The presentation in this book is based on matrix and vector notations. Readers must check their familiarity with these notations. Please read the Appendices A, B, C and D to revise and refresh your memories of this part of linear algebra before starting to read on in the book.

Appendix A gives definitions of vector, matrix and its row and column vectors, transposed matrix, addition and multiplication of matrices, special cases like identity matrix and diagonal matrix, determinant and inverse of a matrix, and several properties of matrices.

Appendix B presents notations for the first-order derivative (gradient) and second-order derivative (Hessian matrix) of a scalar function with regard to a vector of parameters and a vector of a subset of parameters. The abbreviated notations detailed in this Appendix are frequently used throughout the book.

We abbreviate with $\nabla f\left(\mathbf{x}_s'\right)$ the first-order derivative of a function f with regard to a vector $\mathbf{x_s}$ of a subset of the overall parameters $\mathbf{x}$ at a specific point $\mathbf{x}_s'$ of this subset, while the rest of parameters are kept constant.

Similarly, we abbreviate with $\nabla^2 f\left(\mathbf{x}_d', \mathbf{x}_s'\right)$ the second-order derivative of a function f with regard to two vectors $\mathbf{x_d}$, $\mathbf{x_s}$ of a subset of the overall parameters $\mathbf{x}$ at a specific point $\mathbf{x}_d'$, $\mathbf{x}_s'$ of this subset, while the rest of the parameters are kept constant. Please familiarize yourself with these notations.

Appendix C gives the basic formulas for the derivatives in a function model's linear and quadratic terms in matrix/vector notation. These are frequently used throughout the book. Familiarization with it is essential as well. The proofs are not given; they can be obtained by moving from the matrix/vector notation to a component-wise notation, conducting the symbolic derivation, and moving the result back to a matrix/vector notation.

Appendix D lists some vector norms and matrix norms. They will be used to scalarize a vector optimization problem, i.e., to combine several optimization objectives into a single objective.

1.1 Parameters, performance, simulation

The methods presented in this book have been developed for and applied to analog integrated circuits. They represent all kinds of technical problems where a single mathematical function evaluation, e.g., by numerical simulation, is expensive and in the range of minutes to an hour. As numerical optimization is frequently called function evaluation, the optimization cost then sums up to hours or even days. When statistical estimations come into play, the CPU cost is even worse.

We do not rely on or include modeling methods such as, e.g., response surface modeling to reduce the CPU cost by approximating the function evaluation. According to our experience, the simulation cost to establish such models is often better used for an optimization process that works on function evaluation by direct simulation. Of course, any type of function evaluation can be used in optimization.

But the main idea is that function evaluation is expensive and requires specific optimization approaches. This holds for many technical systems, as does the characterization of parameters and performance in Table 1.

We distinguish between parameters and performance in this book. Parameters are problem variables *input* to the simulation/function evaluation. Performance features are problem variables *output* of the simulation/function evaluation.

Three types of parameters are distinguished.

- Design parameters are subject to sizing by the design engineer. In analog integrated circuit design, transistor widths, lengths, and capacitance values are typical design parameters.
- Statistical parameters are subject to manufacturing variations. They carry characteristics of statistical distributions, mostly mean values, variances, and covariances, and a type of statistical distribution, e.g., a multivariate normal distribution. In analog integrated circuit design, a subset of transistor model parameters typically form the statistical parameters, e.g., threshold voltage, oxide thickness, and carrier mobility of transistors.
- Range parameters are also subject to variation. However, only a range, mostly an interval, of the potential variability is given, but not a probability of occurrence or density, respectively. These are often parameters describing an operating range in which the system has to be functional. The supply voltage and temperature are typical range parameters in analog integrated circuit design. If aging is to be considered and an aging model is included in the simulation model, a specific lifetime will be another range parameter.

design parameters	$\mathbf{x}_d \in R^{n_{xd}}$	e.g. transistor widths, capacitances
statistical parameters	$\mathbf{x}_s \in R^{n_{xs}}$	e.g. oxide thickness, threshold voltage
range parameters	$\mathbf{x}_r \in R^{n_{xr}}$	e.g. supply voltage, temperature
(circuit) parameters	$\mathbf{x} = \left[\mathbf{x}_d^T\, \mathbf{x}_s^T\, \mathbf{x}_r^T\right]^T$	
performance feature	φ_i	e.g. gain, bandwidth, delay, power
(circuit) performance	$\boldsymbol{\varphi} = [\cdots \varphi_i \cdots]^T \in R^{n_\varphi}$	
(circuit) simulation	$\mathbf{x} \longmapsto \boldsymbol{\varphi}(\mathbf{x})$	e.g. SPICE

Table 1: Simulation maps different types of parameters to the performance features.

In integrated circuit design, the *p*rocess, *t*emperature, and supply *v*oltage parameters are often denoted as the PVT parameters that are subject to variation. In integrated circuit design, the transistor parameters, while carrying the information of the manufacturing variation, are not subject to design.

Please note that a design parameter and a statistical parameter may be identical or refer to the same physical parameter. An example is the width of a transistor, which is subject to variation due to the manufacturing process. As this variation is often global and identical for all transistors, a statistical parameter, e.g., width reduction ΔW, is forked with zero mean and a given variance. The nominal values of the individual transistor widths W_k are then the design parameters. When calling a simulation, the value of the one statistical parameter is added to the individual design parameter values $W_k + \Delta W$ to obtain the physical width parameter. This proceeding has the advantage of just one statistical parameter, which reduces the complexity of the statistical circuit design.

The transistor threshold voltage is another example of partitioning physical parameters in the design and statistical parts. In addition to a global statistical variation ΔV_{th}, there are local statistical variations $\Delta V_{th,k}$ of each individual transistor k. The physical threshold voltage that is sent to the simulator, therefore, is the sum of the two statistical and the nominal part: $V_{th0} + \Delta V_{th} + \Delta V_{th,k}$. Local statistical variations are often modeled as statistically independent, from which it follows that their correlation is zero.

The statistical parameter modeling will be detailed in chapters 7 and 8.

A variety of individual features describes the performance of a circuit/system. More than a dozen performance features are relevant for an operational amplifier, a fundamental analog circuit. Among these are, for instance, gain, bandwidth, slew rate, phase margin, delay, power, common mode rejection ratio, power supply rejection ratio, and offset. These performance features must be within the specified bounds defined in the next section.

Simulation represents THE abstraction from the physical level in circuit-level integrated circuit design.

Please note that the setup of simulation requires not only the circuit/system itself, e.g., in the form of a circuit netlist in integrated circuit design, but a simulation bench including input stimuli, information about the manufacturing process, i.e., the so-called process development kit (pdk) in integrated circuit design, and other data.

Please note that the optimization process can start once an automatic simulation has been set up. Automatic simulation means that given values of the parameters, the simulation results are values of the performance features. Simulators often provide waveforms of circuit variables like voltages and currents over time (transient simulation) or frequency (AC simulation). The extraction of performance values from these waveforms must be provided as part of an automatic simulation setup.

And please note that simulation means a pointwise evaluation of the mapping $\mathbf{x} \longmapsto \boldsymbol{\varphi}(\mathbf{x})$. Usually, there is no closed-form analytical function available for this mapping.

1.2 Performance specification

For each performance feature, an upper limit, a lower limit, or both an upper and lower limit may be specified. Each bound is denoted as a performance specification feature:

$$\varphi_i \geq \varphi_{L,i} \vee \varphi_i \leq \varphi_{U,i} \tag{1}$$

The lower (upper, respectively) bound for the number of performance specification features n_{PSF} results if either an upper or a lower bound (or both, respectively) is specified for each performance feature (with n_φ as the number of performance features):

$$n_\varphi \leq n_{PSF} \leq 2n_\varphi \tag{2}$$

We define a vector notation for the combination of all performance specification features as follows:

$$\left\{ \begin{array}{c} \varphi_{L,1} \leq \varphi_1(\mathbf{x}) \leq \varphi_{U,1} \\ \vdots \\ \varphi_{L,n_\varphi} \leq \varphi_{n_\varphi}(\mathbf{x}) \leq \varphi_{U,n_\varphi} \end{array} \right\} \iff \boldsymbol{\varphi}_L \leq \boldsymbol{\varphi}(\mathbf{x}) \leq \boldsymbol{\varphi}_U \tag{3}$$

1.3 Design flow

The optimization of the parameters of a system, also called sizing, starts after the system's structure has been designed. In circuit design, the structure is the specific connection of a specific set of transistors.

Once the system is sized, the manufacturing plan is determined. In integrated circuit design, this is the layout.

This makes three main design steps: structural synthesis, parametric synthesis (i.e., sizing), and layout synthesis.

The sizing is partitioned into two main steps. These are the nominal design and the tolerance design.

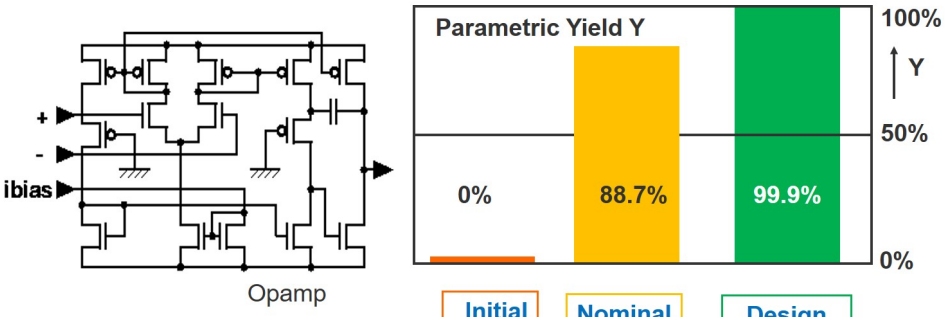
In the nominal design step, the nominal performance values are optimized by tuning the design parameters while keeping the statistical and range parameters at fixed values. These fixed values may be nominal parameter values or worst or corner values. Chap. 4 will explain how such worst or corner values can be determined. The nominal design step can further be divided into substeps by partitioning the set of performance features and the set of circuit/system constraints into ordered subsets, which are optimized one after the other. In each subsequent step, the optimized performance features from all prior steps can move slightly but stay within their bounds.

In the tolerance design step, not the nominal performance values are optimized, but the robustness and the yield (i.e., percentage of circuits fulfilling the specification) under consideration of the statistics of the manufacturing process and the operating range. Chap. 9 will explain how yield can be analyzed. We will distinguish between a statistical and a geometric approach. In literature, geometric approaches for tolerance design are often called design centering. We will use the terms yield optimization and design centering as synonyms in this book.

The partitioning into several nominal design steps and the tolerance design step is done to save simulation costs. Having smaller sets of optimization objectives eases the optimization process. It also mimics the practical proceeding of design engineers. Tolerance design has to include statistical parameters and more complex analysis types, as we will see in the course of the book. A good nominal design provides a good starting point for the more CPU-intensive tolerance design, leading to faster progress toward a solution.

Fig. 1 illustrates the nominal and tolerance design steps for an operational amplifier, which is sketched at the top left of the figure.

In the table at the bottom of the figure, five performance features are given with their respective specification bounds. These are the AC features gain, transit frequency, phase margin, transient feature slew rate, and power consumption, approximated as a DC feature. Do not worry about the technology node; this example can hold for any. It can be seen that all performance features but the phase margin violate their specification initially. This may be the initial situation after moving to another process node, which is usually beneficial for the digital system part, but may jumble up the analog part. After the nominal design, all specifications are fulfilled. A yield analysis shows that the yield is roughly 90% after nominal design; in other words, 90% of produced opamps will fulfill



Performance	Specification	Initial values	Nominal design	Design centering
Gain	≥ 80dB	67dB	100dB	100dB
Transit frequency	≥ 10MHz	5MHz	20MHz	18MHz
Phase margin	≥ 60°	75°	68°	72°
Slew rate	≥ 10V/µs	4.1V/µs	12V/µs	12V/µs
DC power	≤ 50µW	122µW	38µW	39µW

Figure 1. Nominal design and design centering of a circuit example.

the given specification values. Looking at the individual performance safety margins does not allow us to identify which one is the weak element in the chain. Design centering, as presented in Chap. 9.4, therefore looks at what will be called worst-case distances. After design centering, the yield is 99.9%. We can see that the performance of the transit frequency has been slightly relaxed compared to the nominal design, and the phase margin has been slightly improved to maximize the yield.

1.4 Organization of the presentation

The remainder of the book is organized around three main tasks of tolerance design. These tolerance design tasks are worst-case analysis, yield analysis, and yield optimization. First, the necessary mathematical preliminaries to formulate the respective tolerance design task are described. Right afterward, the respective tolerance design task's problem formulations and solution approaches are presented. The resulting presentation sequence is as follows.

In the following chapter 2, some optimization basics will be sketched. Optimization problems will be classified, and the ones treated in this book will be identified. The mathematical formulation of an unconstrained and a constrained optimization problem will be provided. As many of the later derivations will be based on a Taylor series of optimization variables, the multivariate formulation of a Taylor series will be given. And the basic structure of an iterative optimization process will be sketched. In the subsequent chapter 3, the first-and second-order optimality conditions for an unconstrained optimization problem and a constrained optimization problem will be introduced.

We can formulate the tolerance design task of worst-case analysis using optimality conditions. This is done in the next chapter 4. Three different formulations will be presented. These are classical, realistic, and general, worst-case analyses. It will turn out that analytical formulas for the results of classical and realistic worst-case analysis can be derived.

The general worst-case analysis remains a specific type of nonlinear optimization problem, which has to be solved iteratively by numerical methods. *This chapter presents one of the two core messages of this book. It explains why the computation of worst-case parameter sets requires design technology and manufacturing technology, in other words, why the computation of worst-case parameter sets is part of the circuit/system design as it depends on the specific design at hand. Worst-case parameter sets determined upfront without considering the specific design problem are approximations to the true worst-case of unknown accuracy.*

To solve the nonlinear optimization problem of general worst-case analysis, the following two chapters will present the basics of nonlinear optimization in a nutshell. Chap. 5 will deal with unconstrained optimization, and Chap. 6 will deal with constrained optimization. We will reach the problem formulation of Sequential Quadratic Programming. These two chapters follow the presentation of Fletcher's "Practical Methods of Optimization", published by Wiley. Many other books on optimization exist, but the author of the book at hand is a fan of Fletcher's presentation.

The methods described for nonlinear optimization are applicable to the technical circuit/system optimization tasks of nominal design and design centering. These will be treated later on. Before that, the technical problem of yield analysis has to be treated to obtain the design centering's optimization objective. As this brings statistics into the play, we first introduce the basics of multivariate statistical parameter tolerances in Chap. 7, and multivariate formulations of expectation values and their estimation in Chap. 8.

Based on that, Chap. 9 will formulate yield and present two methods for its statistical estimation and geometric approximation, respectively. We will discover that worst-case and yield analyses can be formulated by nonlinear optimization problems closely related through an exchange of constraint and objective. *This chapter presents the second core message of this book. It advertises replacing statistical estimation techniques with a deterministic approach. This deterministic approach defines a characteristic parameter set of a stochastic variable, which is the point of highest probability (probability density, to be precise) to fail a specification value. It will be shown that this characteristic point not only allows an accurate approximation of yield but is more suitable for robustness characterization in the case of high-sigma problems. This characteristic point is a suitable deterministic approach in problems of rare event statistics.*

Based on the worst-case analysis and yield analysis, Chap. 10 deals with formulations of the optimization tasks of circuits/systems. Different optimization objectives of nominal and tolerance design will be given, and first- and second-order derivatives of these objectives will be derived. These objectives and their derivatives can be applied with the optimization methods in chapters 5 and 6 to create application-specific solution approaches to circuit/system sizing.

A design problem is often incompletely specified. This makes automatic optimization impossible, as the optimizer does not have the information to find a technically reasonable solution. Interactive optimization is the method of choice in that case. Detailed diagnosis based on sensitivity analysis and local performance models allows the user to steer the optimization process. For analog circuits, inherent properties can be automatically extracted from the netlist, which complements the input from the user and enables

an automatic optimization process. The presentation of these so-called sizing rules will complete this book in Chap. 11.

So stay on and read on!

2 Some basics of optimization

We first look at some properties of optimization.

2.1 Maximization, minimization, minimum, unconstrained optimization

Optimization aims at either maximizing or minimizing the value of an objective f. A maximization problem can be reformulated as a minimization problem without loss of generality. We obtain a maximization of the objective, for instance, by adding a minus sign to the objective, then minimizing this, and afterward removing the minus sign:

$$\max f \equiv -\min -f \tag{4}$$

Therefore, a formulation as a minimization problem is sufficient; a maximization problem does not have to be formulated separately.

In this book, we will primarily use the formulation of an optimization problem as a minimization problem.

Usually, the abbreviation "min" will be used. Optimization is usually an iterative process toward the optimum so that we can distinguish between the process and the result. Both can be meant with the abbreviation min:

$$\min \equiv \begin{cases} \text{minimum}, & \text{i.e., result} \\ \text{minimize}, & \text{i.e., process} \end{cases} \tag{5}$$

The following two formulations are used for minimizing a single optimization objective f over an optimization variable vector $\mathbf{x}$ in the absence of any constraints:

$$\min f(\mathbf{x}) \quad \equiv \quad f(\mathbf{x}) \overset{!}{\rightarrow} \min \tag{6}$$

The resulting values of unconstrained optimization are often denoted with a $*$. The result comprises the optimal variable values $\mathbf{x}^*$ and the optimal objective value f^*:

$$\min f(\mathbf{x}) \quad \longrightarrow \quad \mathbf{x}^*,\ f(\mathbf{x}^*) = f^* \tag{7}$$

Various optimization objectives f exist in circuit/system design (see Chap. 10). We can imagine that the circuit/system performance features $\boldsymbol{\varphi}$ are the optimization objectives.

The unconstrained optimization problem is formulated in different flavors, e.g.:

$$\begin{array}{rcccccccc} f^* & = & \min f(\mathbf{x}) & \equiv & \min\limits_{\mathbf{x}} f & \equiv & \min\limits_{\mathbf{x}} f(\mathbf{x}) & \equiv & \min\{f(\mathbf{x})\} \\ \mathbf{x}^* & = & \operatorname{argmin} f(\mathbf{x}) & \equiv & \operatorname{argmin}\limits_{\mathbf{x}} f & \equiv & \operatorname{argmin}\limits_{\mathbf{x}} f(\mathbf{x}) & \equiv & \operatorname{argmin}\{f(\mathbf{x})\} \end{array} \tag{8}$$

The second row is used when the optimization variable (the argument) shall be described rather than the optimization objective.

2.2 Constrained optimization

Reality brings limited resources. This leads to the problem of constrained optimization, which is formulated as optimization of the optimization objective f over the optimization variable vector $\mathbf{x}$ subject to equality and inequality constraints:

$$\boxed{\begin{aligned} \min f(\mathbf{x}) \text{ s.t.} \quad & c_i(\mathbf{x}) = 0 \ , i \in E \\ & c_i(\mathbf{x}) \geq 0 \ , i \in I \\ & E \cap I = \emptyset \end{aligned}} \tag{9}$$

f: optimization objective $\quad$ $\mathbf{x}$: optimization variable $\quad$ c_i: constraint

E: set of equality constraints $\quad$ I: set of inequality constraints $\quad$ s.t.: subject to

This formulation includes constraints with a "$\leq$" condition, $c_j(\mathbf{x}) \leq 0$, which are transformed by multiplication with -1 into the form of (9): $c_i(\mathbf{x}) = -c_j(\mathbf{x}) \geq 0$.

The constrained optimization problem can also be formulated in different flavors based on the feasibility region Ω, which describes the region of optimization variables that fulfill the equality and inequality constraints:

$$\min_{\mathbf{x}} f \text{ s.t. } \mathbf{x} \in \Omega \tag{10}$$

$$\min_{\mathbf{x} \in \Omega} f \tag{11}$$

$$\min \{ f(\mathbf{x}) \mid \mathbf{x} \in \Omega \} \tag{12}$$

with,

$$\Omega = \left\{ \mathbf{x} \;\middle|\; \begin{array}{l} c_i(\mathbf{x}) = 0, \ i \in E \\ c_i(\mathbf{x}) \geq 0, \ i \in I \end{array} \right\}$$

Lagrange function

The so-called Lagrange function combines objective and constraints in a single expression in the following way:

$$\boxed{\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{i \in E \cup I} \lambda_i \cdot c_i(\mathbf{x})} \tag{13}$$

λ_i is the so-called Lagrange multiplier associated with constraint i. The purpose of the Lagrange function is to transform the constrained optimization problem into an unconstrained optimization problem of $\mathcal{L}$, whose solution is precisely at the solution of the constrained problem. We can consider the constrained optimization problem as an unconstrained one with the Lagrange function as the objective function and the Lagrange multipliers as additional optimization variables.

First, we inspect the Lagrange function for the case that there is no competition between the objective and inequality constraints. Minimizing the objective will not drive the constraints toward zero or even away, assuming the Lagrange multiplier is non-negative.

The equality constraints are zero, hence, they do not influence the value of the Lagrange function. If there is competition between the objective and an inequality constraint, then decreasing the objective will drive the constraint toward its bound zero.

Take the example of an objective function $f(x) = x^2$, which has its minimum at $\operatorname{argmin} x^2 = 0$. Adding the constraint $c(x) = x - 1$ with $c(x) \geq 0$ only allows x to go down to the value of 1, which is the minimum of this constrained optimization example. The corresponding Lagrange function is $\mathcal{L}(x, \lambda) = x^2 - \lambda \cdot (x - 1)$. It is illustrated in Fig. 2.

The purpose of the Lagrange multiplier is to create a minimum of the Lagrange function exactly where the solution of the constrained problem is. In this example, $\lambda^* = 2$ and the corresponding Lagrange function, $\mathcal{L}(x, 2) = x^2 - 2(x - 1)$, has a minimum at $x^* = 1$.

Fig. 2 illustrates another property of the Lagrange function, which is that the solution is a saddle point of $\mathcal{L}(x, \lambda)$. Around the solution, the Lagrange function has a minimum with regard to the variable and a maximum with regard to the Lagrange multiplier.

$$\mathcal{L}(x^*, \lambda) \leq \mathcal{L}(x^*, \lambda^*) \leq \mathcal{L}(x, \lambda^*) \tag{14}$$

This property is related to the so-called duality principle in optimization, which will not be deepened here.

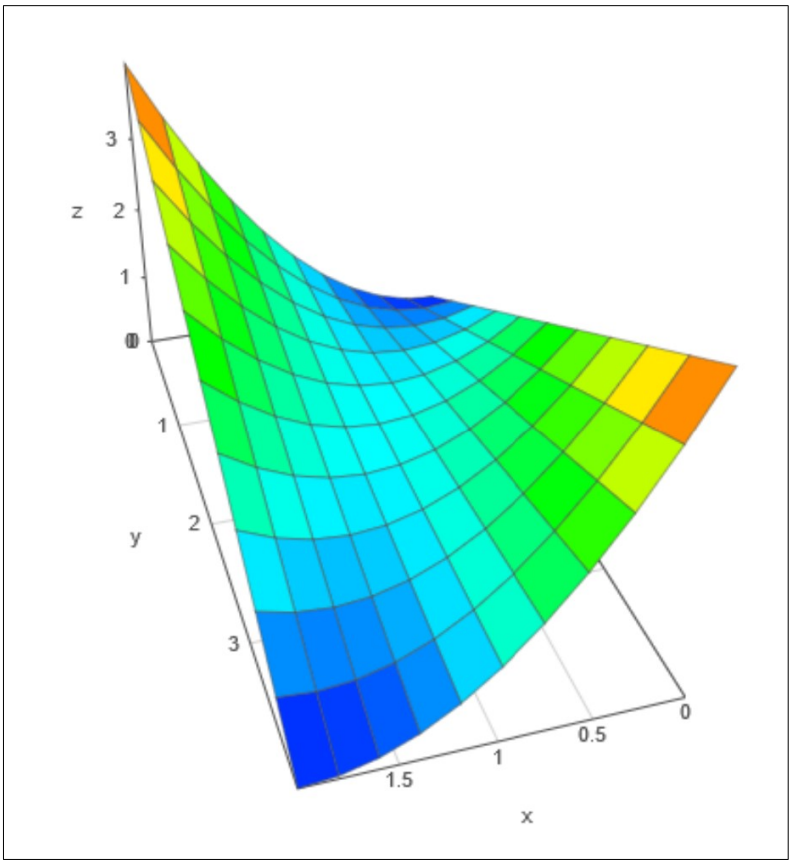


Figure 2. Lagrange function $\mathcal{L}(x, \lambda) = x^2 - \lambda \cdot (x - 1)$ of the example, $y = \lambda$, $z = \mathcal{L}$.

2.3 Classification of optimization problems

There are several types of optimization problems. We will mention some of them here and narrow them down to those that will be the subject of this book. In addition, some basic assumptions will be made.

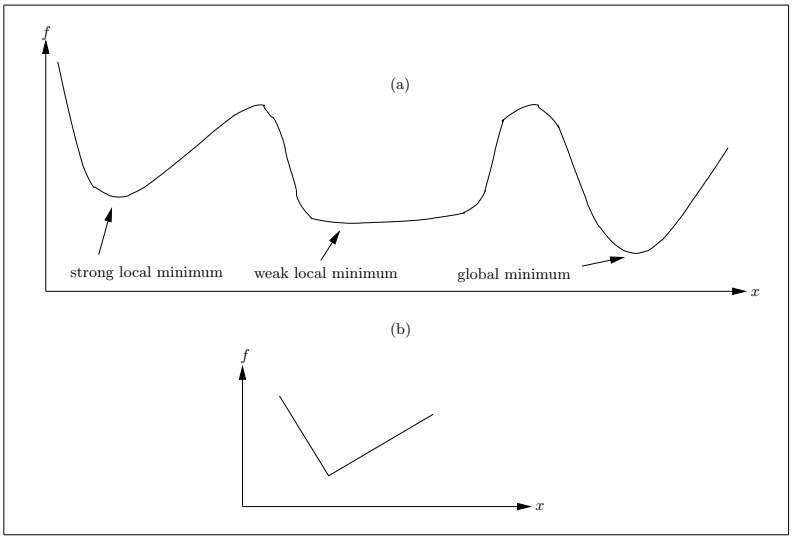


Figure 3. Smooth function (a), i.e., continuous and differentiable at least several times on a closed domain region. Non-smooth continuous function (b).

The first assumption we make is that the functions are smooth, as illustrated in Fig. 3. The second assumption is that we deal with local optimization, i.e., finding the nearest optimum within a specific range of values. Depending on the starting point, local optimization would discover one of the three minima in the example in Fig. 3. Global optimization can build on local optimization and add methods for determining regions where local optimization is performed.

As summarized in the following, the classification of optimization problems can be done according to the type of functions. We will deal with solution approaches to quadratic and nonlinear programming.

objective		constraints	
linear	and	linear	linear programming
quadratic	and	linear	quadratic programming
nonlinear	or	nonlinear	nonlinear programming
convex		linear equality, concave inequality	convex programming (local $\equiv$ global minimum)

More classification criteria are summarized in the following.

deterministic, stochastic	The iterative search process is deterministic or random.
continuous, discrete	Optimization variables can take an infinite number of values, e.g., a set of real numbers or a finite set of values or states.
local, global	The objective value at a local optimal point is better than the objective values of other points in its vicinity. The objective value at a global optimal point is better than the objective values of any other point.
scalar, vector	In a vector optimization problem, multiple objective functions shall be optimized simultaneously (multiple-criteria optimization, MCO). Usually, objectives have to be traded off with each other. A Pareto-optimal point is characterized in that one objective can only be improved at the cost of another. Pareto optimization determines the set of all Pareto-optimal points. Scalar optimization refers to a single objective. A vector optimization problem is scalarized by combining multiple objectives into a single overall objective, e.g., by a weighted sum, least-squares, or min/max.
constrained, unconstrained	Besides the objective function that has to be optimized, constraints on the optimization variables may be given as inequalities or equalities.
with, without derivatives	The optimization process may be based on gradients (first derivative) or gradients and Hessians (second derivative), or it may not require any derivatives of the objective/constraint functions.

We will deal with deterministic solution approaches for scalar and continuous optimization problems. Basic approaches will be presented to combine the different objectives of a vector optimization problem in a scalar cost function. This is necessary, as we will initially formulate several circuit/system optimization problems as vector optimization problems. We will treat both constrained and unconstrained optimization, and solution approaches with or without derivatives.

2.4 Multivariate Taylor series

In optimization, many properties are derived using a Taylor series of the objective or the constraints with regard to the optimization variables up to the second-order term. That means that a quadratic model is considered, but higher-order terms of the Taylor series are not considered. We will formulate a Taylor series of the objective f in the following. The formulas hold analogously for any specific objective or constraint. If there is only one variable, the Tayor series at x_0 up to the second term is:

$$f(x) = f(x_0) + g_0 \cdot (x - x_0) + \frac{1}{2} h_0 \cdot (x - x_0)^2 + \dots \tag{15}$$

g_0 and h_0 are the first and second derivative of f with regard to x at x_0. If there are two variables x_1, x_2 , the Taylor series at $x_{0,1}$, $x_{0,2}$ is:

$$\begin{aligned} f(x_1, x_2) = f(x_{0,1}, x_{0,2}) \quad &+ \quad g_{0,1} \cdot (x_1 - x_{0,1}) + g_{0,2} \cdot (x_2 - x_{0,2}) \\ &+ \quad \frac{1}{2} h_{0,11} \cdot (x_1 - x_{0,1})^2 \\ &+ \quad h_{0,12} \cdot (x_1 - x_{0,1}) \cdot (x_2 - x_{0,2}) + \frac{1}{2} h_{0,22} \cdot (x_2 - x_{0,2})^2 + \dots \end{aligned} \tag{16}$$

$g_{0,i}$ are the first partial derivatives of f with regard to both variables x_i at $x_{0,1}$, $x_{0,2}$. $h_{0,ij}$ are the corresponding second partial derivatives.

If there are n_x variables, the Taylor series is:

$$f(\mathbf{x}) \quad = \quad f_0 + \sum_{i=1}^{n_x} g_{0,i} \cdot (x_i - x_{0,i}) + \frac{1}{2} \sum_{i=1}^{n_x} \sum_{j=1}^{n_x} h_{0,ij} \cdot (x_i - x_{0,i}) \cdot (x_j - x_{0,j}) + \dots \tag{17}$$

This component-wise formulation can be moved to a compact matrix-vector formulation (see Appendix A for formulas to derive it):

$$f(\mathbf{x}) \quad = \quad f(\mathbf{x}_0) + \mathbf{g}_0^T \cdot (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_0)^T \cdot \mathbf{H}_0 \cdot (\mathbf{x} - \mathbf{x}_0) + \dots \tag{18}$$

Please note that $\mathbf{H}_0$ is a symmetric matrix, as the order of the partial derivation with regard to two different parameters is commutative.

Optimization is an iterative process. An intermediate value of the optimization variables is denoted as iteration point $\mathbf{x}^{(\kappa)}$. The Taylor series of a function f at **iteration point** $\mathbf{x}^{(\kappa)}$ is:

$$\begin{aligned} f(\mathbf{x}) = f(\mathbf{x}^{(\kappa)}) \quad &+ \quad \nabla f\left(\mathbf{x}^{(\kappa)}\right)^T \cdot \left(\mathbf{x} - \mathbf{x}^{(\kappa)}\right) \\ &+ \quad \frac{1}{2} \left(\mathbf{x} - \mathbf{x}^{(\kappa)}\right)^T \cdot \nabla^2 f\left(\mathbf{x}^{(\kappa)}\right) \cdot \left(\mathbf{x} - \mathbf{x}^{(\kappa)}\right) \quad + \quad \dots \end{aligned} \tag{19}$$

$$= f^{(\kappa)} \quad + \quad \mathbf{g}^{(\kappa)T} \cdot \left(\mathbf{x} - \mathbf{x}^{(\kappa)}\right) + \frac{1}{2} \left(\mathbf{x} - \mathbf{x}^{(\kappa)}\right)^T \cdot \mathbf{H}^{(\kappa)} \cdot \left(\mathbf{x} - \mathbf{x}^{(\kappa)}\right) \quad + \quad \dots \tag{20}$$

$f^{(\kappa)}$	:	value of function f at point $\mathbf{x}^{(\kappa)}$
$\mathbf{g}^{(\kappa)}$	:	gradient (first derivative, direction of steepest ascent) at point $\mathbf{x}^{(\kappa)}$
$\mathbf{H}^{(\kappa)}$	:	Hessian matrix (second derivative) at point, symmetric $\mathbf{x}^{(\kappa)}$

See Appendix B for defining the gradient and Hessian matrix components.

Iterative optimization distinguishes between a search direction $\mathbf{r}$ starting at the current iteration point $\mathbf{x}^{(\kappa)}$ and a step length α of the search direction $\mathbf{r}^{(\kappa)}$ starting at the current iteration point $\mathbf{x}^{(\kappa)}$. The Taylor series (19), (20) can be reformulated to represent search direction and step length.

The Taylor series with regard to a **search direction r** starting at the current iteration point $\mathbf{x}^{(\kappa)}$ is:

$$\mathbf{x}(\mathbf{r}) = \mathbf{x}^{(\kappa)} + \mathbf{r} \tag{21}$$

$$f(\mathbf{x}^{(\kappa)} + \mathbf{r}) \equiv f(\mathbf{r}) = f^{(\kappa)} + \mathbf{g}^{(\kappa)T} \cdot \mathbf{r} + \frac{1}{2}\mathbf{r}^T \cdot \mathbf{H}^{(\kappa)} \cdot \mathbf{r} + \ldots \tag{22}$$

The Taylor series with regard to the **step length** of the search direction $\mathbf{r}^{(\kappa)}$ starting at current iteration point $\mathbf{x}^{(\kappa)}$ is:

$$\mathbf{x}(\alpha) = \mathbf{x}^{(\kappa)} + \alpha \cdot \mathbf{r}^{(\kappa)} \tag{23}$$

$$f(\mathbf{x}^{(\kappa)} + \alpha \cdot \mathbf{r}^{(\kappa)}) \equiv f(\alpha) = f^{(\kappa)} \; + \; \mathbf{g}^{(\kappa)^T} \cdot \mathbf{r}^{(\kappa)} \cdot \alpha \; + \; \frac{1}{2}\mathbf{r}^{(\kappa)^T} \cdot \mathbf{H}^{(\kappa)} \cdot \mathbf{r}^{(\kappa)} \cdot \alpha^2 \; + \; \ldots \tag{24}$$

$$= f^{(\kappa)} \; + \; \nabla f\,(\alpha = 0) \cdot \alpha \; + \; \frac{1}{2}\nabla^2 f\,(\alpha = 0) \cdot \alpha^2 \; + \; \ldots \tag{25}$$

Please note that Hessian, gradient, and search direction are combined to the first and second derivative of objective f with regard to step length α in (25):

$$\begin{aligned} \nabla f\,(\alpha = 0) \quad &: \quad \text{slope of } f \text{ along direction } \mathbf{r}^{(\kappa)} \\ \nabla^2 f\,(\alpha = 0) \quad &: \quad \text{curvature of } f \text{ along } \mathbf{r}^{(\kappa)} \end{aligned}$$

repeat
  determine the search direction $\mathbf{r}^{(\kappa)}$
  determine the step length $\alpha^{(\kappa)}$ (line search)
  $\mathbf{x}^{(\kappa+1)} = \mathbf{x}^{(\kappa)} + \alpha^{(\kappa)} \cdot \mathbf{r}^{(\kappa)}$
  $\kappa := \kappa + 1$
until termination criteria are fulfilled

Figure 4. Basic structure of an iterative optimization process.

2.5 Structure of an iterative optimization process

The basic structure of an iterative optimization process is given in Fig. 4. Iterative optimization usually partitions an iteration step into two parts. First, compute a suitable search direction and then a step length for that search direction. The second part is also called line search. We will present methods for line search and computation of a search direction in Chapters 5 and 6. The search direction is computed based on a local model of objective and constraints at the current iteration point. The step length then uses single function evaluations.

Steepest descent approach

A simple way to determine a search direction is the so-called steepest descent approach, which is solely based on the gradient in the current iteration step:

$$\mathbf{r}^{(\kappa)} = -\mathbf{g}^{(\kappa)} \tag{26}$$

The steepest descent approach often features very slow convergence, as is illustrated in Fig. 5.

The Rosenbrock function, although featuring just two optimization variables, is a very ill-conditioned optimization problem, as the function is a banana-shaped valley with highly steep side walls and the valley decreasing exceptionally slowly towards the optimum $\mathbf{x}^*$. Using the steepest descent direction, then will always move towards the opposite steep rise of the valley and hardly along the banana trajectory to the optimum. This leads to a strong zig-zagging of the iteration process.

Please note that this book will frequently illustrate optimization aspects with level contours of functions of two variables. It would be helpful if the reader familiarized this method of visualization. Two main features are necessary to "read" pictures of level contours:

- The direction of the gradient is orthogonal to a level contour (see Fig. 6). That means that the direction of the steepest ascent (the gradient direction) and the opposite direction of the steepest descent are orthogonal to a level contour.
- Neighbouring level contours usually represent a constant difference in the function value ("height"). That means that the closer to each other two level contours are, the steeper the slope.

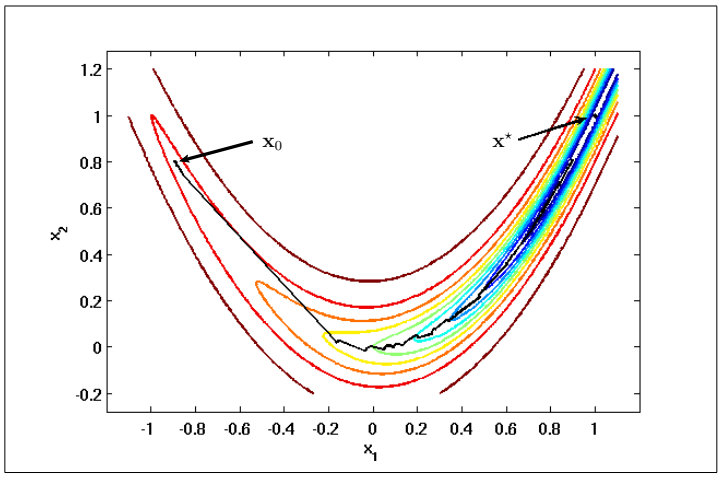


Figure 5. Visual illustration of the steepest-descent approach for Rosenbrock's function $f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$. A backtracking line search is applied (see Sec. 5.4, page 58) with an initial $\mathbf{x}^{(0)} = [-1.0, 0.8]^T$ and $\alpha^{(0)} = 1$, $\alpha := c_3 \cdot \alpha$. The search terminates when the Armijo condition is satisfied with $c_1 = 0.7, c_3 = 0.6$.

Trust-region approach

Trust-region approaches are not partitioning search direction and step length computation but compute them simultaneously. The process adapts the search direction according to increasing step length. A simple way is to model the objective function,

$$f(\mathbf{x}) \approx m\left(\mathbf{x}^{(\kappa)} + \mathbf{r}\right), \tag{27}$$

and determine a search direction that minimizes this model in a certain trust region, e.g., for the step length:

$$\min_{\mathbf{r}} m\left(\mathbf{x}^{(\kappa)} + \mathbf{r}\right) \quad \text{s.t.} \quad \mathbf{r} \in \text{ trust region} \tag{28}$$
$$\text{e.g., } \|\mathbf{r}\| < \Delta$$

Solving this problem (which can be done efficiently, as model evaluation is cheap) for several sizes of trust regions gives a set of search directions of different lengths.

Consideration of constraints

One approach is to combine the constraint functions and the objective function in an unconstrained optimization model in each iteration step. There are several ways to do that, e.g.:

- Lagrange formulation
- Penalty function
- Sequential Quadratic Programming (SQP)

These will be further elaborated or sketched in Chap. 6.

Another approach is a projection onto so-called active constraints in each iteration step. Active constraints are constraints that hit their bound. They are set to equality constraints for the moment. Equality-constrained problems can be transformed into unconstrained problems by coordinate transformation. This is called **active set method** and will further be elaborated in Chap. 6.

3 Optimality conditions

3.1 Optimality conditions – unconstrained optimization

The conditions for the optimality of a point will be derived based on a Taylor series of the objective f along a search direction $\mathbf{r}$ at the optimum point $\mathbf{x}^\star$. We use (22) and replace them with the respective terms at the optimum point:

$$f(\mathbf{x}^\star + \mathbf{r}) \equiv f(\mathbf{r}) = f^\star + \mathbf{g}^{\star T} \cdot \mathbf{r} + \frac{1}{2}\mathbf{r}^T \cdot \mathbf{H}^\star \cdot \mathbf{r} + \dots \tag{29}$$

The abbreviations are:

$f^\star = f(\mathbf{x}^\star)$:	optimum objective value at the optimum variable value $\mathbf{x}^\star$
$\mathbf{g}^\star = \nabla f(\mathbf{x}^\star)$:	gradient at optimum $\mathbf{x}^\star$
$\mathbf{H}^\star = \nabla^2 f(\mathbf{x}^\star)$:	Hessian matrix at optimum $\mathbf{x}^\star$

The point $\mathbf{x}^\star$ can only be (locally) optimal if there is no way to reduce the corresponding optimum objective value $f^\star$ in the surrounding of $\mathbf{x}^\star$. In other words, this can be formulated as follows:

$$\begin{aligned} &\mathbf{x}^\star \text{ is locally optimal } \Longleftrightarrow \\ &\text{there is no descent direction } \mathbf{r} \text{ that leads to a smaller objective value } f(\mathbf{r}) < f^\star \end{aligned} \tag{30}$$

A descent direction is derived from the linear term in the Taylor series (29):

$$\text{descent direction } \mathbf{r}: \ \nabla f\left(\mathbf{x}^{(\kappa)}\right)^T \cdot \mathbf{r} < 0 \tag{31}$$

The steepest descent is opposite to the gradient, which in turn represents the steepest ascent:

$$\text{steepest descent direction: } \mathbf{r} = -\nabla f\left(\mathbf{x}^{(\kappa)}\right) \tag{32}$$

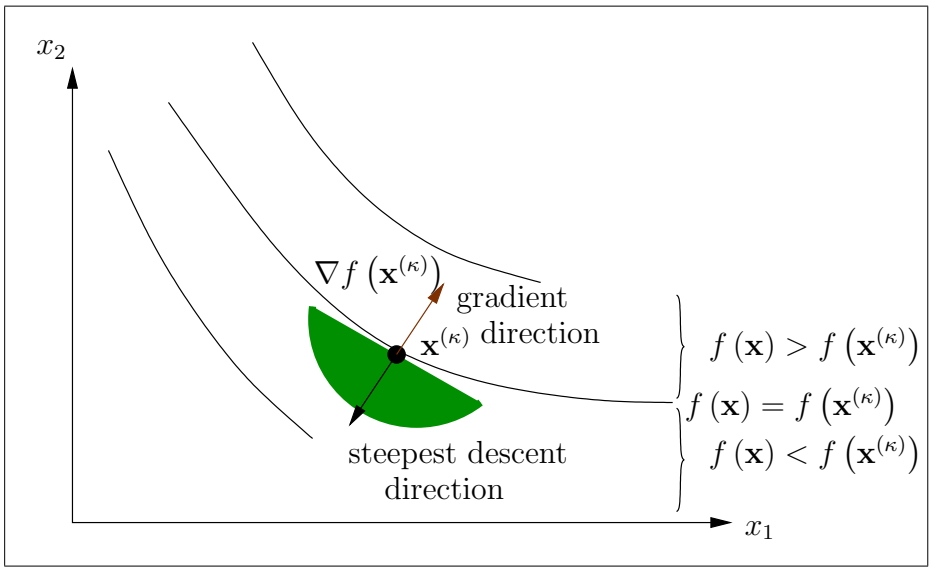


Figure 6. A descent direction from $\mathbf{x}^{(\kappa)}$ lies in the green shaded area.

Fig. 6 illustrates a situation where descent directions exist.

At point $\mathbf{x}^{(\kappa)}$, an intermediate solution of an iterative optimization process, the gradient is drawn perpendicular to the level contour of the optimization objective f. This is the direction of the steepest ascent towards higher values of f. Opposite is the steepest descent. Simplifying, all directions with an angle smaller than 90 degrees to the steepest descent represent descent directions. They are just not as steep. Please note that the 90-degree statement only holds for the linearization of objective f at $\mathbf{x}^{(\kappa)}$. Due to the curvature at this point, either direction with more than 90 degrees, as in this example point, may provide descent, or less, if the curvature is to the side of the descent.

The optimality conditions are derived based on a separate consideration of the first-order term and second-order term of the Taylor series (29). Each term individually shall be guaranteed not to provide a descent. This leads to the first-order optimality condition and second-order optimality condition, respectively.

3.1.1 First-order condition for a local minimum of an unconstrained optimization problem

No descent in the objective function is obtained if the linear term in the Taylor series (29) is non-negative for any direction $\mathbf{r}$:

$$\underset{\mathbf{r}\neq\mathbf{0}}{\forall} \; \mathbf{g}^{\star^T} \cdot \mathbf{r} \geq 0 \tag{33}$$

This can only be achieved if the gradient is zero, which means that all components of the gradient vector are zero:

$$\boxed{\mathbf{g}^\star = \nabla f\left(\mathbf{x}^\star\right) = \mathbf{0}} \tag{34}$$

If any component is non-zero, then there would be a descent opportunity in the direction of the corresponding variable.

The first-order optimality condition is a necessary condition for a local minimum.

A point $\mathbf{x}^\star$ satisfying the first-order optimality condition is denoted as **stationary point**.

3.1.2 Second-order condition for a local minimum of an unconstrained optimization problem

The second-order optimality conditions refer to the quadratic term in the Taylor series (29). As a necessary condition, it requires that this quadratic term does not provide a descent of the objective in any direction. As a sufficient condition, it requires that this quadratic term provides only increase in the objective value in any direction:

Necessary:

$$\boxed{\begin{aligned} \underset{\mathbf{r}\neq\mathbf{0}}{\forall} \; \mathbf{r}^T \cdot \nabla^2 f\left(\mathbf{x}^\star\right) \cdot \mathbf{r} \geq 0 \quad &\Longleftrightarrow \quad \nabla^2 f\left(\mathbf{x}^\star\right) \text{ is positive semidefinite} \\ &\Longleftrightarrow \quad f \text{ has non-negative curvature} \end{aligned}} \tag{35}$$

Sufficient:

$$\boxed{\begin{aligned} \underset{\mathbf{r}\neq\mathbf{0}}{\forall} \; \mathbf{r}^T \cdot \nabla^2 f\left(\mathbf{x}^\star\right) \cdot \mathbf{r} > 0 \quad &\Longleftrightarrow \quad \nabla^2 f\left(\mathbf{x}^\star\right) \text{ is positive definite} \\ &\Longleftrightarrow \quad \text{has positive curvature} \end{aligned}} \tag{36}$$

The second-order optimality condition immediately refers to the positive (semi-)definiteness of the Hessian matrix of the objective function at the optimum.

Fig. 7 illustrates quadratic objective functions in two variables with four situations of the definiteness of its Hessian matrix: positive definiteness (a), negative definiteness (b), indefiniteness (saddle point, c), and positive semidefiniteness (d).

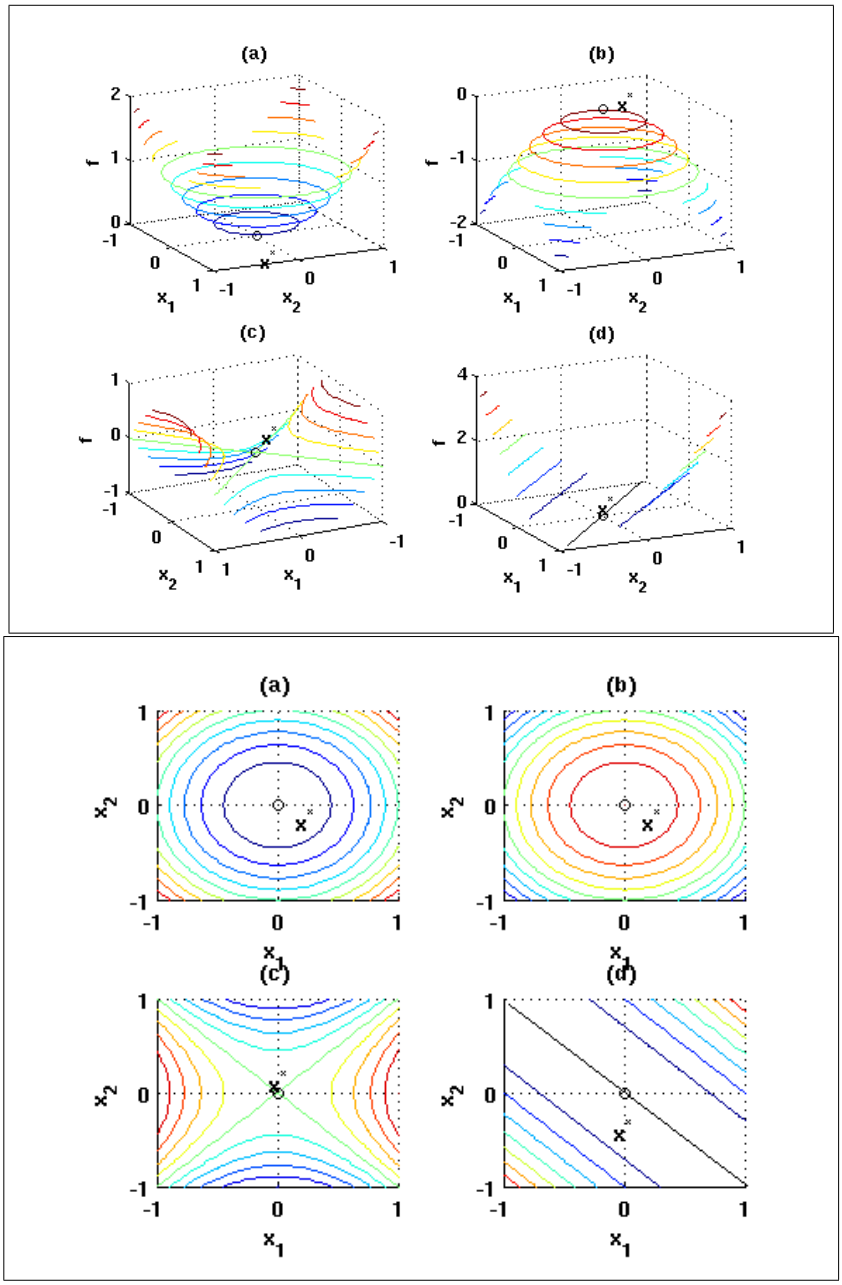


Figure 7. Quadratic functions (top) and their level contours (bottom): (a) positive definite Hessian matrix, minimum at $\mathbf{x}^\star$, (b) negative definite Hessian matrix, maximum at $\mathbf{x}^\star$, (c) indefinite Hessian matrix, saddle point at $\mathbf{x}^\star$, (d) positive semidefinite Hessian matrix, multiple minima along trench.

We can see that the stationary point is unique in each of the cases (a), (b), and (c). Depending on the curvature of the function, which is determined by the second-order derivative, the stationary point can be a maximum, minimum, or saddle point. Case (a) satisfies the sufficient second-order optimality condition (36). Case (d) satisfies the necessary second-order optimality condition (35). The objective has a minimum value here, but multiple points that lead to the minimum value.

There are several methods to check the positive definiteness of a second-order derivative $\nabla^2 f(\mathbf{x}^\star)$. Examples are:

- Eigenvalue decomposition: A symmetric matrix is positive definitive if all eigenvalues are > 0. This is the situation in case (a) of Fig. 7. In case (b), the eigenvalues are both negative. In case (c), one eigenvalue is positive, and one eigenvalue is negative. In case (d), one eigenvalue is positive, and the other is zero.
- Cholesky decomposition $\nabla f^2(\mathbf{x}^\star) = \mathbf{L} \cdot \mathbf{L}^T$: If all diagonal elements are positive, i.e., $l_{ii} > 0$, then the Hessian is positive definite.
- Cholesky decomposition $\nabla f^2(\mathbf{x}^\star) = \mathbf{L} \cdot \mathbf{D} \cdot \mathbf{L}^T$ with $l_{ii} = 1$: If all diagonal elements are positive, i.e., $d_{ii} > 0$, then the Hessian is positive definite.
- Gaussian elimination: If all pivot elements during the Gaussian elimination without pivoting are positive, and then the Hessian is positive definite.
- Determinant (see App. A): If all leading principal minors of a symmetric matrix are > 0, then the Hessian is positive definite.

3.2 Optimality conditions – constrained optimization

The optimality conditions for an unconstrained optimization problem have been derived based on a Taylor series of the optimization objective and by looking individually at the linear and quadratic model terms to define conditions for these terms that guarantee no more descent in the surrounding of the local optimum. In a constrained optimization problem, this concept of descent direction has to be expanded by the concept of feasible direction. A feasible direction is a direction that keeps a constraint satisfied. The following section will describe the idea of a feasible descent direction for inequality constraints. Equality constraints will be considered later.

3.2.1 Feasible descent direction r

A feasible descent direction is a direction that provides a descent in the objective function f while not violating an inequality constraint c_i. Both conditions are defined using the linear terms of the respective function models.

$$\text{descent direction:} \quad f\left(\mathbf{x}^{(\kappa)}+\mathbf{r}\right) \simeq f^{(\kappa)} + \nabla f\left(\mathbf{x}^{(\kappa)}\right)^T \cdot \mathbf{r}, \text{ i.e., } \nabla f\left(\mathbf{x}^{(\kappa)}\right)^T \cdot \mathbf{r} < 0 \quad (37)$$

$$\text{feasible direction:} \quad c_i\left(\mathbf{x}^{(\kappa)}+\mathbf{r}\right) \simeq c_i\left(\mathbf{x}^{(\kappa)}\right) + \nabla c_i\left(\mathbf{x}^{(\kappa)}\right)^T \cdot \mathbf{r} \geq 0 \quad (38)$$

Two cases are distinguished with regard to the constraint: it is either inactive or active. An inactive constraint still has some margin to its bound, while an active constraint has reached its bound. The notation is related to the active set method for constrained optimization (Chap. 6), where inequality constraints that reach their bound i.e., the bound is active, temporarily become equality constraints. The two cases are treated in the following.

Inactive constraint

A constraint c_i is inactive at the current iteration point $x^{(\kappa)}$, if the constraint value is greater than the bound:

$$\text{constraint } i \text{ is inactive} \iff c_i\left(\mathbf{x}^{(\kappa)}\right) > 0 \quad (39)$$

In that case, any search direction $\mathbf{r}$ is feasible as long as it does not trespass the bound. As length limitation, the linear constraint model (38) is used. For instance, the following search direction is satisfying (38) while going in the steepest descent direction with regard to the optimization objective:

$$\mathbf{r} = -\frac{c_i\left(\mathbf{x}^{(\kappa)}\right)}{\|\nabla c_i\left(\mathbf{x}^{(\kappa)}\right)\| \cdot \|\nabla f\left(\mathbf{x}^{(\kappa)}\right)\|} \cdot \nabla f\left(\mathbf{x}^{(\kappa)}\right) \quad (40)$$

This can be proven by inserting (40) into (38), which leads to:

$$c_i\left(\mathbf{x}^{(\kappa)}\right) \cdot \left[1 - \frac{\nabla c_i\left(\mathbf{x}^{(\kappa)}\right)^T \cdot \nabla f\left(\mathbf{x}^{(\kappa)}\right)}{\|\nabla c_i\left(\mathbf{x}^{(\kappa)}\right)\| \cdot \|\nabla f\left(\mathbf{x}^{(\kappa)}\right)\|}\right] \geq 0 \quad (41)$$

The inner term in the bracket of (41) describes the angle between constraint gradient and objective gradient at $x^{(\kappa)}$, for which holds:

$$-1 \leq \frac{\nabla c_i\left(\mathbf{x}^{(\kappa)}\right)^T \cdot \nabla f\left(\mathbf{x}^{(\kappa)}\right)}{\|\nabla c_i\left(\mathbf{x}^{(\kappa)}\right)\| \cdot \|\nabla f\left(\mathbf{x}^{(\kappa)}\right)\|} \leq 1 \quad (42)$$

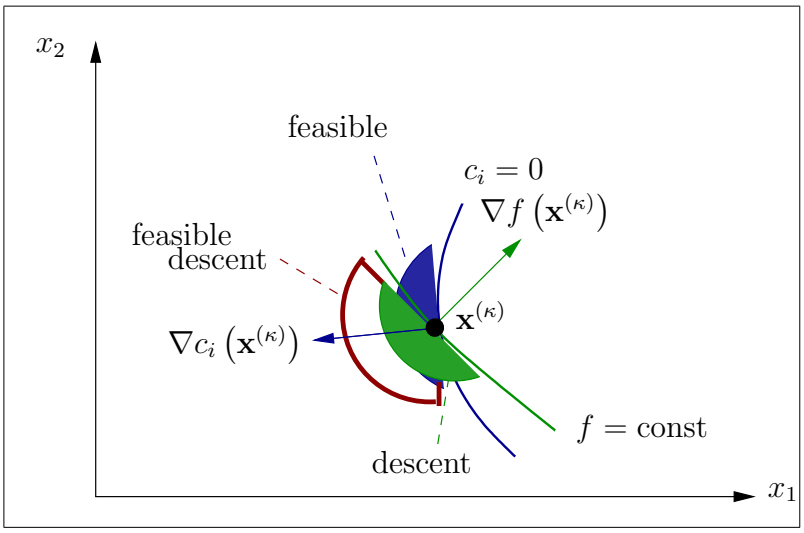


Figure 8. Dark/blue shaded area: feasible directions according to (45), light/green shaded area: descent directions according to (44), overlap: feasible descent directions.

Active constraint A constraint c_i is active at the current iteration point $x^{(\kappa)}$, if the constraint value is exactly at the bound:

$$\text{constraint } i \text{ is active} \iff c_i\left(\mathbf{x}^{(\kappa)}\right) = 0 \tag{43}$$

In that case, (37) and (38) become the following two conditions that describe a feasible descent direction:

$$\text{descent direction:} \quad \nabla f\left(\mathbf{x}^{(\kappa)}\right)^T \cdot \mathbf{r} < 0 \tag{44}$$

$$\text{feasible direction:} \quad \nabla c_i\left(\mathbf{x}^{(\kappa)}\right)^T \cdot \mathbf{r} \geq 0 \tag{45}$$

Fig. 8 illustrates the situation for an active inequality constraint c_i and an objective f at an iteration point $\mathbf{x}^{(\kappa)}$. The green curve shows the current level contour of the objective function, and the green arrow shows the corresponding objective gradient. The green-shaded circular sector indicates all descent directions from $\mathbf{x}^{(\kappa)}$. The blue curve shows the current level contour of the active constraint $c_i = 0$, and the blue arrow shows the corresponding constraint gradient. The blue-shaded circular sector indicates all feasible directions from $\mathbf{x}^{(\kappa)}$. Please note that optimization is towards smaller objective values; hence we go on the opposite side of the objective gradient. And a constraint is to be larger than zero, hence we go in the direction of the constraint gradient for an active constraint. Feasible descent directions are then determined by the overlapping part of the two circular sectors, which is indicated in purple.

Optimality of a constrained optimization problem means that there is no feasible descent.

Geometrically this means that the purple circular sector in Fig. 8 is empty. This is obtained if the optimal point $\mathbf{x}^\star$ satisfies:

$$\nabla f\left(\mathbf{x}^\star\right) = \lambda_i^\star \cdot \nabla c_i\left(\mathbf{x}^\star\right) \quad \text{with } \lambda_i^\star \geq 0 \tag{46}$$

The geometrical consideration can be extended to more than one active inequality constraint. This leads to the condition that the gradient of the objective is a linear combination with positive coefficients of all active inequality constraints:

$$\nabla f(\mathbf{x}^\star) = \sum_i \lambda_i^\star \cdot \nabla c_i(\mathbf{x}^\star) \text{ with } \forall_i \lambda_i^\star \geq 0 \tag{47}$$

Fig. 9 illustrates this for two active inequality constraints c_1, c_2 in two variables and one objective f and three different geometric situations. In the two upper cases, the linear combination of the gradients of the two active constraints yields the objective gradient only if the Lagrange multipliers are positive. At the same time, there is no overlapping region among feasibility and descent, hence no more feasible descent. Therefore, this is a stationary point of the constrained optimization problem. In the lower case, the Lagrange multipliers are negative and feasible descent is still possible; hence this is not a stationary point.

The complete first-order conditions for a local minimum of a constrained optimization problem are given in the following section.

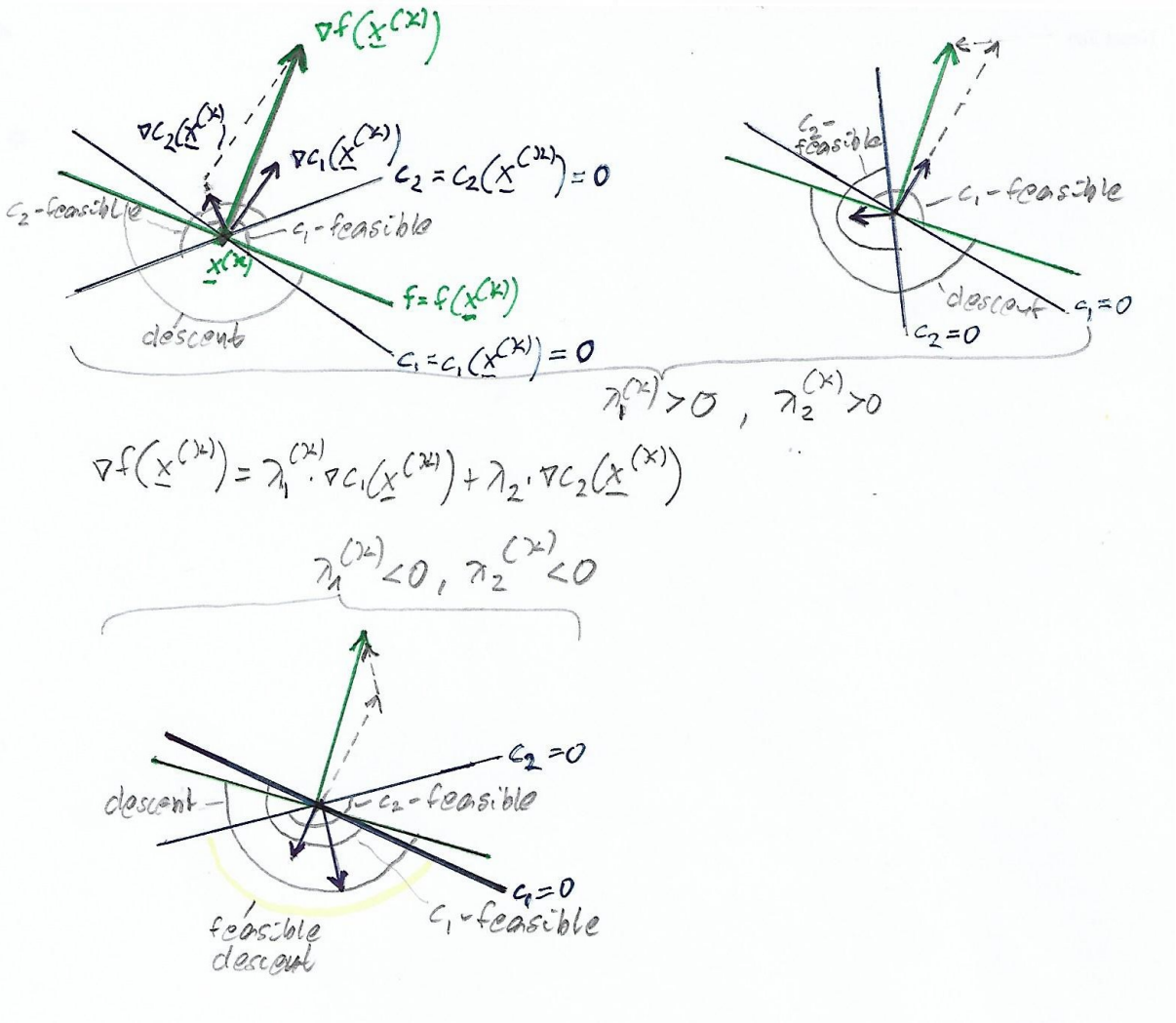


Figure 9. Illustration of condition (47) for two active constraints in two variables.

3.2.2 First-order conditions for a local minimum of a constrained optimization problem (Karush-Kuhn-Tucker (KKT) conditions)

The first-order necessary optimality conditions are formulated based on the Lagrange function. The optimum values of the variable vector, objective, Lagrange multiplier, and Lagrange function are denoted as $\mathbf{x}^\star$, $f^\star = f(\mathbf{x}^\star)$, $\boldsymbol{\lambda}^\star$, $\mathcal{L}^\star = \mathcal{L}(\mathbf{x}^\star, \boldsymbol{\lambda}^\star)$, respectively.

$$\nabla \mathcal{L}(\mathbf{x}^\star) = \mathbf{0} \tag{48}$$
$$c_i(\mathbf{x}^\star) = 0 \quad i \in E \tag{49}$$
$$c_i(\mathbf{x}^\star) \geq 0 \quad i \in I \tag{50}$$
$$\lambda_i^\star \geq 0 \quad i \in I \tag{51}$$
$$\lambda_i^\star \cdot c_i(\mathbf{x}^\star) = 0 \quad i \in E \cup I \tag{52}$$

We assume the index numbers in E and I are different: $E \cap I = \{\}$. Please note that we have already given a motivation for the Lagrange function as an objective function of an unconstrained optimization problem that models the original constrained optimization problem. The KKT condition (48) represents the first-order optimality condition for a stationary point of the unconstrained optimization problem (34) on the Lagrange function.

Inserting the Lagrange function (13) into (48) leads to:

$$\nabla f(\mathbf{x}^\star) - \sum_{i \in \mathcal{A}(\mathbf{x}^\star)} \lambda_i^\star \cdot \nabla c_i(\mathbf{x}^\star) = \mathbf{0} \tag{53}$$

Here, $\mathcal{A}(\mathbf{x}^\star)$ is the set of active constraints at $\mathbf{x}^\star$:

$$\mathcal{A}^\star = \mathcal{A}(\mathbf{x}^\star) = E \cup \{i \in I \mid c_i(\mathbf{x}^\star) = 0\} \tag{54}$$

Please note that equality constraints are always active. The set of active constraints contains all equality constraints and the set of inequality constraints that are at their respective bound.

(49) and (50) repeat the constraints of the original problem formulation. They result from the condition of a stationary point of the Lagrange function with regard to the Lagrange multiplier $\nabla \mathcal{L}(\boldsymbol{\lambda}^*) = \mathbf{0}$.

Please also note that (53) has already been motivated from a geometrical point of view into the same form (47) and that (51) has also been motivated from a geometrical point of view.

Please note that (51) reflects that nothing can be said for the sign of $\lambda_i^\star$ of an equality constraint apart from the fact that it is permanently active. This can be motivated by splitting an equality constraint into two inequality constraints:

$$c_i = 0 \iff c_i \leq 0 \wedge c_i \geq 0 \tag{55}$$

The two inequality constraints for the same function would yield contradicting conditions on the sign of $\lambda_i^\star$. It cannot be defined which of the two inequality constraints becomes active at the end of an iteration process.

(52) is called the **complementarity condition**. It says that either the Lagrange multiplier is 0 (for an inactive constraint), or the corresponding constraint c_i is 0 (i.e., is active in the optimum). This can be interpreted as an instruction for the optimization process to actively reset the Lagrange multipliers for all inactive constraints to zero. The formulation either/or is incorrect: there may be the special case that the solution of the constrained problem is the same as the solution without the constraint; nevertheless, the constraint is at its bound (i.e., is zero). In this special case, the Lagrange multiplier and the constraint are zero.

Inserting (52) into the Lagrange function (13) leads to the result that in the optimum, all terms with Lagrange multipliers disappear, such that the optimal value of the Lagrange function at (13) is just the optimal value of the objective:

$$\mathcal{L}^\star = f^\star \tag{56}$$

Condition (51) can also be derived in the following way. Take a point $\mathbf{x}_C$ close to the optimal point $\mathbf{x}^\star$ that satisfies the constraint $c_i(\mathbf{x}_C) \geq 0$. The Lagrange function being the minimum value of the objective, inserting $\mathbf{x}_C$ into the Lagrange function results in a larger value than the Lagrange function at $\mathbf{x}^\star$, and with (56) we obtain:

$$\mathcal{L}(\mathbf{x}_C, \lambda^\star) \geq \mathcal{L}(\mathbf{x}^\star, \lambda^\star) = f(\mathbf{x}^\star) \tag{57}$$

Inserting the Lagrange function (13) on the left side leads to

$$f(\mathbf{x}_C) - \sum_{i \in A^\star} \lambda_i^\star \cdot c_i(\mathbf{x}_C) \geq f(\mathbf{x}^\star), \tag{58}$$

which can be rewritten as

$$f(\mathbf{x}_C) \geq f(\mathbf{x}^\star) + \sum_{i \in A^\star} \lambda_i^\star \cdot c_i(\mathbf{x}_C) \tag{59}$$

As the constraint values are greater or equal to zero, the Lagrange multipliers must also be greater or equal to zero to fulfill the inequality.

3.2.3 Second-order condition for a local minimum of a constrained optimization problem

The second-order optimality condition for constrained optimization is derived by looking at the objective function at the optimal point $\mathbf{x}^\star$ along a search direction $\mathbf{r}$, using (56), and then developing the Lagrange function in a Taylor series up to the second term:

$$\begin{aligned} f(\mathbf{x}^\star + \mathbf{r}) &= \mathcal{L}(\mathbf{x}^\star + \mathbf{r}, \boldsymbol{\lambda}^\star) && (60)\\ &= \mathcal{L}(\mathbf{x}^\star, \boldsymbol{\lambda}^\star) + \mathbf{r}^T \cdot \underbrace{\nabla \mathcal{L}(\mathbf{x}^\star)}_{\mathbf{0}} + \frac{1}{2}\mathbf{r}^T \cdot \nabla^2 \mathcal{L}(\mathbf{x}^\star) \cdot \mathbf{r} + \cdots \\ &= f^\star + \frac{1}{2}\mathbf{r}^T \cdot \nabla^2 \mathcal{L}(\mathbf{x}^\star) \cdot \mathbf{r} + \cdots && (61)\\ \text{with } \nabla^2 \mathcal{L}(\mathbf{x}^\star) &= \nabla^2 f(\mathbf{x}^\star) - \sum_{i \in \mathcal{A}(\mathbf{x}^\star)} \lambda_i^\star \cdot \nabla^2 c_i(\mathbf{x}^\star) && (62) \end{aligned}$$

We have used (56) to obtain a quadratic model around the optimum objective value. We can see that the first-order derivative term disappears due to the condition (48). As in the unconstrained case, the Hessian of the Lagrange function at the optimum point has to satisfy positive curvature along specific directions.

A difference to the unconstrained case is that the Hessian of the Lagrange function is a combination of the Hessian matrix of the objective and the Hessian matrices of the active constraints. Another difference is that we do not require positive definiteness of the Hessian matrix of the Lagrange function but positive curvature only along **feasible stationary directions r** at $\mathbf{x}^\star$, which are are defined as:

$$\boxed{\mathcal{F}_\mathbf{r} = \left\{ \mathbf{r} \;\middle|\; \begin{array}{ll} \mathbf{r} \neq \mathbf{0} & \\ \nabla c_i(\mathbf{x}^\star)^T \cdot \mathbf{r} = 0 & i \in \mathcal{A}_+^\star = \left\{ j \in \mathcal{A}(\mathbf{x}^\star) \,\middle|\, j \in E \vee \lambda_j^\star > 0 \right\} \\ \nabla c_i(\mathbf{x}^\star)^T \cdot \mathbf{r} \geq 0 & i \in \mathcal{A}(\mathbf{x}^\star) \setminus \mathcal{A}_+^\star \end{array} \right\}} \tag{63}$$

The necessary second-order optimality condition for constrained optimization using $\mathcal{F}_\mathbf{r}$ then is:

$$\boxed{\underset{\mathbf{r} \in \mathcal{F}_\mathbf{r}}{\forall} \mathbf{r}^T \cdot \nabla^2 \mathcal{L}(\mathbf{x}^\star) \cdot \mathbf{r} \geq 0} \tag{64}$$

The sufficient second-order optimality condition for constrained optimization using $\mathcal{F}_\mathbf{r}$ is:

$$\boxed{\underset{\mathbf{r} \in \mathcal{F}_\mathbf{r}}{\forall} \mathbf{r}^T \cdot \nabla^2 \mathcal{L}(\mathbf{x}^\star) \cdot \mathbf{r} > 0} \tag{65}$$

The set of feasible stationary directions $\mathcal{F}_\mathbf{r}$ reflects that a particular set of active constraints defines the optimum. The solution cannot be somewhere where these constraints are inactive; therefore, it is only necessary to check that the Lagrange Hessian has a positive curvature in the subspace where these constraints stay active. This is defined by the second row in (63), which defines search directions that keep the linear model term of the constraint zero so that the constraint does not move away from its bound. The second row in (63) is specifically for essential equality and inequality constraints because their Lagrange multiplier is positive. There may be inequality constraints that became active in

the optimum, but deleting them would not change the solution. These are characterized by a zero constraint value and a zero Lagrange multiplier. We do have to check the positive curvature of the Lagrange Hessian in the complete half-space where these constraints are not active, as in the unconstrained case. This leads to the third line in (63). Here, the given inequality for its linear model term defines the feasible side of the constraint.

3.2.4 Sensitivity of the optimum objective value with regard to a change in an active constraint

How much does the solution change if a constraint is slightly changed? This question can be answered by performing a new iterative optimization process with a modified constraint. It can also be approximated by determining the sensitivity of the solution with regard to the constraint.

Assume a perturbation of an active constraint at $\mathbf{x}^\star$ by some $\Delta_i \geq 0$. This changes the constraint $c_i(\mathbf{x}) \geq 0$ to $c_i(\mathbf{x}) \geq \Delta_i \Leftrightarrow c_i(\mathbf{x}) - \Delta_i \geq 0$, which gives a modified Lagrange function

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\Delta}) = f(\mathbf{x}) - \sum_i \lambda_i \cdot (c_i(\mathbf{x}) - \Delta_i) \tag{66}$$

The partial derivative of this Lagrange function with regard to Δ_i is calculated as follows:

$$\begin{aligned} \nabla_{\Delta_i} f^\star &= \nabla_{\Delta_i} \mathcal{L}^\star \ /^* \text{ with } \mathcal{L}^\star = \mathcal{L}^\star(\mathbf{x}^\star(\Delta_i), \lambda^\star(\Delta_i), (\Delta_i)) \ ^*/ \\ &= \Bigg(\underbrace{\frac{\partial \mathcal{L}}{\partial \mathbf{x}^T}}_{\mathbf{0}^T} \cdot \frac{\partial \mathbf{x}}{\partial \Delta_i} + \underbrace{\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}^T}}_{\mathbf{0}^T} \cdot \frac{\partial \boldsymbol{\lambda}}{\partial \Delta_i} + \frac{\partial \mathcal{L}}{\partial \Delta_i^T} \Bigg) \Bigg|_{\mathbf{x}^\star, \boldsymbol{\lambda}^\star} \\ &= \left. \frac{\partial \mathcal{L}}{\partial \Delta_i} \right|_{\mathbf{x}^\star, \lambda^\star} = \lambda_i^\star \end{aligned} \tag{67}$$

This means, in summary:

$$\boxed{\nabla_{\Delta_i} f^\star = \lambda_i^\star} \tag{68}$$

In words: The sensitivity of the optimal objective value $f^\star$ with regard to a change in an active constraint c_i is defined by the corresponding Lagrange factor $\lambda_i^\star$.

This attributes the Lagrange multiplier with a significant practical interpretation. The sensitivity interpretation is used in the active set method for constrained optimization in Chap. 6.

4 Worst-case analysis

The technical problem of worst-case analysis will be formulated based on optimization and optimality conditions. First, the task will be described in words and then "translated" into the mathematical problem formulation. Before that, some preliminaries on tolerance regions and linear performance models will be given.

Please note that the worst-case analysis task will be formulated for one performance feature φ without index i for simplicity. Please keep in mind that the following problems then have to be solved for all performance features individually. Individual worst-case analysis for specific performance features instead of one-fits-all worst-case parameter sets, or few-fit-all, is one of the main messages of this book!

Please note that no index d, s, or r for the parameter vector $\mathbf{x}$ will be given for simplicity. It will turn out later that each of the three types of worst-case analysis presented in the following, i.e., classical, realistic, general, is suited for different parameter types or statistical distribution types. For now, $\mathbf{x}$ stands for any or all parameters.

4.1 Task

The prevalent understanding of worst-case analysis is to simulate a given set of worst-case parameter sets, often called corner cases in practice, and check if the performance specification is satisfied.

These corner cases are determined for certain manufacturing process conditions that are known to drive a few key performance features to the worst values. Such a key performance feature is the delay in integrated circuits, which determines the speed and operating frequency. Corner cases are determined by driving PMOS and NMOS transistor parameters into regions that make them either slow or fast. Both can be made slow or fast, or the PMOS transistors are made slow and the NMOS transistors fast, or vice versa. This leads to corner cases slow-slow, fast-fast, fast-slow, slow-fast. More complex combinations have been developed.

But things are more complicated in general. There are more performance features beyond delay, so how do such corners reflect their worst cases? And what probability does a corner case represent? Then, there are local variations. Here, the practice moves to statistical methods like Monte-Carlo simulation, but that is already a solution method. A problem formulation should be done before thinking about the solution approach. So let us properly formulate a worst-case analysis:

The task of worst-case analysis consists of computing

- the **worst-case performance** value φ_W for a given performance feature
- and the corresponding **worst-case parameter set** $\mathbf{x}_W$
- that the circuit takes over a given **tolerance region** T of parameters.

We distinguish between performance features that should obtain maximum values, e.g., gain, slew rate, and performance features that should get minimum values, e.g., power, area, by design. If a good performance value is large, the worst-case is a small value ("lower worst-case"), and it should not go below a specified lower bound. If a good performance value is small, the worst-case is a large value ("upper worst-case"), and it should not exceed a specified upper bound.

"good" performance	specification type	"bad" performance	worst-case performance and parameter set
$\varphi \uparrow$	$\varphi \geq \varphi_L$ (lower bound)	$\varphi \downarrow$	$\varphi_{WL} = \varphi(\mathbf{x}_{WL})$ (lower worst case)
$\varphi \downarrow$	$\varphi \leq \varphi_U$ (upper bound)	$\varphi \uparrow$	$\varphi_{WU} = \varphi(\mathbf{x}_{WU})$ (upper worst case)

The mathematical formulation of computing the lower or upper worst-case performance value over a given tolerance region then yields the following constrained optimization problem of worst-case analysis:

$$\boxed{\min / \max \varphi(\mathbf{x}) \text{ , s.t. } \mathbf{x} \in T_{R/E} \quad \longrightarrow \quad \mathbf{x}_{WL/U}, \varphi_{WL/U} = \varphi\left(\mathbf{x}_{WL/U}\right)} \tag{69}$$

4.2 Typical tolerance regions

Two typical tolerance regions are rectangular or ellipsoidal forms. A rectangular form is called an interval, rectangle, box, or hyperrectangle for dimensions 1, 2, 3, and beyond. An ellipsoidal form does not exist in a one-dimensional space where just intervals exist. An ellipsoidal form is called ellipsis, ellipsoid, and hyperellipsoid for dimensions 2, 3, and beyond. The formulas for a hyperrectangle and a hyperellipsoid are given as follows:

Hyperrectangle: $T_R = \{\mathbf{x} \,|\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U\}$

Hyperellipsoid: $T_E = \left\{\mathbf{x} \,\middle|\, \beta^2(\mathbf{x}) = (\mathbf{x}-\mathbf{x}_0)^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{x}-\mathbf{x}_0) \leq \beta_W^2\right\}$

$\mathbf{C}$ is symmetric, positive definite

Please note that the bounds of a hyperrectangle are independent for each parameter, while the bounds of a hyperellipsoid are a combination of all parameters. The quadratic formulation of a hyperellipsoid is chosen to represent the exponent of a multivariate normal distribution (Chap. 7).

Fig. 10 illustrates both types of tolerance regions for two parameters, i.e., a rectangle and an ellipsis.

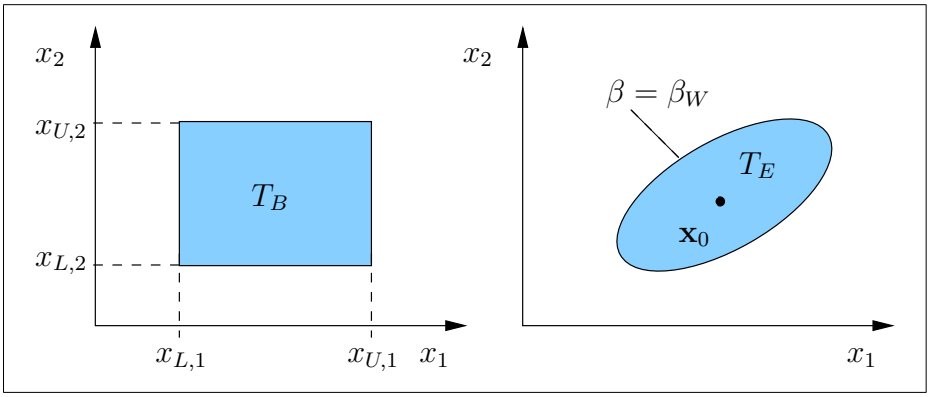


Figure 10. Tolerance hyperrectangle, tolerance hyperellipsoid for two parameters.

Tolerance hyperrectangles are especially suited to describe tolerance intervals of range parameters and tolerances of discrete parameters (their values are mainly at the end of the interval due to the manufacturing testing).

Tolerance hyperellipsoids are especially suited to describe the statistical distribution of statistical parameters, e.g., transistor model parameters.

4.3 Linear performance model

The classical and the realistic worst-case analysis that will be presented next are based on a linear model of the performance feature. A linear model can be established based on a sensitivity analysis of the circuit/system at any point $\mathbf{x}_a$:

$$\overline{\varphi}(\mathbf{x}) = \varphi_a + \mathbf{g}^T \cdot (\mathbf{x} - \mathbf{x}_a) \tag{70}$$

$$\varphi_a = \varphi(\mathbf{x}_a) \tag{71}$$

$$\mathbf{g} = \nabla\varphi(\mathbf{x}_a) \tag{72}$$

In practice, the derivatives of performance functions are often not available in the simulation. Then, a finite difference approximation of the gradient can be computed. The finite difference approximation can be forward, backward, or central. The **forward finite difference approximation** is determined by additional simulations with one respective parameter altered by an amount Δx_i at a time and computing the quotient of the performance difference between starting value x_i and altered value $x_i + \Delta x_i$ and of the parameter difference Δx_i. In matrix/vector notation, this can be formulated with the help of a vector $\mathbf{e}_i$ that is zero except the component i, where it is one, such that it addresses an alteration in the parameter vector $\mathbf{x}$ at the i-th position:

$$\nabla\varphi(x_{a,i}) \approx g_i = \frac{\varphi(\mathbf{x}_a + \Delta x_i \cdot \mathbf{e}_i) - \varphi(\mathbf{x}_a)}{\Delta x_i} \tag{73}$$

$$\mathbf{e}_i = [0 \ldots 0 \quad \underset{\uparrow}{1} \quad 0 \ldots 0]^T \qquad \uparrow \ i\text{-th position} \tag{74}$$

Fig. 11 illustrates the two approaches for a single parameter. The gradient is either computed exactly or approximated by a secant. The parameter Δ_i has to be chosen carefully. If it is too small, it will be "swallowed" by the noise in the simulation. If it is too large, it will deviate significantly from the gradient. In certain applications, a linear model may be wanted that gives a broader trend in performance than the gradient can provide. Δ_i will be determined for that purpose rather than approximating the gradient. For instance, the performance trend in a 1-sigma or 3-sigma range of statistical parameters may be targeted.

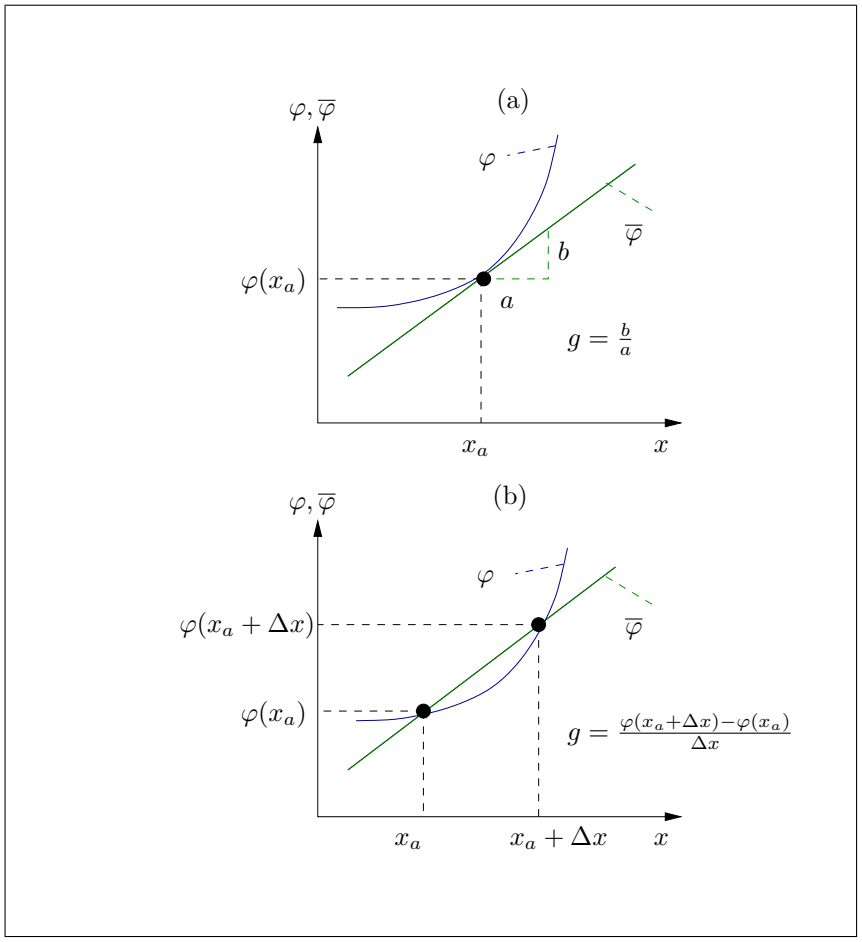


Figure 11. Linear performance model based on gradient (a), based on forward finite-difference approximation of gradient (b).

4.4 Classical worst-case analysis

The classical worst-case analysis is based on

- a hyperrectangle tolerance region $T_R = \{\mathbf{x} \,|\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U\}$, and
- a linear performance model $\overline{\varphi}(\mathbf{x}) = \varphi_a + \mathbf{g}^T \cdot (\mathbf{x} - \mathbf{x}_a)$

Fig. 12 illustrates the situation for two parameters. The tolerance rectangle T_R and some equidistant level contours of the linear performance model $\overline{\varphi}$ are shown. The gradient $\mathbf{g}$ points in the direction of the steepest ascent (i.e., strongest increase) in the performance function. From the formulation of worst-case analysis as the upper worst-case or lower worst-case value of the performance function over a tolerance region, we can identify the corresponding worst-case parameter sets $\mathbf{x}_{WL}$ and $\mathbf{x}_{WU}$ from visual inspection: $\mathbf{x}_{WU}$ is at the upper right corner. Here is the level contour with the maximum (i.e., upper worst case) performance value that can be reached with a point within the tolerance region. Analogously, $\mathbf{x}_{WL}$ is at the lower left corner. Here is the level contour with minimum (i.e., lower worst case) performance value that can be reached with a point within the tolerance region. The classical worst-case analysis relates to so-called corner cases, as we go into the corner of a hyperrectangle (in the case of a non-zero gradient component). Please note that intervals of parameters are individually defined without consideration of any (e.g., statistical) relation among parameters.

In the following, the mathematical problem formulation of classical worst-case analysis for the lower worst-case will be given, and an analytical solution will be derived based on optimality conditions.

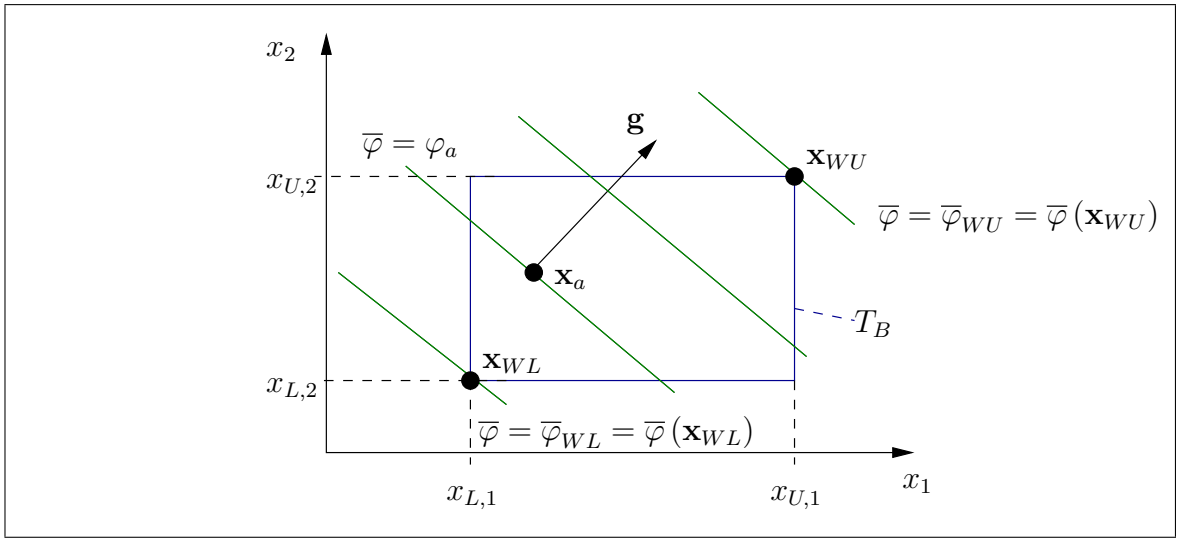


Figure 12. Classical worst-case analysis with tolerance hyperrectangle and linear performance model.

4.4.1 Classical worst-case analysis, lower worst case

The lower worst case searches the minimum value of the performance function over the tolerance hyperrectangle. That leads to a constrained minimization problem with the linear performance model as objective and the tolerance hyperrectangle as a constraint. We can leave out the constant parts in the linear model and move the remaining scalar product $\mathbf{g}^T \cdot \mathbf{x}$ into a component-wise notation. Likewise, we can formulate the hyperrectangle with individual intervals for the parameters. This leads to the following formulation:

$$\min \sum_i g_i \cdot x_i \text{ s.t. } \forall_i \left(x_{L,i} \leq x_i \leq x_{U,i}\right) \tag{75}$$

Inspection of this formulation reveals that each parameter can be treated individually, which gives the following problem formulation of classical worst-case analysis:

$$\boxed{\forall_i \left(\min g_i \cdot x_i \text{ s.t. } \left(x_{L,i} \leq x_i \leq x_{U,i}\right)\right)} \tag{76}$$

This is a special case of a linear programming problem. While linear programming problems require iterative solutions in general, we can calculate an analytical solution for this special case in the following. We write the corresponding Lagrange function for one parameter:

$$\mathcal{L}(x_i, \lambda_{L,i}, \lambda_{U,i}) = g_i \cdot x_i - \lambda_{L,i} \cdot (x_i - x_{L,i}) - \lambda_{U,i} \cdot (x_{U,i} - x_i) \tag{77}$$

The gradient of this Lagrange function with regard to x_i is:

$$\nabla \mathcal{L}(x_i) = g_i - \lambda_{L,i} + \lambda_{U,i} \tag{78}$$

Let us check the second-order optimality condition before calculating the stationary point. The Hessian matrix of the Lagrange function is a zero matrix. This results from the problem formulation being a linear programming problem with no higher-order derivatives. But a zero matrix at least satisfies the necessary second-order optimality condition.

For the stationary point, we have to distinguish two cases.

Case I: The gradient of parameter i is non-zero: $g_i \neq 0$.
As the objective is linear and does not have a minimum, we can state that the solution is in an active constraint. From inspection of the constraint, we can state that neither bounds can be active. That means either the lower bound $x_{L,i}$ is active and the upper bound is inactive $x_{U,i}$, or the upper bound is active and the lower bound is inactive:

$$\left[\lambda^\star_{L,i} > 0 \wedge \lambda^\star_{U,i} = 0\right] \oplus \left[\lambda^\star_{L,i} = 0 \wedge \lambda^\star_{U,i} = 0\right] \tag{79}$$

This is equivalent to:

$$\left[x^\star_i = x_{L,i}\right] \oplus \left[x^\star_i = x_{U,i}\right] \tag{80}$$

How do we decide which one it is? Using (79) and considering that Lagrange multipliers are positive in the optimum, (78) with (48) leads to:

$$\left[g_i = \lambda^\star_{L,i} > 0\right] \oplus \left[g_i = -\lambda^\star_{U,i} < 0\right] \tag{81}$$

This means that the sign of the gradient determines where the lower worst-case is. The lower worst-case is at the lower interval bound if the gradient is positive. The lower worst-case is at the upper interval bound if the gradient is negative. This is what we confirm from visual inspection of Fig. 12.

Case II: The gradient of parameter i is zero: $g_i = 0$.
The objective in (76) disappears, and we are left with satisfying the constraint. That means no worst-case parameter set excists; it is undefined within the given interval. This corresponds to a zero gradient, which says that the performance is insensitive to a parameter, i.e., it does not change if the parameter changes.

Overall, we obtain the following analytical solution of classical lower worst-case analysis. One component $x_{WL,i}$ of the worst-case parameter vector $\mathbf{x}_{WL}$ is:

$$\boxed{x_{WL,i} = \begin{cases} x_{L,i}, & g_i > 0 \\ x_{U,i}, & g_i < 0 \\ \text{undefined}, & g_i = 0 \end{cases}} \tag{82}$$

The resulting lower worst-case performance value $\overline{\varphi}_{WL}$ of the linear performance model is:

$$\boxed{\overline{\varphi}_{WL} = \overline{\varphi}(\mathbf{x}_{WL}) = \varphi_a + \mathbf{g}^T \cdot (\mathbf{x}_{WL} - \mathbf{x}_a) = \varphi_a + \sum_i g_i \cdot (x_{WL,i} - x_{a,i})} \tag{83}$$

Simulation can be used to obtain the true performance value at $\mathbf{x}_{WL}$:

$$\varphi_{WL} = \varphi(\mathbf{x}_{WL}) \tag{84}$$

4.4.2 Classical worst-case analysis, upper worst case

The upper worst case is obtained by maximizing the performance. This is formulated by adding a minus sign to the objective in (76) and leads to the following analytical solution of classical upper worst-case analysis. One component $x_{WU,i}$ of the worst-case parameter vector $\mathbf{x}_{WU}$ is:

$$\boxed{x_{WU,i} = \begin{cases} x_{L,i}, & g_i < 0 \\ x_{U,i}, & g_i > 0 \\ \text{undefined}, & g_i = 0 \end{cases}} \tag{85}$$

The resulting upper worst-case performance value $\overline{\varphi}_{WU}$ of the linear performance model is:

$$\boxed{\overline{\varphi}_{WL} = \overline{\varphi}(\mathbf{x}_{WU}) = \varphi_a + \mathbf{g}^T \cdot (\mathbf{x}_{WU} - \mathbf{x}_a) = \varphi_a + \sum_i g_i \cdot (x_{WU,i} - x_{a,i})} \tag{86}$$

Simulation can be used to obtain the true performance value at $\mathbf{x}_{WU}$:

$$\varphi_{WU} = \varphi(\mathbf{x}_{WU}) \tag{87}$$

4.5 Realistic worst-case analysis

The realistic worst-case analysis is based on

- a hyperellipsoid tolerance region $T_E = \left\{ \mathbf{x} \,\middle|\, (\mathbf{x} - \mathbf{x}_0)^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{x} - \mathbf{x}_0) \leq \beta_W^2 \right\}$, and
- a linear performance model around any parameter point $\mathbf{x}_a$, which is transformed into a linear performance model around the center of the hyperellipsoid $\mathbf{x}_0$, $\overline{\varphi}(\mathbf{x}) = \varphi_a + \mathbf{g}^T \cdot (\mathbf{x} - \mathbf{x}_a) = \varphi_0 + \mathbf{g}^T \cdot (\mathbf{x} - \mathbf{x}_0)$ with $\varphi_0 = \varphi_a + \mathbf{g}^T \cdot (\mathbf{x}_0 - \mathbf{x}_a)$

Fig. 13 illustrates the situation for two parameters. The tolerance ellipsis T_E and some equidistant level contours of the linear performance model $\overline{\varphi}$ are shown. The gradient $\mathbf{g}$ points in the direction of the steepest ascent (i.e., strongest increase) in the performance function. From the formulation of worst-case analysis as the upper worst-case or lower worst-case value of the performance function over a tolerance region, we can identify the corresponding worst-case parameter sets $\mathbf{x}_{WL}$ and $\mathbf{x}_{WU}$ from visual inspection: $\mathbf{x}_{WU}$ is where a level contour of the performance is touching the tolerance ellipsoid (i.e., is tangential to the tolerance ellipsoid) in the direction of the gradient. Here is the level contour with the maximum (i.e., upper worst case) performance value that can be reached with a point within the tolerance region. Analogously, $\mathbf{x}_{WL}$ is where a level contour of the performance is touching the tolerance ellipsoid (i.e., is tangential to the tolerance ellipsoid) in the opposite direction of the gradient. Here is the level contour with minimum (i.e., lower worst case) performance value that can be reached with a point within the tolerance region. Unlike in the classical worst-case analysis where (except zero gradient) always a corner represents the worst-case, the worst-case parameter set can be any point on the surface of the hyperellipsoid in a realistic worst-case analysis. Which point depends not only on the signs of the gradient components but on the values. Visual inspection alone does not allow us to find a formulation of the worst-case parameter sets. Please note that the realistic worst-case analysis allows statistical correlation among parameters to be considered, as its level contours coincide with the level contours of a multivariate normal distribution. This will lead to a more realistic characterization of the worst case than the classical worst-case analysis, hence the term.

In the following, the mathematical problem formulation of realistic worst-case analysis for the lower worst-case will be given, and an analytical solution will be derived based on optimality conditions.

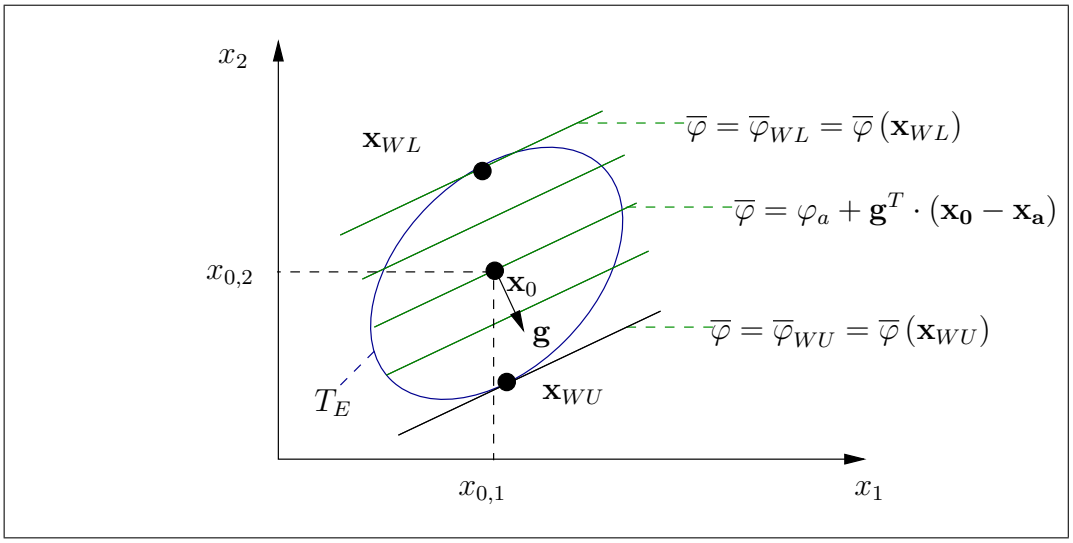


Figure 13. Realistic worst-case analysis with tolerance ellipsoid and linear performance model.

4.5.1 Realistic worst-case analysis, lower worst case

The lower worst case represents the minimum value of the performance function over the tolerance hyperellipsoid. That leads to a constrained minimization problem with the linear performance model as objective and the tolerance hyperellipsoid as a constraint. We can leave out the constant parts in the linear model, then the scalar product $\mathbf{g}^T \cdot \mathbf{x}$ remains as objective with the tolerance ellipsoid as a constraint:

$$\boxed{\min \mathbf{g}^T \cdot \mathbf{x} \quad \text{s.t.} \quad (\mathbf{x} - \mathbf{x}_0)^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{x} - \mathbf{x}_0) \leq \beta_W^2} \tag{88}$$

This is a specific nonlinear programming problem; it has a linear objective function and one quadratic constraint function. In the following, we will derive an analytical solution. The corresponding Lagrange function of the problem is:

$$\mathcal{L}(\mathbf{x}, \lambda) = \mathbf{g}^T \cdot \mathbf{x} - \lambda \left(\beta_W^2 - (\mathbf{x} - \mathbf{x}_0)^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{x} - \mathbf{x}_0) \right) \tag{89}$$

The first-order optimality conditions for the stationary point $\mathbf{x}_{WL}$ and Lagrange multiplier λ_{WL} are the following:

$$\nabla \mathcal{L}(\mathbf{x}) = \mathbf{g} + 2\lambda_{WL} \cdot \mathbf{C}^{-1} \cdot (\mathbf{x}_{WL} - \mathbf{x}_0) = \mathbf{0} \tag{90}$$

$$(\mathbf{x}_{WL} - \mathbf{x}_0)^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{x}_{WL} - \mathbf{x}_0) = \beta_W^2 \tag{91}$$

$$\lambda_{WL} > 0 \tag{92}$$

The constraint is active because of the following consideration: assume it would be inactive, then $\lambda_{WL} = 0$, and from (90), it would follow that $\mathbf{g} = \mathbf{0}$, which contradicts our implicit assumption: that the problem is technically reasonable and φ is sensitive to at least one of the considered parameters.

The second-order optimality condition holds for $\nabla^2 \mathcal{L}(\mathbf{x}) = 2\lambda_{WL} \cdot \mathbf{C}^{-1}$, because $\mathbf{C}^{-1}$ is positive definite, and because of (92). That means the Hessian matrix of the Lagrange function is positive definite.

We can rewrite (90) as follows,

$$\mathbf{x}_{WL} - \mathbf{x}_0 = -\frac{1}{2\lambda_{WL}} \cdot \mathbf{C} \cdot \mathbf{g} \tag{93}$$

and insert it into (91) to obtain:

$$\frac{1}{4\lambda_{WL}^2} \mathbf{g}^T \cdot \mathbf{C} \cdot \mathbf{g} = \beta_W^2 \tag{94}$$

Extracting $\frac{1}{2\lambda_{WL}}$ from (94), using that the final Lagrange factor is positive, and inserting it into (93) gives an analytical formula for the lower worst-case parameter vector in terms of the performance gradient and tolerance region constants:

$$\boxed{\begin{aligned} \mathbf{x}_{WL} - \mathbf{x}_0 &= -\frac{\beta_W}{\sqrt{\mathbf{g}^T \cdot \mathbf{C} \cdot \mathbf{g}}} \cdot \mathbf{C} \cdot \mathbf{g} \\ &= -\frac{\beta_W}{\sigma_{\overline{\varphi}}} \cdot \mathbf{C} \cdot \mathbf{g} \end{aligned}} \tag{95}$$

$\sigma_{\overline{\varphi}}$ denotes the standard deviation of the linearized performance. We will prove the correctness of this relation in Chap. 8.

By substituting (95) into the linear performance model, the corresponding lower worst-case performance values are obtained as:

$$\begin{aligned} \overline{\varphi}_{WL} &= \overline{\varphi}\left(\mathbf{x}_{WL}\right) = \varphi_0 + \mathbf{g}^T \cdot \left(\mathbf{x}_{WL} - \mathbf{x}_0\right) \\ &= \varphi_0 - \beta_W \cdot \sqrt{\mathbf{g}^T \cdot \mathbf{C} \cdot \mathbf{g}} \\ &= \varphi_0 - \beta_W \cdot \sigma_{\overline{\varphi}} \end{aligned} \tag{96}$$

Here, we must stop and look at what (96) reveals. In words, it says:

> The worst-case performance value is β_W times the performance standard deviation away from its nominal value.

This gives us, on the one hand, information on how we should choose the size, i.e., β_W, of the tolerance hyperellipsoid in a realistic worst-case analysis.

On the other hand, β_W gives us a characteristic variable of tolerance design. You can hear people talk about **x-sigma design**, but virtually no one can tell you what it means if there is a multivariate parameter space.

> The size β_W of the tolerance region defines the β_W**-sigma worst case** of a single performance feature for a multivariate parameter space, or, in other words, describes the β_W**-sigma robustness** of this performance feature.
>
> β_W is defined as **worst-case distance**.

If we are interested in the 3-sigma robustness, we use $\beta_W = 3$. If we are interested in the 6-sigma robustness, we use $\beta_W = 6$, etc.

We do not only anticipate from the later Chap. 7 that (95) features the standard deviation of the linearized performance but also that a linear performance model maps a multivariate normal distribution of parameters into a normal distribution of the corresponding performance. β_W, therefore, represents the multiple of the standard deviation of a normally distributed (linear) performance. That means we can use our knowledge of the percentage of occurrence in intervals of one-dimensional normal distributions. E.g., the interval from $-\infty$ to 3-sigma refers to 99.9%.

In other words:

> The yield of a half-space of a multivariate normal distribution cut by a hyperplane (Fig. 13) such that the maximum hyperellisoid inside the half-space has a size of β_W determined by the $[-\infty$ to $\beta_W]$-sigma range of a one-dimensional normal distribution.

This is an exact yield value; no statistical estimation is needed! We will give evidence and more illustration of this finding later on.

Please note from (96) that we do not have to compute the worst-case parameter set if we are interested in the worst-case value of the linear performance model.

Simulation can be used to obtain the true performance value at $\mathbf{x}_{WL}$:

$$\varphi_{WL} = \varphi(\mathbf{x}_{WL}) \tag{97}$$

4.5.2 Realistic worst-case analysis, upper worst case

The upper worst case is obtained by maximizing the performance. This is formulated by adding a minus sign to the objective in (88) and leads to the following analytical solution of realistic upper worst-case analysis.

$$\boxed{\begin{array}{rcl} \mathbf{x}_{WU} - \mathbf{x}_0 & = & +\frac{\beta_W}{\sqrt{\mathbf{g}^T \cdot \mathbf{C} \cdot \mathbf{g}}} \cdot \mathbf{C} \cdot \mathbf{g} \\ & = & +\frac{\beta_W}{\sigma_{\overline{\varphi}}} \cdot \mathbf{C} \cdot \mathbf{g} \end{array}} \tag{98}$$

The resulting upper worst-case performance value $\overline{\phi}_{WU}$ of the linear performance model is:

$$\boxed{\begin{array}{rcl} \overline{\varphi}_{WU} & = & \overline{\varphi}\left(\mathbf{x}_{WU}\right) = \varphi_0 + \mathbf{g}^T \cdot (\mathbf{x}_{WU} - \mathbf{x}_0) \\ & = & \varphi_0 + \beta_W \cdot \sqrt{\mathbf{g}^T \cdot \mathbf{C} \cdot \mathbf{g}} \\ & = & \varphi_0 + \beta_W \cdot \sigma_{\overline{\varphi}} \end{array}} \tag{99}$$

Simulation can be used to obtain the true performance value at $\mathbf{x}_{WU}$:

$$\varphi_{WU} = \varphi(\mathbf{x}_{WU}) \tag{100}$$

4.6 General worst-case analysis

The general worst-case analysis is based on

- a hyperellipsoid tolerance region $T_E = \left\{ \mathbf{x} \,\middle|\, (\mathbf{x}-\mathbf{x}_0)^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{x}-\mathbf{x}_0) \le \beta_W^2 \right\}$, and/or a hyperrectangle tolerance region $T_R = \{\mathbf{x} \,|\mathbf{x}_L \le \mathbf{x} \le \mathbf{x}_U\}$, for subsets of parameters, and
- a general nonlinear performance that can often only be evaluated point-wise by simulation $\mathbf{x} \mapsto \varphi$

In the following, we will treat the case of a tolerance hyperellipsoid to analyze how far the interpretation of β_W-design extends to the general case of nonlinear performance features.

Fig. 14 illustrates the situation for two parameters. The tolerance hyperellipsoid T_E and some of the level contours of the performance φ are shown. These contour lines are now nonlinear. The example function has curved contours fanning out with increasing performance values from the lower right to the upper left. Still, some contours touch the border of the tolerance hyperellipsoid and represent a lower and upper worst-case performance. And we can identify visually where the related worst-case parameter sets are. But the gradients in those boundary points differ from each other and the gradient at the nominal point. This leads to an individual linear performance model $\overline{\varphi}^{(WU)}$ and $\overline{\varphi}^{(WL)}$ in each worst-case point, which can only be determined **after** having computed the worst-case point.

The following will give the mathematical problem formulation of general worst-case analysis for the lower worst-case with tolerance hyperellipsoids.

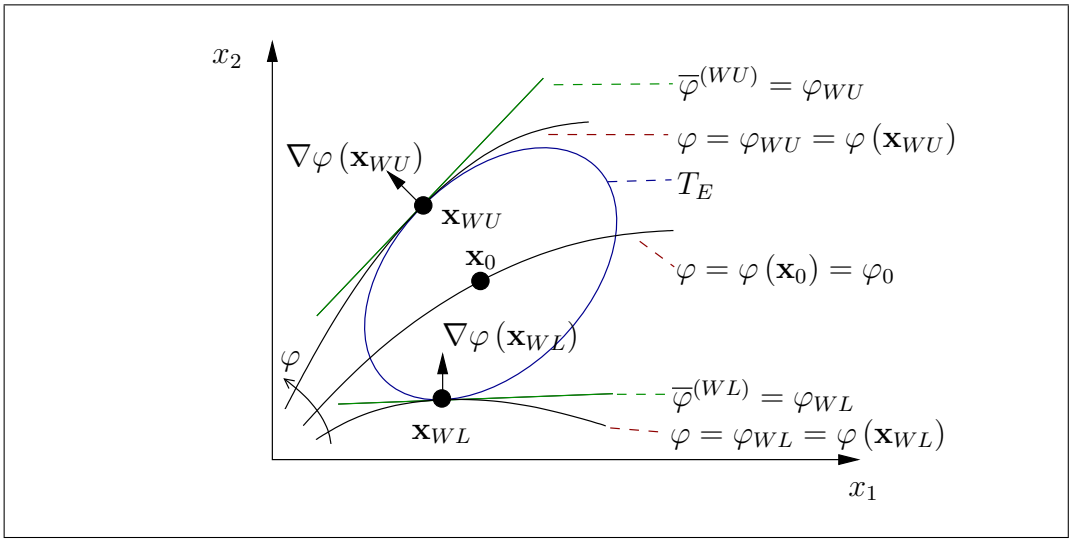


Figure 14. General worst-case analysis with tolerance ellipsoid and nonlinear performance function.

4.6.1 General worst-case analysis, lower worst case, tolerance hyperellipsoid

The lower worst case is the minimum value of the performance function over the tolerance hyperellipsoid, as in the realistic worst-case analysis. The difference is that now the full nonlinear performance is considered. This leads to a constrained minimization problem with the performance function as objective and the tolerance hyperellipsoid as a constraint.

$$\boxed{\min \varphi(\mathbf{x}) \quad \text{s.t.} \quad (\mathbf{x}-\mathbf{x}_0)^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{x}-\mathbf{x}_0) \leq \beta_W^2} \tag{101}$$

This is a particular nonlinear programming problem with a nonlinear objective function and one quadratic inequality constraint. This optimization problem cannot be solved analytically and requires an iterative solution process, as outlined in the beginning. A fundamental nonlinear optimization method is Sequential Quadratic Programming (SQP), whose problem formulation and prerequisites will be built up in the subsequent chapters 5 and 6. As a result of an SQP-based solution process, the gradient $\nabla f(\mathbf{x}_{WL})$ of the performance at the worst-case point is also available.

Please note that the solution may also be inside the tolerance hyperellipsoid, or more than one solution may exist. In integrated analog circuit design, two solutions are frequently encountered in so-called mismatch situations, e.g., of transistor pairs. And sometimes, the solution is inside T_E.

In the following, we will **assume that the solution is unique and on the border of** T_E and take a look at formulas that we can establish **based on the results of an iterative solution** of (101).

With the results of (101), a linear model at the worst-case point can be written:

$$\overline{\varphi}^{(WL)}(\mathbf{x}) = \varphi_{WL} + \nabla\varphi(\mathbf{x}_{WL})^T \cdot (\mathbf{x}-\mathbf{x}_{WL}) \tag{102}$$

Let us put the cart before the horse and insert this result of (101) into (101) itself:

$$\min \nabla\varphi(\mathbf{x}_{WL})^T \cdot \mathbf{x} \quad \text{s.t.} \quad (\mathbf{x}-\mathbf{x}_0)^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{x}-\mathbf{x}_0) \leq \beta_W^2 \tag{103}$$

We recognize that (103) is identical to the realistic worst-case analysis (88), only the gradient $\mathbf{g}$ in (88) has been replaced with the gradient $\nabla\varphi(\mathbf{x}_{WL})$ obtained after having solved (101). Therefore we can write the equation for the worst-case parameter vector of the general worst-case analysis by replacing $\mathbf{g}$ in the formula of the realistic worst-case analysis with $\nabla\varphi(\mathbf{x}_{WL})$. We obtain for the worst-case parameter vector:

$$\begin{aligned} \mathbf{x}_{WL}-\mathbf{x}_0 &= -\frac{\beta_W}{\sqrt{\nabla\varphi(\mathbf{x}_{WL})^T \cdot \mathbf{C} \cdot \nabla\varphi(\mathbf{x}_{WL})}} \cdot \mathbf{C} \cdot \nabla\varphi(\mathbf{x}_{WL}) \\ &= -\frac{\beta_W}{\sigma_{\overline{\varphi}(WL)}} \cdot \mathbf{C} \cdot \nabla\varphi(\mathbf{x}_{WL}) \end{aligned} \tag{104}$$

Please note that $\sigma_{\overline{\varphi}(WL)}$ is the **standard deviation of the linear performance model in the worst-case point** $\mathbf{x}_{WL}$.

To obtain a formula for the worst-case performance value, we first formulate the **performance value $\overline{\varphi}_0^{(WL)}$ at the nominal point, using the linear performance model in the worst-case**:

$$\begin{aligned} \overline{\varphi}^{(WL)}(\mathbf{x}_0) = \overline{\varphi}_0^{(WL)} &= \varphi_{WL} + \nabla\varphi(\mathbf{x}_{WL})^T \cdot (\mathbf{x}_0 - \mathbf{x}_{WL}) \\ \varphi_{WL} &= \overline{\varphi}_0^{(WL)} + \nabla\varphi(\mathbf{x}_{WL})^T \cdot (\mathbf{x}_{WL} - \mathbf{x}_0) \end{aligned} \tag{105}$$

Substituting (104) in (105) gives the following formula worst-case performance value:

$$\begin{aligned} \varphi_{WL} &= \overline{\varphi}_0^{(WL)} - \beta_W \cdot \sqrt{\nabla\varphi(\mathbf{x}_{WL})^T \cdot \mathbf{C} \cdot \nabla\varphi(\mathbf{x}_{WL})} \\ &= \overline{\varphi}_0^{(WL)} - \beta_W \cdot \sigma_{\overline{\varphi}^{(WL)}} \end{aligned} \tag{106}$$

Please be reminded that these formulas are based on the results of a general worst-case analysis after the iterative solution of the optimization problem.

(106) shows that the interpretation of β_W as worst-case distance and as β_W-design of realistic worst-case analysis from (96) is valid in general.

The main difference is that nominal performance and performance standard deviation are determined from a linear model in the worst-case parameter set.

As a side note: If we replace $\mathbf{x}_{WL}$ on the left side of (104) by $\mathbf{x}_{WL}^{(\kappa+1)}$ and $\nabla\varphi(\mathbf{x}_{WL})$ on the right side by $\nabla\varphi(\mathbf{x}_{WL}^{(\kappa)})$, we obtain an iteration formula for solving the general worst-case analysis problem. It would iteratively "shoot" from the nominal point with iteratively improved gradients until the final gradient in the worst-case point is reached. Please note that there is no investigation into when and how this iteration formula converges.

The general worst-case analysis is a nonlinear optimization problem that cannot be solved analytically but requires iterative approaches. The following two chapters 5 and 6 are dedicated to the presentation of "optimization in a nutshell" and to iterative approaches to nonlinear optimization problems, for instance, the general worst-case analysis and circuit optimization.

4.6.2 General worst-case analysis, upper worst case, tolerance hyperellisoid

For the upper worst case, the objective receives a minus sign in the formulas for the lower worst case. (101) becomes:

$$\min\ -\varphi\left(\mathbf{x}\right)\quad \text{s.t.}\quad \left(\mathbf{x}-\mathbf{x}_0\right)^T\cdot\mathbf{C}^{-1}\cdot\left(\mathbf{x}-\mathbf{x}_0\right)\leq\beta_W^2 \tag{107}$$

(102) becomes:

$$\overline{\varphi}^{(WU)}\left(\mathbf{x}\right)=\varphi_{WU}+\nabla\varphi\left(\mathbf{x}_{WU}\right)^T\cdot\left(\mathbf{x}-\mathbf{x}_{WU}\right) \tag{108}$$

(103) becomes:

$$\min\ -\nabla\varphi\left(\mathbf{x}_{WU}\right)^T\cdot\mathbf{x}\quad \text{s.t.}\quad \left(\mathbf{x}-\mathbf{x}_0\right)^T\cdot\mathbf{C}^{-1}\cdot\left(\mathbf{x}-\mathbf{x}_0\right)\leq\beta_W^2 \tag{109}$$

The worst-case parameter vector can be written as:

$$\begin{aligned}\mathbf{x}_{WU}-\mathbf{x}_0 &= +\frac{\beta_W}{\sqrt{\nabla\varphi(\mathbf{x}_{WU})^T\cdot\mathbf{C}\cdot\nabla\varphi(\mathbf{x}_{WU})}}\cdot\mathbf{C}\cdot\nabla\varphi\left(\mathbf{x}_{WU}\right)\\ &= +\frac{\beta_W}{\sigma_{\overline{\varphi}^{(WU)}}}\cdot\mathbf{C}\cdot\nabla\varphi\left(\mathbf{x}_{WU}\right)\end{aligned} \tag{110}$$

The worst-case performance value can be written as:

$$\begin{aligned}\varphi_{WU} &= \overline{\varphi}_0^{(WU)}+\beta_W\cdot\sqrt{\nabla\varphi\left(\mathbf{x}_{WU}\right)^T\cdot\mathbf{C}\cdot\nabla\varphi\left(\mathbf{x}_{WU}\right)}\\ &= \overline{\varphi}_0^{(WU)}+\beta_W\cdot\sigma_{\overline{\varphi}^{(WU)}}\end{aligned} \tag{111}$$

4.6.3 General worst-case analysis, tolerance hyperrectangle

The general worst-case analysis with tolerance hyperrectangle is written for the lower worst-case as:

$$\min\varphi\left(\mathbf{x}\right)\quad \text{s.t.}\quad \begin{cases}\mathbf{x}\geq\mathbf{x}_L\\ \mathbf{x}\leq\mathbf{x}_U\end{cases} \tag{112}$$

and for the upper worst-case as:

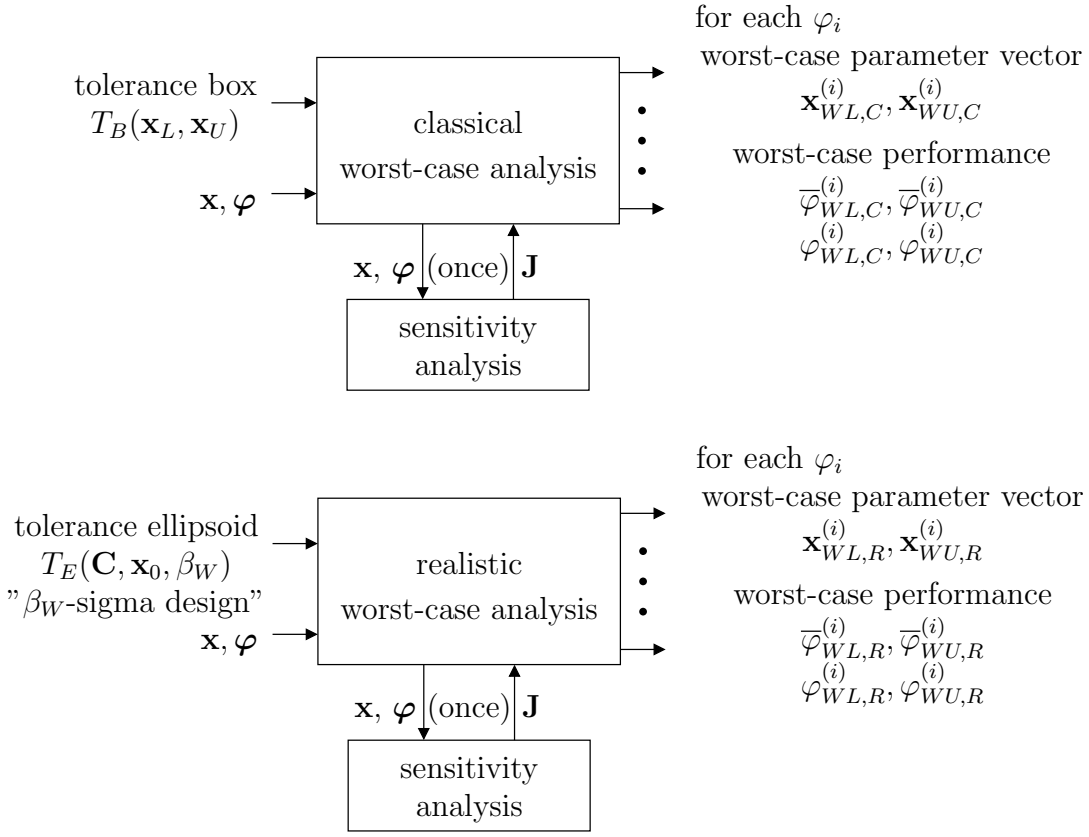
$$\min-\varphi\left(\mathbf{x}\right)\quad \text{s.t.}\quad \begin{cases}\mathbf{x}\geq\mathbf{x}_L\\ \mathbf{x}\leq\mathbf{x}_U\end{cases} \tag{113}$$

The combination of tolerance hyperrectangle and tolerance hyperellipsoid in the general worst-case analysis leads to a respective combination of constraints in the problem formulation.

4.7 Input/output of worst-case analysis

In the following, the three types of worst-case analysis are illustrated from a black-box point of view with their input, output, and calling of sensitivity analysis and simulation.

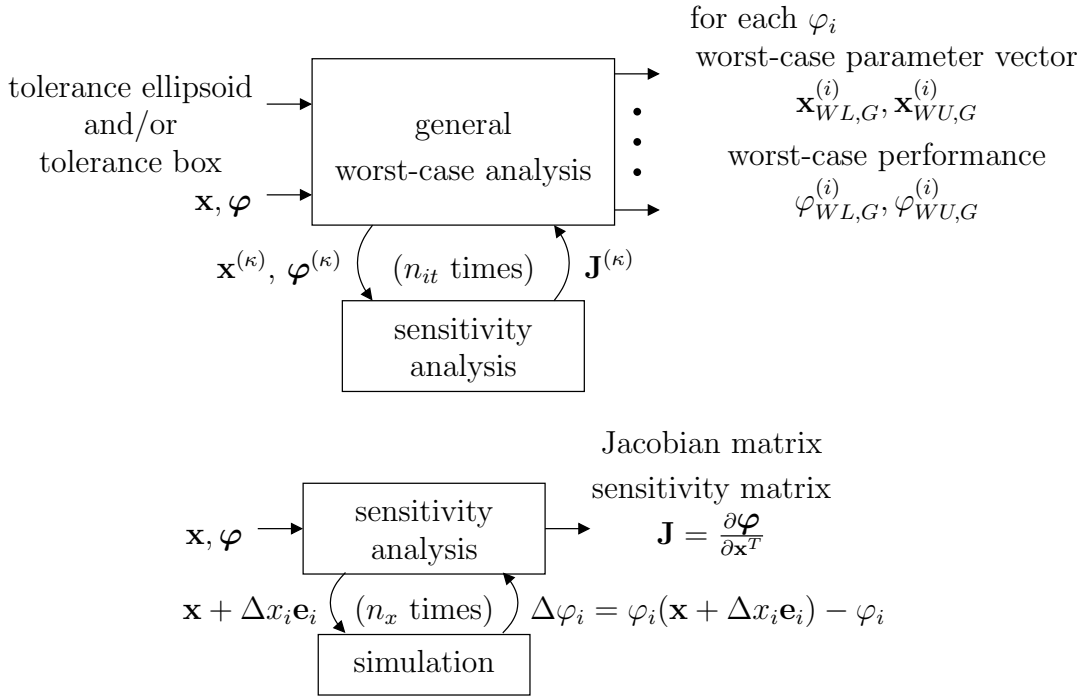
Classical and realistic worst-case analyses differ in their inputs; for classical, it is a hyperrectangle, and for realistic, it is a hyperellipsoid. Both need the set of parameters and performance features as input. Both call a sensitivity analysis **once** to get the gradient in the nominal point. Both have as output the lower and/or upper worst-case parameter sets and the lower and/or upper worst-case performance values. The latter are evaluated based on the linear performance model and/or a simulation in the worst-case parameter sets. This is done for each performance feature separately.



The general worst-case analysis has tolerance hyperellipsoids and/or tolerance hyperrectangles as input and provides the true worst-case parameter sets and true worst-case performance values as output. For this purpose, an iterative optimization process is performed, which calls sensitivity analysis in each iteration step.

A sensitivity analysis usually computes the gradients of all performance features by a finite difference approach. Each step alters one parameter and calls a simulation to compute all performance features for this one modified parameter. That means it calls simulation n_x times in addition to the simulation at the nominal point. The resulting sensitivity matrix is also called the Jacobian matrix.

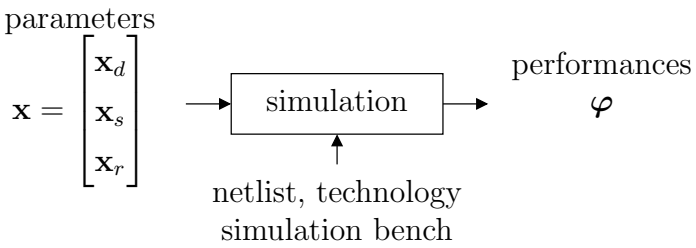
The Jacobian matrix, or sensitivity matrix, is rectangular; it has all the performance features in the rows and all the parameters in the columns. The element J_{ij} is the partial derivative of performance feature φ_i with regard to parameter x_j. A row $J_{i\cdot}$ contains the



gradient of performance feature φ_i with regard to all parameters. A column $J_{\cdot j}$ contains the partial derivatives of all performance features with regard to parameter x_j, or the delta in all performance features with regard to Δx_i.

$$\mathbf{J} = \begin{bmatrix} \frac{\partial\varphi_1}{\partial x_1} & \frac{\partial\varphi_1}{\partial x_2} & \cdots & \frac{\partial\varphi_1}{\partial x_{n_x}} \\ \frac{\partial\varphi_2}{\partial x_1} & \frac{\partial\varphi_2}{\partial x_2} & \cdots & \frac{\partial\varphi_2}{\partial x_{n_x}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\varphi_{n_\varphi}}{\partial x_1} & \frac{\partial\varphi_{n_\varphi}}{\partial x_2} & \cdots & \frac{\partial\varphi_{n_\varphi}}{\partial x_{n_x}} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_1^T = \frac{\partial\varphi_1}{\partial \mathbf{x}^T} \\ \mathbf{g}_2^T = \frac{\partial\varphi_2}{\partial \mathbf{x}^T} \\ \vdots \\ \mathbf{g}_{n_\varphi}^T = \frac{\partial\varphi_{n_\varphi}}{\partial \mathbf{x}^T} \end{bmatrix} \approx \begin{bmatrix} \cdots \frac{1}{\Delta x_i} \Delta\boldsymbol{\varphi}_i \cdots \end{bmatrix}$$

Simulation finally gets the parameter values as input and provides the performance values as output. Simulation is done based on the specific circuit netlist, the manufacturing process technology data, and a simulation bench with input stimuli.



5 Unconstrained optimization

The presentation starts at the innermost part of an iterative optimization process: the line search. Aspects of line search, as the requirement of a unimodal function, partitioning into bracketing and section, Wolfe-Powell conditions to control the bracketing and sectioning will be presented. Three sectioning approaches will be sketched: golden sectioning, quadratic model, and backtracking.

Next, multivariate optimization will be approached, first for unconstrained problems in this chapter. First, methods that do not require derivatives will be presented: coordinate search and the polytope method. Then, methods working on derivatives will be given. These are the steepest descent, Newton method, Quasi-Newton method, Levenberg Marquardt approach, least-squares approach, and conjugate-gradient approach.

5.1 Line search

The line search optimizes the objective function f over a single parameter. Therefore, it can be regarded as a univariate optimization problem. Line search starts within an iterative optimization process when a search direction has been determined, and the factor α on the length of the step along the search direction has to be determined. The function that is investigated is given by (23), (24) and (25).

Convergence

The convergence of an iterative optimization method has to be investigated, not just for the line search but for the overall procedure. The convergence is defined from the end, that is, an **error vector** is defined that denotes the difference between the current solution vector and the optimal solution:

$$\boldsymbol{\epsilon}^{(\kappa)} = \mathbf{x}^{(\kappa)} - \mathbf{x}^{\star} \tag{114}$$

The **global convergence** determines that the iterative process converges at all:

$$\lim_{\kappa \to \infty} \boldsymbol{\epsilon}^{(\kappa)} = \mathbf{0} \tag{115}$$

In addition, the speed of convergence is to be investigated. This is done through the **convergence rate**:

$$\left\| \boldsymbol{\epsilon}^{(\kappa+1)} \right\| = L \cdot \left\| \boldsymbol{\epsilon}^{(\kappa)} \right\|^{p} \tag{116}$$

Here it is assumed that the norm of the error vector is less than 1. p is the convergence rate, L is the convergence factor. The main convergence rates are linear and quadratic:

$$p = 1: \quad \text{linear convergence}, L < 1 \tag{117}$$

$$p = 2: \quad \text{quadratic convergence} \tag{118}$$

To illustrate the difference between linear and quadratic convergence, please see the following one-dimensional numerical example with $\epsilon^{(0)} = 0.1$:

	$p = 1,\ L = 0.5$	$p = 2,\ L = 1$
$\epsilon^{(1)}$	0.05	0.01
$\epsilon^{(2)}$	0.025	10^{-4}
$\epsilon^{(3)}$	0.0125	10^{-8}
$\epsilon^{(4)}$	0.00625	10^{-16}

We can see that floating point accuracy has been achieved within 4 iteration steps in the case of quadratic convergence, while linear convergence takes much longer.

In practice, non-integer convergence rates p are considered; these are called super-linear, i.e., better than linear, worse than quadratic.

Unimodal function

A line search shall be done over a start interval with exactly one minimum to obtain convergence. Such a function is defined as **unimodal** over an interval $[L, R]$ if there exists exactly one value $\alpha_{opt} \in [L, R]$ for which holds:

$$\underset{L<\alpha_1<\alpha_2<R}{\forall} \alpha_2 < \alpha_{opt} \rightarrow f(\alpha_1) > f(\alpha_2) \quad \text{(Fig. 15(a))} \tag{119}$$

$$\text{and} \quad \underset{L<\alpha_1<\alpha_2<R}{\forall} \alpha_1 > \alpha_{opt} \rightarrow f(\alpha_1) < f(\alpha_2) \quad \text{(Fig. 15(b))} \tag{120}$$

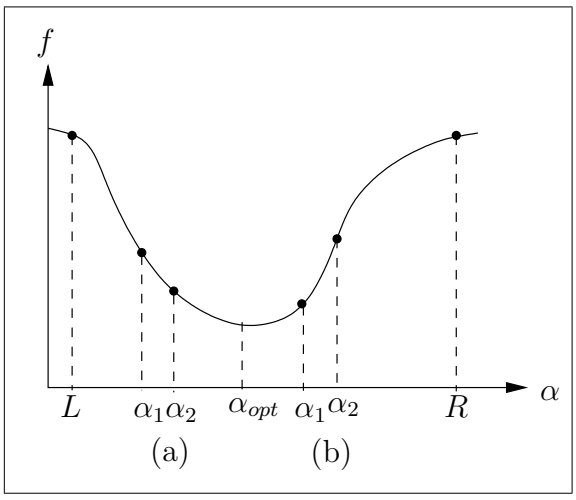


Figure 15. Unimodal function.

Please note that unimodality is a weaker requirement on a function than convexity (see App. H).

Bracketing, sectioning

A line search consists of two stages; these are bracketing and sectioning:

- Bracketing: Find an initial interval $\alpha \in [0, R]$ in which $f(\alpha)$ is unimodal.
- Sectioning: Iteratively reduce the size of the interval while keeping the objective minimum within the interval until one is close enough to the minimum.

The requirements on a line search in general are defined as finding an α that brings a

- sufficient reduction in f, and a
- big enough step length.

These criteria are used to define termination criteria for both the bracketing and the sectioning phase of a line search. They are formalized in the Wolfe-Powell conditions.

Please note that it is often more efficient to perform the line search only approximately.

Wolfe-Powell conditions

The Wolfe-Powell conditions consist of the following three conditions. They are illustrated in Fig. 16.

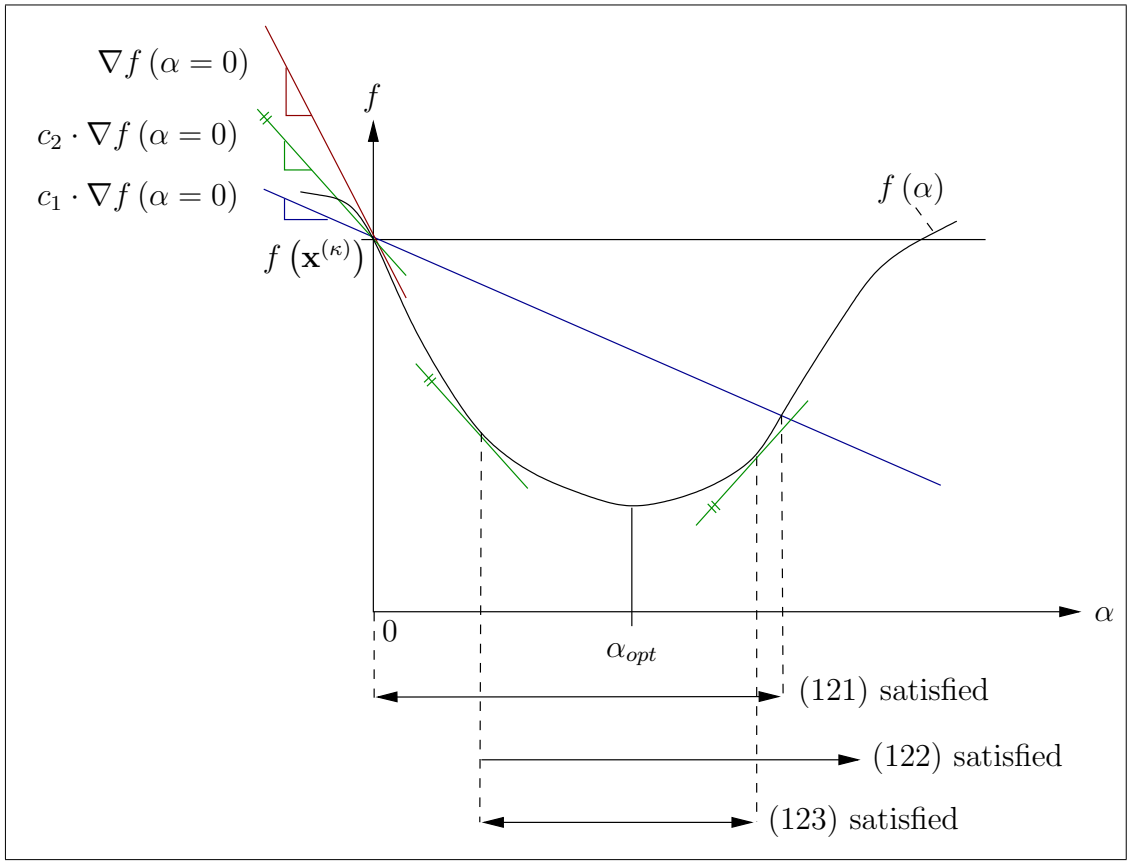


Figure 16. Illustration of Wolfe-Powell conditions.

- **Armijo condition**: The Armijo condition describes a sufficient objective reduction using the gradient at the current point, i.e., $\alpha = 0$, multiplied by some factor $0 < c_1 < 1$:

$$f(\alpha) \leq f(0) + \alpha \cdot c_1 \cdot \underbrace{\nabla f \left(\mathbf{x}^{(\kappa)}\right)^T \cdot \mathbf{r}}_{\nabla f(\alpha=0)} \tag{121}$$

  A unimodal function goes through a minimum before rising again. The Armijo condition defines a linearly decreasing function from the start point that should not be exceeded and hence defines a right interval bound on α where it is satisfied. It can be interpreted as requiring more function decrease for more step length.

- **Curvature condition**: The curvature condition describes a sufficient gradient increase in relation to the gradient at the current point, i.e., $\alpha = 0$, multiplied by some factor $c_1 < c_2 < 1$:

$$\underbrace{\nabla f \left(\mathbf{x}^{(\kappa)} + \alpha \cdot \mathbf{r}\right)^T \cdot \mathbf{r}}_{\nabla f(\alpha)} \geq c_2 \cdot \underbrace{\nabla f \left(\mathbf{x}^{(\kappa)}\right)^T \cdot \mathbf{r}}_{\nabla f(\alpha=0)} \tag{122}$$

  It uses the property that a unimodal function starts at $\alpha = 0$ with a negative slope which may first even become more negative before rising again. The curvature condition identifies this point of reaching the initial gradient times c_2 again and defines a left interval bound on α where it is satisfied.

- **strong curvature condition**: The strong curvature condition describes a more narrow interval around the minimum of the line search by defining an interval with a certain flatness of the gradient:

$$|\nabla f(\alpha)| \leq c_2 \cdot |\nabla f(\alpha = 0)| \tag{123}$$

$$0 < c_1 < c_2 < 1 \tag{124}$$

  It can be interpreted as a **relaxed first-order optimality condition** for a stationary point.

The Wolfe-Powell conditions are used to prove the convergence of a line search method.

5.2 Golden sectioning

As a side comment, please note that the root-finding of a univariate monotone function works with a start interval that includes the root, i.e., the point where the function sign switches. The sectioning then is a bi-sectioning and reducing the interval to either the left or right half, such that the switch of function sign is maintained in the reduced interval.

Unlike root finding, minimizing a unimodal univariate function requires two sampling points within the interval. This is due to the properties (119) and (120) of unimodality. Take a look at Fig. 17. Two additional functional evaluations, i.e., simulations, have been done inside the interval $[L, R]$ of width Δ. The minimum cannot be in the interval $[\alpha_2, R]$, because the function value was already rising from α_1 to α_2 and cannot go down again in a unimodal function. But it cannot be said if the minimum is in $[L, \alpha_1]$ or $[\alpha_1, \alpha_2]$. Therefore, the new reduced interval is $[L, \alpha_2]$ of width $\Delta^{(2)}$.

The question is whether we can reuse one of the two inner simulation points for the next sectioning step and only add one more to save simulation costs. And this should be possible for both alternatives of new intervals, $[L, \alpha_2]$ and $[\alpha_1, R]$. The approach is a sectioning parameter, τ, which is used to determine the sectioning intervals' ratio as shown in the first three rows under the coordinate system in Fig. 17.

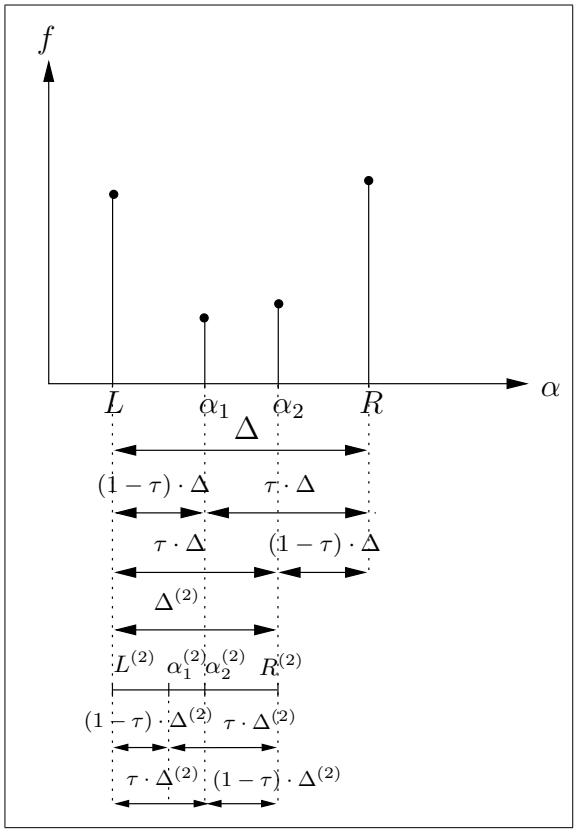


Figure 17. Two steps of Golden sectioning

For the new interval shown in row 4 under the coordinate system in Fig. 17, the same sectioning principle shall be applied, such that the old left inner point α_1 becomes the new right inner point $\alpha^{(2)}$. This is illustrated in rows 5 to 7 under the coordinate system in Fig. 17.

From the widths of the left sub-intervals in rows 3 and 4, we can write: $\Delta^{(2)} = \tau \cdot \Delta$. And from the widths of the left sub-intervals in rows 2 and 7, we can write $\tau \cdot \Delta^{(2)} = (1-\tau) \cdot \Delta$. Together this yields $\tau^2 + \tau - 1 = 0$ and a golden sectioning ratio of $\tau = \frac{-1+\sqrt{5}}{2} \approx 0.618$.

Another approach leads to the same result: the three right sub-intervals in rows 3 to 1 of Fig. 17 should relate equally to each other as: $\frac{(1-\tau)\cdot\Delta}{\tau\cdot\Delta} = \frac{\tau\cdot\Delta}{\Delta}$. This results in the same **golden sectioning ratio** τ:

$$\frac{1-\tau}{\tau} = \frac{\tau}{1} \iff \tau^2 + \tau - 1 = 0 : \tau = \frac{-1+\sqrt{5}}{2} \approx 0.618 \tag{125}$$

A pseudocode for the Golden sectioning method is given in the following:

```
/* left inner point */
α1 := L + (1 − τ) · |R − L|
/* right inner point */
α2 := L + τ · |R − L|
REPEAT
  IF f(α1) < f(α2)
    /* α* ∈ [L, α2], α1 is new right inner point */
    R := α2
    α2 := α1
    α1 := L + (1 − τ) · |R − L|
  ELSE
    /* α* ∈ [α1, R], α2 is new left inner point */
    L := α1
    α1 := α2
    α2 := L + τ · |R − L|
UNTIL Δ < Δmin
```

How many steps to get to given accuracy?

The convergence rate of Golden sectioning is linear with a convergence factor of τ because:

$$\Delta^{(\kappa+1)} = \tau \cdot \Delta^{(\kappa)} \tag{126}$$

How many steps are required to reduce a start interval of width $\Delta^{(0)}$ down to a specified interval width of $\Delta^{(\kappa)}$? For that purpose, we note that

$$\Delta^{(\kappa)} = \tau^{\kappa} \cdot \Delta^{(0)} \tag{127}$$

Solving for κ we obtain:

$$\kappa = \left\lceil -\frac{1}{\log \tau} \cdot \log \frac{\Delta^{(0)}}{\Delta^{(\kappa)}} \right\rceil \approx \left\lceil 4.78 \cdot \log \frac{\Delta^{(0)}}{\Delta^{(\kappa)}} \right\rceil \tag{128}$$

5.3 Line search by quadratic model

After performing bracketing and determining an initial interval with unimodal function behavior, the minimum in that interval can be approximated with the help of a quadratic function model. A quadratic model $m(\alpha)$

$$m(\alpha) = f_0 + g \cdot \alpha + \frac{1}{2} \cdot h \cdot \alpha^2 \tag{129}$$

can be computed with

- three sampling points, or
- first and second-order derivatives of the function.

Fig. 18 illustrates the quadratic modeling with three sampling points. α_t is determined based on the quadratic model $m(\alpha)$. The small circle indicates the true function value at α_t.

The minimum of the univariate quadratic model is determined by applying the optimality conditions on the quadratic model. This approach is called a univariate **Newton approach**. The first-order optimality condition yields:

$$\min m(\alpha) \rightarrow \nabla m(\alpha) = g + h \cdot \alpha_t \stackrel{!}{=} 0 \rightarrow \alpha_t = -g/h \tag{130}$$

with the second-order optimality condition:

$$h > 0 \tag{131}$$

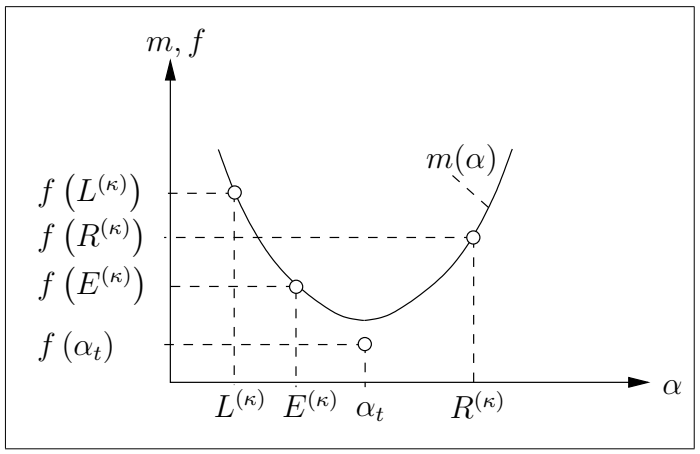


Figure 18. Quadratic model.

5.4 Backtracking line search

Another line search approach is backtracking. The principle is to start with a large step length value and reduce it iteratively until the Armijo condition is satisfied. That means we start with a too-long step length and pull it back till we are inside the upper interval end of the Wolfe-Powell conditions. A pseudo-code of backtracking would be:

$0 < c_3 < 1$
determine step length α_t /* big enough, not too big */
WHILE α_t violates (121)
[$\alpha_t := c_3 \cdot \alpha_t$

5.5 Coordinate search for multivariate unconstrained optimization without derivatives

We are now moving forward to optimization over more than one variable. For the moment, we are looking at methods without derivatives.

What would you do if you want to optimize over several variables, but all you have is the line search we just discussed? Well, you do a line search in each of the variables. This is called coordinate search.

The next question is whether you are ready once you review all coordinates. The answer is no because a line search in one variable was performed for specific values of the other variables. After you change one of them, the line search has to be done again for the other parameters. This must be repeated until there is no more change in all coordinates.

A pseudo-code for coordinate search, therefore, looks like this:

REPEAT
  FOR $j = 1, \cdots, n_x$
    $\alpha_t := \arg\min_{\alpha} f(\mathbf{x} + \alpha \cdot \mathbf{e}_j)$
    $\mathbf{x} := \mathbf{x} + \alpha_t \cdot \mathbf{e}_j$
UNTIL convergence or maximum allowed iterations reached

$$\mathbf{e}_j = [0 \ldots 0 \quad \underset{\uparrow \text{ j-th position}}{1} \quad 0 \ldots 0]^T$$

An illustration of coordinate search is given in the top part of Fig. 19.

In this example's first run of the repeat loop, the y-coordinate is run through a line search. We assume an exact line search and a quadratic objective function. Then, the level contour of the objective function with the smallest radius that can be reached on the vertical line shown is looked for. This is a point where the vertical line touches an ellipsis of the objective function. Next, the x-coordinate is searched through exactly, yielding the next touching point of this horizontal line with an ellipsis. This completes the first run. It can be seen that the repeat loop has to be performed many times. The resulting search trajectory is "zig-zagging" towards the optimum.

However, the steepest descent approach is not much better. This is illustrated in the lower part of Fig. 19. Using the gradient as a search direction instead of a coordinate axis, we see search lines perpendicular to the level contours. The exact line search goes along the steepest descent till it touches an ellipsis as the level contour of the objective function. We can see a similarly zig-zagging search trajectory for the steepest descent approach. But in each iteration step, we usually have to compute the gradient based on finite difference approximation, which brings extra simulation effort compared to coordinate search.

Please note that the steepest descent is never the method of choice for the search direction, only for the first step. We will soon see how to do better.

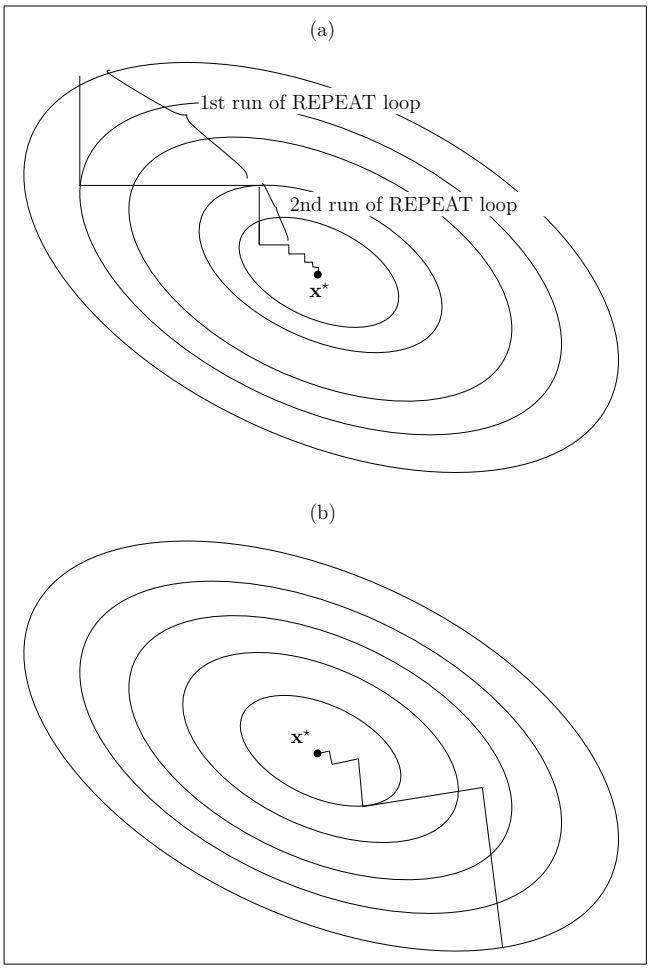


Figure 19. Coordinate search with exact line search for quadratic objective function (a). Steepest-descent method for quadratic objective function (b).

Please also note that coordinate search becomes faster the more "loosely coupled" or "uncorrelated" the variables are, i.e., the stronger the diagonal dominance of the Hessian matrix is. If the Hessian is even a diagonal matrix, then a coordinate search with an exact line search may ideally need one run of the repeat loop, as illustrated in Fig. 20.

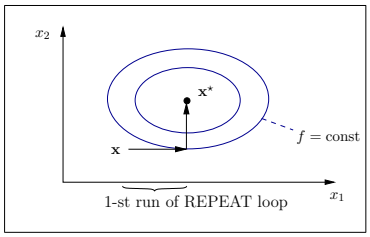


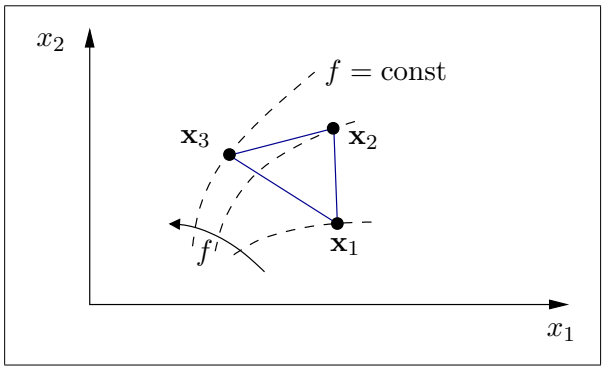
Figure 20. Coordinate search with exact line search for quadratic objective function with diagonal Hessian matrix.

5.6 Polytope method (Nelder-Mead simplex method) for multivariate unconstrained optimization without derivatives

The polytope method is another method for multivariate unconstrained optimization that works without derivatives. It is also called the Nelder-Mead simplex method to correspond to its creators. The term simplex stems from its initial geometric structure and must not be confused with the Simplex algorithm for linear programming, an entirely different problem and solution area.

We will see that the polotype method is straightforward to implement. It may be very effective and efficient. But it is not an optimization method of the mathematical canon, as there is no proof of convergence nor any theory of convergence rate.

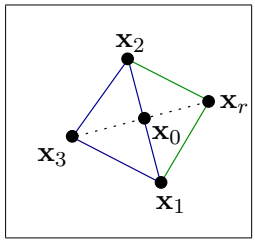
The idea of the polytope method is to create a linear model of the objective with an n_x-simplex in the n_x-dimensional variable space. This is illustrated in the following figure for two parameters.



An n_x-simplex is the convex hull of a set of parameter vector vertices $\mathbf{x}_i, i = 1, \cdots, n_x+1$. The vertices are ordered such that $f_i = f(\mathbf{x}_i)$, $f_1 \leq f_2 \leq \cdots \leq f_{n_x+1}$.

The idea is to move the simplex towards smaller objective values in the easiest way by keeping its form and changing one point. With regard to the simplex, this means to flip the point with the highest objective value, i.e., $\mathbf{x}_{n_x+1}$ on the other side of the $n_x - 1$ -dimensional simplex of the remaining points. This is done by the reflection of $\mathbf{x}_{n_x+1}$ through the center of gravity of the simplex of the remaining points:

Reflection



$$\mathbf{x}_0 = \frac{1}{n_x} \cdot \sum_{i=1}^{n_x} \mathbf{x}_i \tag{132}$$

$$\mathbf{x}_r = \mathbf{x}_0 + \rho \cdot (\mathbf{x}_0 - \mathbf{x}_{n_x+1}) \tag{133}$$

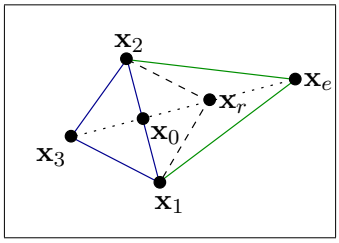
ρ is called the reflection coefficient $\rho > 0$, its default value is $\rho = 1$.

This basic procedure is repeated after reordering the simplex points according to the order of their objective values. One can also interpret the proceeding in that the respective point of the largest objective value iteratively tunnels through the rest simplex on the other side to minimize the maximum value of the objective values over the simplex points.

Several refinements are added to the primary procedure. These refinements will be defined first. After that, the overall procedure of the polytope method will be given. Please note that all refinement steps change the shape of the simplex to a general polytope.

If the reflection appears promising, an expansion is considered in the current iteration step. This means that the simplex corner point is not just mirrored on the other side of the simplex, but it is expanded:

Expansion

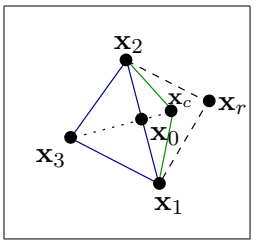


$$\mathbf{x}_e = \mathbf{x}_0 + \chi \cdot (\mathbf{x}_r - \mathbf{x}_0) \tag{134}$$

χ is called the expansion coefficient, with $\chi > 1$ and$\chi > \rho$. The default value is $\chi = 2$.

If, on the other hand, the reflection does not look promising, an outer contraction of the reflection is considered. That means the reflection is maintained but the steplength is contracted.

Outer contraction



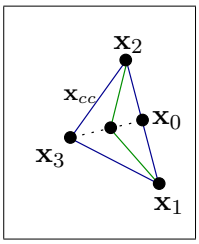
$$\mathbf{x}_c = \mathbf{x}_0 + \gamma \cdot (\mathbf{x}_r - \mathbf{x}_0) \tag{135}$$

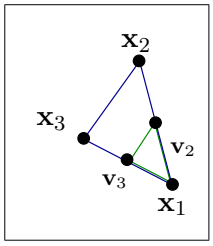
γ is the contraction coefficient, with $0 < \gamma < 1$. The default value is $\gamma = \frac{1}{2}$.

If the reflection looks even less promising, an inner contraction is considered. Here, the step does not even flip the polytope, but it rather reduces the size by moving the point with the largest objective value closer to the center of gravity of the rest of the polytope:

Inner contraction

$$\mathbf{x}_{cc} = \mathbf{x}_0 - \gamma \cdot (\mathbf{x}_0 - \mathbf{x}_{n_x+1}) \tag{136}$$





If this does not help either, then a reduction of the size of the polytope is made by moving all but the polytope corner with the smallest objective value closer to it:

Reduction (shrink)

$$\mathbf{v}_i = \mathbf{x}_1 + \sigma \cdot (\mathbf{x}_i - \mathbf{x}_1)\,, i = 2, \cdots, n_x + 1 \tag{137}$$

σ is called the reduction coefficient, with $0 < \sigma < 1$. The default value is $\sigma = \frac{1}{2}$.

These ingredients are combined to the following overall polytope method:

Cases of an iteration step of the polytope method

<table>
<tr><td colspan="6">(re-)order f_i; reflection</td></tr>
<tr><td colspan="2">$f_r < f_1$</td><td>$f_1 \le f_r < f_{n_x}$</td><td colspan="2">$f_{n_x} \le f_r < f_{n_x+1}$</td><td>$f_{n_x+1} \le f_r$</td></tr>
<tr><td colspan="2">expansion</td><td rowspan="4">insert $\mathbf{x}_r$
delete $\mathbf{x}_{n+1}$</td><td colspan="2">outer contraction</td><td>inner contraction</td></tr>
<tr><td colspan="2">$f_e < f_r$?</td><td colspan="2">$f_c \le f_r$?</td><td>$f_{cc} < f_{n_x+1}$?</td></tr>
<tr><td>yes</td><td>no</td><td>yes</td><td>no</td><td>no</td><td>yes</td></tr>
<tr><td>$\mathbf{x}_{n_x+1} = \mathbf{x}_e$</td><td>$\mathbf{x}_{n_x+1} = \mathbf{x}_r$</td><td>$\mathbf{x}_{n_x+1} = \mathbf{x}_c$</td><td colspan="2">reduction
$\mathbf{x}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{n_x+1}$</td><td>$\mathbf{x}_{n_x+1} = \mathbf{x}_{cc}$</td></tr>
</table>

The case distinction depends on comparing the objective value at the intermediate reflection point with the objective values of all polytope corners. An expansion is tried if it is smaller than at any polytope corner. If the objective value is then even better, expansion is done, else we stay with regular reflection. If the objective value at the reflection point is smaller than the largest value at the polytope corners, then outer contraction is tried. If this brings improvement compared to the reflection point, it is done, else a shrink is performed. Analogously, inner contraction is tested if the new objective value at the reflection point is larger than at any of the polytope corners. If this brings a better objective value than at the worst polytope corner, it is done, else a reduction is performed.

By the reduction and the other steps that decrease the size of the polytope, we are reducing the step size and increasing the solution's accuracy. Depending on the problem, the simplex's initial size must be determined before starting the algorithm.

5.7 Newton approach for multivariate unconstrained optimization with derivatives

Now we move forward to optimization methods for unconstrained optimization that use derivatives. The method using the first derivative, i.e., the steepest descent approach, has already been discussed. We switch to the approach using the quadratic objective model, which is called **Newton approach for a search direction**. It is based on a second-order model of the objective function based on the Taylor series given in (19) and (20), truncated after the second-order derivative, which gives the quadratic model $m^{(\kappa)}(\mathbf{r})$. Applying the first-order optimality condition to (19) to the minimization of this quadratic model, i.e., $\nabla m^{(\kappa)}(\mathbf{r}) = \mathbf{0}$, yields:

$$\boxed{\mathbf{H}^{(\kappa)} \cdot \mathbf{r} = -\mathbf{g}^{(\kappa)} \quad \rightarrow \quad \mathbf{r}^{(\kappa)}} \tag{138}$$

$\mathbf{r}^{(\kappa)}$ is obtained through the solution of a system of linear equations, and it represents the search direction for a subsequent line search.

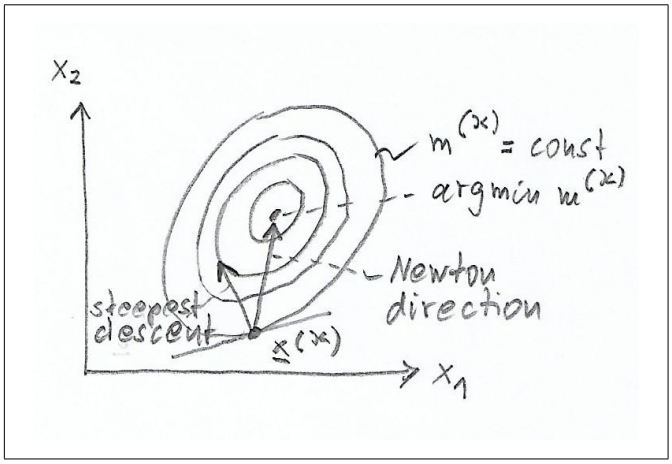


Figure 21. Steepest descent direction and Newton direction of quadratic model at $\mathbf{x}^{(\kappa)}$

Fig. 21 illustrates the Newton direction at $\mathbf{x}^{(\kappa)}$. While the steepest descent points perpendicular to the level contour at that point, the Newton direction points directly to the minimum of the quadratic model.

Question: Why must we add a line search, as the Newton direction points directly to the minimum of $m^{(\kappa)}$? Answer: Because it is only a model. This model is valid in a certain environment of $\mathbf{r}^{(\kappa)}$ but not at the minimum it suggests. It is much better than a linear model but still not capturing the full nonlinearity. Hence, a subsequent line search is required.

According to the sufficient second-order optimality condition for the stationary point of the model function $m^{(\kappa)}$, the second-order derivative has to be positive definite:

$$\boxed{\nabla^2 m(\mathbf{r}) = \mathbf{H}^{(\kappa)} \text{ is positive definite}} \tag{139}$$

Second-order optimality condition not fulfilled

The next question arises: What can we do if the second-order optimality condition is not fulfilled? There are three basic ways to deal with this:

- The first approach is to give up on the quadratic model and fall back on steepest descent $\mathbf{r}^{(\kappa)} = -\mathbf{g}^{(\kappa)}$.
- The second approach is to compute an eigenvalue decomposition of the Hessian matrix $\mathbf{H}^{(\kappa)}$ and switch the signs of the negative eigenvalues. Then, the stationary point is a minimum of the model, but the model does not properly reproduce the real objective. The idea behind this approach is that usually, a small number of eigenvalues is negative compared to the dimension of the variable space. For all the positive eigenvalues, the model function correctly reproduces the behavior of the objective. The Hessian then has a certain relevance that is used by this approach.
- The third approach is less "rude". It keeps better the relation among the eigenvalues and thus the reproduction of the true objective by adding a constant value λ to all diagonal elements of the Hessian matrix. This is called the **Levenberg-Marquardt** approach: $\left(\mathbf{H}^{(\kappa)} + \lambda \cdot \mathbf{I}\right) \cdot \mathbf{r} = -\mathbf{g}^{(\kappa)}$. The background of the Levenberg-Marquardt approach is the rule that a diagonal-dominant matrix is positive definite. Diagonal dominance means that the absolute value of each diagonal element is greater than or equal to the sum of absolute values of non-diagonal elements in the same row or column. The matrix will eventually become diagonal dominant and positive definite by adding the same value to all diagonal elements. This is similar to increasing all eigenvalues until they are all positive and, at the same time, keep the relation between them. Please note that the Levenberg-Marquardt approach is a homotopy method between the steepest descent ($\lambda \to \infty$) and Newton direction ($\lambda = 0$).

5.8 Quasi-Newton approach for multivariate unconstrained optimization with derivatives

The second question concerning the Newton approach is what we do if no Hessian matrix is available. This is typically the case in many technical problems with numerical simulation, where already the gradient has to be approximated with finite differences, not to mention the second-order derivative.

The good news is that there are methods to approximate the Hessian from the different gradients over the iterative steps of the optimization process. These gradients are used to build up information on the curvature of the function.

For instance, it is enough to have two gradients to establish the quadratic model in a one-dimensional space. In a high-dimensional space, it is more complicated. Each new gradient spans another direction in the space, and depending on the different gradient directions, quadratic information is built up in the subspaces of these gradients.

This approximated Hessian matrix is used to determine quasi a Newton direction, hence the name **quasi-Newton approach**.

The approximation of the Hessian matrix is denoted as $\mathbf{B}$. It is usually initialized with the identity matrix: $\mathbf{B}^{(0)} = \mathbf{I}$.

Using the quadratic model in iteration step κ, i.e. (19) and (20), the second derivative in the next iteration $\kappa + 1$ can be approximated by:

$$\nabla f\left(\mathbf{x}^{(\kappa+1)}\right) = \mathbf{g}^{(\kappa+1)} \approx \mathbf{g}^{(\kappa)} + \mathbf{H}^{(\kappa)} \cdot \alpha^{(\kappa)} \cdot \mathbf{r}^{(\kappa)} = \mathbf{g}^{(\kappa)} + \mathbf{H}^{(\kappa)} \cdot \left(\mathbf{x}^{(\kappa+1)} - \mathbf{x}^{(\kappa)}\right) \tag{140}$$

From this the **quasi-Newton condition** for an approximation $\mathbf{B}$ of the Hessian matrix is derived:

$$\boxed{\underbrace{\mathbf{g}^{(\kappa+1)} - \mathbf{g}^{(\kappa)}}_{\mathbf{y}^{(\kappa)}} = \mathbf{B}^{(\kappa+1)} \cdot (\underbrace{\mathbf{x}^{(\kappa+1)} - \mathbf{x}^{(\kappa)}}_{\mathbf{s}^{(\kappa)}})} \tag{141}$$

(141) describes an implicit relation between the difference in two subsequent gradients, the difference in two subsequent solution points, and the Hessian matrix approximation. This condition can be used to design update formulas for the approximation $\mathbf{B}^{(\kappa+1)}$.

A simple example of an update formula is the following:

Symmetric rank-1 update (SR1)

The approach starts from the following formula:

$$\mathbf{B}^{(\kappa+1)} = \mathbf{B}^{(\kappa)} + \mathbf{u} \cdot \mathbf{v}^T \tag{142}$$

The available new gradient is a vector that will be incorporated into the two vectors $\mathbf{u}$ and $\mathbf{v}$ of (142). A dyadic product of two vectors as in (142) is of rank one, hence the

name of the approach. (142) is inserted in the quasi-Newton condition (141), which leads to a formula for the vector $\mathbf{u}$:

$$\begin{aligned}\left(\mathbf{B}^{(\kappa)} + \mathbf{u} \cdot \mathbf{v}^T\right) \cdot \mathbf{s}^{(\kappa)} &= \mathbf{y}^{(\kappa)} \\ \mathbf{u} \cdot \mathbf{v}^T \cdot \mathbf{s}^{(\kappa)} &= \mathbf{y}^{(\kappa)} - \mathbf{B}^{(\kappa)} \cdot \mathbf{s}^{(\kappa)} \\ \mathbf{u} &= \frac{\mathbf{y}^{(\kappa)} - \mathbf{B}^{(\kappa)} \cdot \mathbf{s}^{(\kappa)}}{\mathbf{v}^T \cdot \mathbf{s}^{(\kappa)}}\end{aligned} \tag{143}$$

Here, the dyadic product has been dissolved into a scalar product. The correctness results from inserting (143) in the previous formula. We now substitute (143) in (142),

$$\mathbf{B}^{(\kappa+1)} = \mathbf{B}^{(\kappa)} + \frac{\left(\mathbf{y}^{(\kappa)} - \mathbf{B}^{(\kappa)} \cdot \mathbf{s}^{(\kappa)}\right) \cdot \mathbf{v}^T}{\mathbf{v}^T \cdot \mathbf{s}^{(\kappa)}} \tag{144}$$

and, due to the symmetry requirement of a Hessian matrix, we obtain the SR1-update formula:

$$\mathbf{B}^{(\kappa+1)} = \mathbf{B}^{(\kappa)} + \frac{\left(\mathbf{y}^{(\kappa)} - \mathbf{B}^{(\kappa)} \cdot \mathbf{s}^{(\kappa)}\right) \cdot \left(\mathbf{y}^{(\kappa)} - \mathbf{B}^{(\kappa)} \cdot \mathbf{s}^{(\kappa)}\right)^T}{\left(\mathbf{y}^{(\kappa)} - \mathbf{B}^{(\kappa)} \cdot \mathbf{s}^{(\kappa)}\right)^T \cdot \mathbf{s}^{(\kappa)}} \tag{145}$$

In case that $\mathbf{y}^{(\kappa)} - \mathbf{B}^{(\kappa)} \cdot \mathbf{s}^{(\kappa)} = \mathbf{0}$, the update would be skipped: $\mathbf{B}^{(\kappa+1)} = \mathbf{B}^{(\kappa)}$.

Please note that the SR1 update does not guarantee positive definiteness of the Hessian approximation $\mathbf{B}$. Well-known alternatives that maintain positive definiteness are, e.g., Davidon-Fletcher-Powell (DFP), Broyden-Fletcher-Goldfarb-Shanno (BFGS). BFGS update is a frequent method for quasi-Newton approaches.

As the computation of the Newton direction requires the solution of a linear equation system, often an approximation of the inverse $\mathbf{B}^{-1}$ instead of $\mathbf{B}$ or an approximation of a decomposition of the system matrix of linear equation system (138) is done, to facilitate the computation of the search direction $\mathbf{r}^{(\kappa)}$.

Please also note that if the vectors $\mathbf{s}^{(\kappa)}, \kappa = 1, \cdots, n_x$ are linear independent and if f would be a quadratic function with Hessian matrix $\mathbf{H}$, the quasi-Newton with SR1 update would terminate after not more than $n_x + 1$ steps with $\mathbf{B}^{(\kappa+1)} = \mathbf{H}$.

5.9 Levenberg-Marquardt approach (Newton direction plus trust region) for multivariate unconstrained optimization with derivatives

This method has a constraint, but it is one from the algorithm, not the technical problem. Therefore we prefer to place it under unconstrained optimization. The idea is to include a trust region in the computation of the search direction, which limits its length:

$$\boxed{\min m\left(\mathbf{r}\right) \quad \text{s.t.} \quad \underbrace{\left\|\mathbf{r}\right\|^2}_{\mathbf{r}^T\cdot\mathbf{r}} \leq \Delta} \tag{146}$$

The constraint is a trust region of the model $m\left(\mathbf{r}\right)$, e.g., with regard to the validity of the quadratic model of the objective or with regard to the positive definitiveness of the Hessian matrix.

The corresponding Lagrange function using the quadratic model of the objective is:

$$\mathcal{L}\left(\mathbf{r},\lambda\right) = f^{(\kappa)} + \mathbf{g}^{(\kappa)T}\cdot\mathbf{r} + \frac{1}{2}\mathbf{r}^T\cdot\mathbf{H}^{(\kappa)}\cdot\mathbf{r} - \lambda\cdot\left(\Delta - \mathbf{r}^T\cdot\mathbf{r}\right) \tag{147}$$

The first-order optimality condition for a stationary point $\nabla\mathcal{L}\left(\mathbf{r}\right) = 0$ leads to:

$$\boxed{\left(\mathbf{H}^{(\kappa)} + 2\cdot\lambda\cdot\mathbf{I}\right)\cdot\mathbf{r} = -\mathbf{g}^{(\kappa)}} \tag{148}$$

Please note that this formula is (aside from factor 2) identical to the form derived for indefinite Hessian matrices of the Newton approach. The motivation was from the Hessian matrix's positive definiteness through the diagonal dominance of the matrix. The motivation is from a trust region constraint in formulating the Newton approach.

5.10 Least-squares (plus trust-region) approach for multivariate unconstrained optimization with derivatives

We have so far dealt with scalar optimization, i.e., optimization of a single objective. Technical problems usually feature many objectives, not just one, resulting in a so-called multi-objective or multi-criteria optimization problem (MOO, MCO). In addition, the objectives are competing with each other, making optimal trade-offs necessary. Finding the set of possible trade-offs is called Pareto optimization.

A common approach to combine several objectives into a scalar cost function is the **least-squares approach**. A target value is specified for each objective. Then, the differences between the objective and target values are squared and summed up in a scalar cost function. If we then add a trust-region constraint on the length Δ of the search direction to be computed, we obtain:

$$\boxed{\min_{\mathbf{r}} \|\mathbf{f}(\mathbf{r}) - \mathbf{f}_{target}\|^2 \text{ s.t. } \|\mathbf{r}\|^2 \leq \Delta} \tag{149}$$

This is an easily implementable scalarization approach, as reasonable optimization targets can be specified in technical applications. If all targets are reached at the end of the optimization process, they can be pulled tighter, and the process can be restarted to see if further improvement is achievable.

The next important ingredient is to create a **linear model of the objective**:

$$\mathbf{f}(\mathbf{r}) \approx \overline{\mathbf{f}}^{(\kappa)}(\mathbf{r}) = \underbrace{\mathbf{f}\left(\mathbf{x}^{(\kappa)}\right)}_{\mathbf{f}_0^{(\kappa)}} + \underbrace{\nabla\mathbf{f}\left(\mathbf{x}^{(\kappa)^T}\right)}_{\mathbf{S}^{(\kappa)}} \cdot \mathbf{r} \tag{150}$$

We introduce an error vector to denote the difference between the objective and its target vector, and we insert the linear model of the objective function while leaving out the index (κ) to obtain the following:

$$\begin{aligned} \|\overline{\boldsymbol{\epsilon}}(\mathbf{r})\|^2 &= \|\overline{\mathbf{f}}(\mathbf{r}) - \mathbf{f}_{target}\|^2 && (151)\\ &= \overline{\boldsymbol{\epsilon}}^T(\mathbf{r}) \cdot \overline{\boldsymbol{\epsilon}}(\mathbf{r}) = (\underbrace{\mathbf{f}_0 - \mathbf{f}_{target}}_{\boldsymbol{\epsilon}_0} + \mathbf{S}\cdot\mathbf{r})^T \cdot (\boldsymbol{\epsilon}_0 + \mathbf{S}\cdot\mathbf{r}) && (152)\\ &= \boldsymbol{\epsilon}_0^T \cdot \boldsymbol{\epsilon}_0 + 2\cdot\mathbf{r}^T\cdot\mathbf{S}^T\cdot\boldsymbol{\epsilon}_0 + \mathbf{r}^T\cdot\mathbf{S}^T\cdot\mathbf{S}\cdot\mathbf{r} && (153) \end{aligned}$$

The optimization problem then is:

$$\boxed{\min_{\mathbf{r}} \|\overline{\boldsymbol{\epsilon}}(\mathbf{r})\|^2 \text{ s.t. } \|\mathbf{r}\|^2 \leq \Delta} \tag{154}$$

Please note that $\mathbf{S}^T\cdot\mathbf{S}$ is a part of the second derivative of $\|\boldsymbol{\epsilon}(\mathbf{r})\|^2$. It results from squaring a linear function.

The corresponding Lagrange function is:

$$\mathcal{L}(\mathbf{r}, \lambda) = \boldsymbol{\epsilon}_0^T \cdot \boldsymbol{\epsilon}_0 + 2 \cdot \mathbf{r}^T \cdot \mathbf{S}^T \cdot \boldsymbol{\epsilon}_0 + \mathbf{r}^T \cdot \mathbf{S}^T \cdot \mathbf{S} \cdot \mathbf{r} - \lambda \cdot \left(\Delta - \mathbf{r}^T \cdot \mathbf{r}\right) \tag{155}$$

It has a stationary point $\nabla \mathcal{L}(\mathbf{r}) = 0$ at

$$2 \cdot \mathbf{S}^T \cdot \boldsymbol{\epsilon}_0 + 2 \cdot \mathbf{S}^T \cdot \mathbf{S} \cdot \mathbf{r} + 2 \cdot \lambda \cdot \mathbf{r} = \mathbf{0} \tag{156}$$

which is obtained by solving the following equation system:

$$\boxed{\left(\mathbf{S}^T \cdot \mathbf{S} + \lambda \cdot \mathbf{I}\right) \cdot \mathbf{r} = -\mathbf{S}^T \cdot \boldsymbol{\epsilon}_0} \tag{157}$$

Please note the structural similarity to the Levenberg-Marquardt approach (148). We can identify a quadratic component on the left side, which was the Hessian in the Levenberg-Marquardt approach and is a part of the Hessian in the least-squares approach, and a gradient component on the right side.

If the trust region is skipped, i.e., $\lambda = 0$, we obtain the so-called **Gauss-Newton** method:

$$\boxed{\mathbf{S}^T \cdot \mathbf{S} \cdot \mathbf{r} = -\mathbf{S}^T \cdot \boldsymbol{\epsilon}_0} \tag{158}$$

Using a least-squares objective formulation, the Gauss-Newton method creates a Newton approach for a linear objective function. It is a popular approach to benefit from the Newton approach if the function model originally is just linear.

Characteristic boundary curve

The solution of the equation system (157) yields a search direction $\mathbf{r}^{(\kappa)}$ for a given factor λ.

Different from the usual search direction and line search partitioning, a trust-region approach for an iterative optimization step can be established by sweeping the factor λ in (157).

For each λ with $0 \leq \lambda < \infty$, we obtain a search direction of length $\|\mathbf{r}^{(\kappa)}\|$ and a corresponding cost function value, specifically, the linearized least-squares error between original objectives and their targets $\|\overline{\boldsymbol{\epsilon}}^{(\kappa)}\left(\mathbf{r}^{(\kappa)}\right)\|$.

The set of different solutions is presented in the form of a Pareto curve, e.g., as in Fig. 22.

The x-axis in Fig. 22 gives the length $\|\mathbf{r}^{(\kappa)}\|$, please note that the directions $\mathbf{r}^{(\kappa)}$ themselves are all different (see Fig. 23). The y-axis gives the achievable cost function $\|\overline{\boldsymbol{\epsilon}}^{(\kappa)}\left(\mathbf{r}^{(\kappa)}\right)\|$ with linearized objective $\overline{f}$ for that respective search step $\mathbf{r}^{(\kappa)}$.

The initial error value is given for $\lambda \to \infty$, as a zero step results. For $\lambda = 0$, no trust region constraint is active, and the complete step to reach the minimum cost function is taken. As the original objective model in (154) is linear, the target value can be achieved such that the cost function reaches zero.

Please note that the linear model is only valid in certain surroundings of the current iteration point. Therefore, the accuracy will deteriorate with increasing step length, and the Pareto front based on the linearized objective model will become wrong. This is

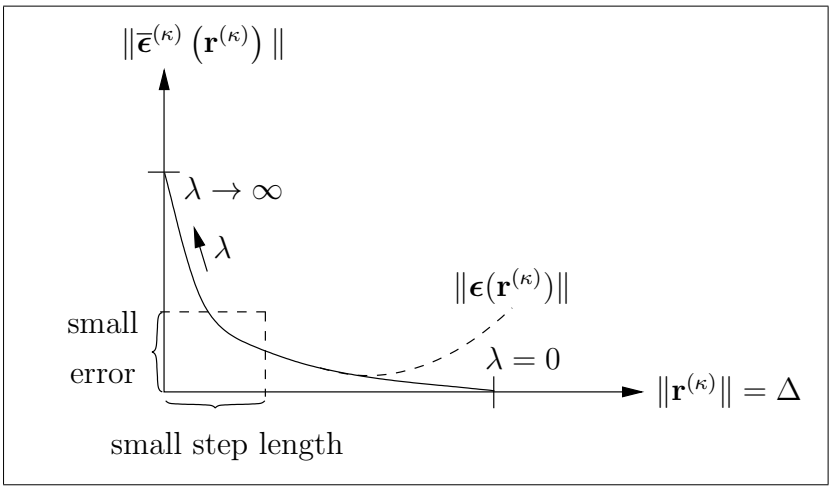


Figure 22. Characteristic boundary curve in a least-squares with trust-region optimization step, typical sharp bend for ill-conditioned problems.

illustrated in Fig. 22 by the dashed line, which shows how the true error possibly deviates from the linearized the more extended the step length becomes.

From the Pareto front, a reasonable step can be derived. Typical technical problems have many degrees of freedom in the solution. This means that different parameter solutions lead to the same objective values. The design degree of freedom translates to mathematical ill-conditioning and leads to a typical bend in the shape of the Parot front. The bend is used to identify a good step with a lot of reduction in the cost function and a still small step length. It has to be taken care that the bend is within the range of validity of the linear objective model.

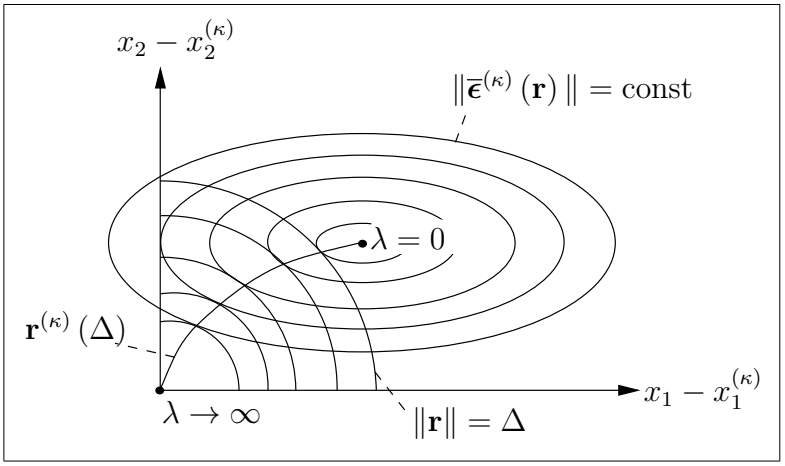


Figure 23. Characteristic boundary curve of a two-dimensional parameter space.

Fig. 23 illustrates the characteristic boundary curve in the space of two parameters. The origin represents the current iteration point. The concentric circles around the origin are the different trust regions of constrained step length. The concentric ellipses are the level contours of the scalarized cost function. The problem condition results in the degree of spread of one of the two axes of the ellipses, which corresponds to the two eigenvalues of the part $\mathbf{S}^T \cdot \mathbf{S}$ of the Hessian of the objective function. For each feasible step length, the touching point of the circle with a cost ellipsis is the corresponding step $\mathbf{r}^{(\kappa)}$. All

possible steps are shown on the curved trajectory from the origin to the minimum of the cost function. One can imagine that the more ill-conditioned the problem is, the stronger the difference in eigenvalues and the more bent the curve will be.

Significant research effort has been spent at the Chair of Electronic Design Automation of the Technical University of Munich to develop methods to efficiently compute the characteristic boundary curve for unconstrained and constrained optimization problems. Several commercial circuit and transistor parameter optimization tools have adopted the characteristic boundary curve.

5.11 Conjugate-gradient (CG) approach for multivariate unconstrained optimization with derivatives

Please recall that the minimization of a quadratic objective function leads to the solution of a linear equation system due to the first-order optimality condition in the Newton approach (138). The conjugate-gradient approach was originally developed to solve a linear equation system. We will use these notations:

$$f_q(\mathbf{x}) = \mathbf{b}^T \cdot \mathbf{x} + \frac{1}{2} \cdot \mathbf{x}^T \cdot \mathbf{H} \cdot \mathbf{x} \overset{!}{\rightarrow} \min, \quad \mathbf{H} \text{ is symmetric, positive definite} \tag{159}$$

$$\nabla f_q(\mathbf{x}) = \mathbf{g}(\mathbf{x}) = \mathbf{0} \rightarrow \mathbf{H} \cdot \mathbf{x} = -\mathbf{b} \rightarrow \mathbf{x}^\star \tag{160}$$

The conjugate-gradient approach is dedicated to a linear equation system, such as (160) that is **large** and has a system matrix $\mathbf{H}$ that is **sparse**. In this case, the usual matrix decomposition or an eigenvalue decomposition are prohibitive, as they destroy the sparsity and lead to unnecessary computational cost and even memory overflow. A more efficient solution approach is needed. The concept of conjugate directions achieves this.

Conjugate directions

Conjugate directions are characterized by the property of **H-orthogonality** of any two directions $\mathbf{r}^{(i)}$ and $\mathbf{r}^{(j)}$:

$$\boxed{\underset{i \neq j}{\forall} \mathbf{r}^{(i)^T} \cdot \mathbf{H} \cdot \mathbf{r}^{(j)} = 0 \quad \text{(H-orthogonal)}} \tag{161}$$

H-orthogonality describes that the orthogonal transformation of two vectors according to an eigenvalue decomposition would make them orthogonal. Take the eigenvalue decomposition of $\mathbf{H}$:

$$\mathbf{H} = \mathbf{U} \cdot \mathbf{D}^{\frac{1}{2}} \cdot \mathbf{D}^{\frac{1}{2}} \cdot \mathbf{U}^T, \;\; \mathbf{U}^{-1} = \mathbf{U}^T, \;\; \mathbf{U} \cdot \mathbf{U}^T = \mathbf{I} \tag{162}$$

where $\mathbf{D}^{\frac{1}{2}}$ is the diagonal matrix of square roots of eigenvalues, and where the columns of $\mathbf{U}$ are the corresponding orthonormal eigenvectors.

Inserting (162) into (161) gives:

$$\begin{aligned} & \underset{i \neq j}{\forall} \mathbf{r}^{(i)^T} \cdot \mathbf{H} \cdot \mathbf{r}^{(j)} = 0 \\ \Longleftrightarrow \quad & \underset{i \neq j}{\forall} \left(\mathbf{D}^{\frac{1}{2}} \cdot \mathbf{U}^T \cdot \mathbf{r}^{(i)} \right)^T \cdot \left(\mathbf{D}^{\frac{1}{2}} \cdot \mathbf{U}^T \cdot \mathbf{r}^{(j)} \right) = 0 \\ \Longleftrightarrow \quad & \underset{i \neq j}{\forall} \mathbf{r}^{(i)'^T} \cdot \mathbf{r}^{(j)'} = 0 \end{aligned} \tag{163}$$

This shows that the transformed conjugate directions

$$\mathbf{r}^{(i)'^T} = \mathbf{D}^{\frac{1}{2}} \cdot \mathbf{U}^T \cdot \mathbf{r}^{(i)} \tag{164}$$

are orthogonal. In the following, a solution to the problem of minimizing a quadratic objective function will be presented.

Search direction

A new search direction is defined to be a **linear combination of the actual steepest descent and the previous search direction**:

$$\boxed{\mathbf{r}^{(\kappa+1)} = -\mathbf{g}^{(\kappa+1)} + \beta^{(\kappa+1)} \cdot \mathbf{r}^{(\kappa)}} \tag{165}$$

Substituting (165) into (161) leads to a formula of the combination factor $\beta^{(\kappa+1)}$:

$$-\mathbf{g}^{(\kappa+1)^T} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)} + \beta^{(\kappa+1)} \cdot \mathbf{r}^{(\kappa)^T} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)} = 0 \tag{166}$$

$$\boxed{\beta^{(\kappa+1)} = \frac{\mathbf{g}^{(\kappa+1)^T} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)}}{\mathbf{r}^{(\kappa)^T} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)}}} \tag{167}$$

Step length

The step length for a given search direction can be formulated analytically for a quadratic objective function. We substitute $\mathbf{x} = \mathbf{x}^{(\kappa)} + \alpha \cdot \mathbf{r}^{(\kappa)}$ in (159):

$$f_q(\alpha) = \mathbf{b}^T \cdot \left(\mathbf{x}^{(\kappa)} + \alpha \cdot \mathbf{r}^{(\kappa)}\right) + \frac{1}{2} \cdot \left(\mathbf{x}^{(\kappa)} + \alpha \cdot \mathbf{r}^{(\kappa)}\right)^T \cdot \mathbf{H} \cdot \left(\mathbf{x}^{(\kappa)} + \alpha \cdot \mathbf{r}^{(\kappa)}\right) \tag{168}$$

$$\begin{aligned} \nabla f_q(\alpha) &= \mathbf{b}^T \cdot \mathbf{r}^{(\kappa)} + \left(\mathbf{x}^{(\kappa)} + \alpha \cdot \mathbf{r}^{(\kappa)}\right)^T \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)} && (169) \\ &\overset{(160)}{=} \underbrace{\nabla f_q\left(\mathbf{x}^{(\kappa)}\right)^T}_{\mathbf{g}^{(\kappa)^T}} \cdot \mathbf{r}^{(\kappa)} + \alpha \cdot \mathbf{r}^{(\kappa)T} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)} && (170) \end{aligned}$$

The first-order optimality condition $\nabla f_q(\alpha) = 0$ then leads to:

$$\boxed{\alpha^{(\kappa)} = \frac{-\mathbf{g}^{(\kappa)T} \cdot \mathbf{r}^{(\kappa)}}{\mathbf{r}^{(\kappa)T} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)}}} \tag{171}$$

At this point, we have all ingredients for an iterative search process. Before formulating it, we will look at some gradients' properties and search directions we have just defined. We will show that the condition (161) holds while formulating these conditions. Afterward, we will derive simplified formulas for the search direction and step length, which are cheaper to evaluate. Then, the conjugate gradient algorithm will be formulated. And it will be extended to nonlinear optimization!

Some properties

We formulate the gradient of the next iteration step from the formula for a new step,

$$\begin{aligned} \mathbf{g}^{(\kappa+1)} &= \mathbf{b} + \mathbf{H} \cdot \mathbf{x}^{(\kappa+1)} = \mathbf{b} + \mathbf{H} \cdot \mathbf{x}^{(\kappa)} + \alpha^{(\kappa)} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)} \\ &= \mathbf{g}^{(\kappa)} + \alpha^{(\kappa)} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)} \end{aligned} \tag{172}$$

and insert this into:

$$\begin{aligned}
\mathbf{g}^{(\kappa+1)^T} \cdot \mathbf{r}^{(\kappa)} &\overset{(172)}{=} \mathbf{g}^{(\kappa)^T} \cdot \mathbf{r}^{(\kappa)} + \alpha^{(\kappa)} \cdot \mathbf{r}^{(\kappa)^T} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)} \\
&\overset{(171)}{=} \mathbf{g}^{(\kappa)^T} \cdot \mathbf{r}^{(\kappa)} - \mathbf{g}^{(\kappa)^T} \cdot \mathbf{r}^{(\kappa)} \cdot \frac{\mathbf{r}^{(\kappa)^T} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)}}{\mathbf{r}^{(\kappa)^T} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)}} \\
&= 0
\end{aligned} \tag{173}$$

This result shows that the actual gradient is orthogonal to the previous search direction.

We establish the following relation between the actual gradient and the actual search direction:

$$\begin{aligned}
-\mathbf{g}^{(\kappa)^T} \cdot \mathbf{r}^{(\kappa)} &\overset{(165)}{=} -\mathbf{g}^{(\kappa)^T} \cdot \left(-\mathbf{g}^{(\kappa)} + \beta^{(\kappa)} \cdot \mathbf{r}^{(\kappa-1)}\right) \\
&= \mathbf{g}^{(\kappa)^T} \cdot \mathbf{g}^{(\kappa)} - \beta^{(\kappa)} \cdot \underbrace{\mathbf{g}^{(\kappa)^T} \cdot \mathbf{r}^{(\kappa-1)}}_{0 \text{ by } (173)} \\
&= \mathbf{g}^{(\kappa)^T} \cdot \mathbf{g}^{(\kappa)}
\end{aligned} \tag{174}$$

This property will be used for a simplified formula for the step length.

Next, we look at the scalar product of two subsequent gradients:

$$\begin{aligned}
\mathbf{g}^{(\kappa+1)^T} \cdot \mathbf{g}^{(\kappa)} &\overset{(172)}{=} \mathbf{g}^{(\kappa)^T} \cdot \mathbf{g}^{(\kappa)} + \alpha^{(\kappa)} \cdot \mathbf{r}^{(\kappa)^T} \cdot \mathbf{H} \cdot \mathbf{g}^{(\kappa)} \\
&\overset{(174),(165)}{=} -\mathbf{g}^{(\kappa)^T} \cdot \mathbf{r}^{(\kappa)} + \alpha^{(\kappa)} \cdot \mathbf{r}^{(\kappa)^T} \cdot \mathbf{H} \cdot \left(-\mathbf{r}^{(\kappa)} + \beta^{(\kappa)} \cdot \mathbf{r}^{(\kappa-1)}\right) \\
&\overset{(161)}{=} -\mathbf{g}^{(\kappa)^T} \cdot \mathbf{r}^{(\kappa)} - \alpha^{(\kappa)} \cdot \mathbf{r}^{(\kappa)^T} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)} \\
&\overset{(171)}{=} -\mathbf{g}^{(\kappa)^T} \cdot \mathbf{r}^{(\kappa)} + \mathbf{g}^{(\kappa)^T} \cdot \mathbf{r}^{(\kappa)} \\
&= 0
\end{aligned} \tag{175}$$

That shows that the actual gradient is orthogonal to the previous gradient.

Substituting (165) into (173)

$$\begin{aligned}
&\mathbf{g}^{(\kappa+1)^T} \cdot \left(-\mathbf{g}^{(\kappa)} + \beta^{(\kappa)} \cdot \mathbf{r}^{(\kappa-1)}\right) = 0 \\
\overset{(175)}{\Longleftrightarrow} \quad &\mathbf{g}^{(\kappa+1)^T} \cdot \mathbf{r}^{(\kappa-1)} = 0
\end{aligned} \tag{176}$$

shows that the new gradient of the next step is orthogonal to the search direction of the previous step. Repeatedly inserting (165) in this equation (176) then leads to the result that the **following gradient is orthogonal to ALL previous search directions**.

Substituting (172) into (175)

$$\begin{aligned}
&\mathbf{g}^{(\kappa+1)^T} \cdot \left(\mathbf{g}^{(\kappa-1)} + \alpha^{(\kappa-1)} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa-1)}\right) = 0 \\
\overset{(165)}{\Longleftrightarrow} \quad &\mathbf{g}^{(\kappa+1)^T} \cdot \mathbf{g}^{(\kappa-1)} + \alpha^{(\kappa-1)} \cdot \left(-\mathbf{r}^{(\kappa+1)} + \beta^{(\kappa+1)} \cdot \mathbf{r}^{(\kappa)}\right) \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa-1)} = 0 \\
\overset{(161)}{\Longleftrightarrow} \quad &\mathbf{g}^{(\kappa+1)^T} \cdot \mathbf{g}^{(\kappa-1)} = 0
\end{aligned} \tag{177}$$

shows that the next gradient is orthogonal to the previous gradient. Repeatedly inserting (172) into (177) then leads to the result that the **next gradient is orthogonal to ALL previous gradients**.

Simplified computation of conjugate gradient step length

Substituting (174) into (171) gives the following formula for the step length:

$$\boxed{\alpha^{(\kappa)} = \frac{\mathbf{g}^{(\kappa)^T} \cdot \mathbf{g}^{(\kappa)}}{\mathbf{r}^{(\kappa)^T} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)}}} \tag{178}$$

Simplified computation of conjugate gradient search direction

Substituting $\mathbf{H} \cdot \mathbf{r}^{(\kappa)}$ from (172) into (167) gives:

$$\beta^{(\kappa+1)} = \frac{\mathbf{g}^{(\kappa+1)^T} \cdot \frac{1}{\alpha^{(\kappa)}} \cdot \left(\mathbf{g}^{(\kappa+1)} - \mathbf{g}^{(\kappa)}\right)}{\mathbf{r}^{(\kappa)^T} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)}} \tag{179}$$

Using (178) and (175) in (179) finally leads to:

$$\boxed{\beta^{(\kappa+1)} = \frac{\mathbf{g}^{(\kappa+1)^T} \cdot \mathbf{g}^{(\kappa+1)}}{\mathbf{g}^{(\kappa)^T} \cdot \mathbf{g}^{(\kappa)}}} \tag{180}$$

Here, no matrix-vector multiplication is needed, which reduces the computational cost.

Conjugate gradient algorithm

The following pseudo-code summarizes the iterative conjugate-gradient approach.

$\mathbf{g}^{(0)} = \mathbf{b} + \mathbf{H} \cdot \mathbf{x}^{(0)}$
$\mathbf{r}^{(0)} = -\mathbf{g}^{(0)}$
$\kappa = 0$
while $\mathbf{r}^{(k)} \neq \mathbf{0}$
  step length: $\alpha^{(\kappa)} = \frac{\mathbf{g}^{(\kappa)^T} \cdot \mathbf{g}^{(\kappa)}}{\mathbf{r}^{(\kappa)^T} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)}}$ $\Rightarrow$ line search in nonlinear programming
  new point: $\mathbf{x}^{(\kappa+1)} = \mathbf{x}^{(\kappa)} + \alpha^{(\kappa)} \cdot \mathbf{r}^{(\kappa)}$
  new gradient: $\mathbf{g}^{(\kappa+1)} = \mathbf{g}^{(\kappa)} + \alpha^{(\kappa)} \cdot \mathbf{H} \cdot \mathbf{r}^{(\kappa)}$ $\Rightarrow$ gradient computation in nonlinear programming
  Fletcher-Reeves: $\beta^{(\kappa+1)} = \frac{\mathbf{g}^{(\kappa+1)^T} \cdot \mathbf{g}^{(\kappa+1)}}{\mathbf{g}^{(\kappa)^T} \cdot \mathbf{g}^{(\kappa)}}$
  new search-direction: $\mathbf{r}^{(\kappa+1)} = -\mathbf{g}^{(\kappa+1)} + \beta^{(\kappa+1)} \cdot \mathbf{r}^{(\kappa)}$
  $\kappa := \kappa + 1$
end while

Please note that the CG algorithm solves optimization problems with a quadratic objective function (159), (160) in at most n_x steps.

Please also note that the formula we have derived for β in the search direction is the **Fletcher-Reeves** formula. Other formulas have been derived, such as, e.g., the **Polak-Ribière** formula:

$$\beta^{(\kappa+1)} = \frac{\mathbf{g}^{(\kappa+1)^T} \cdot \left(\mathbf{g}^{(\kappa+1)} - \mathbf{g}^{(\kappa)}\right)}{\mathbf{g}^{(\kappa)^T} \cdot \mathbf{g}^{(\kappa)}} \tag{181}$$

The structure of the conjugate gradient algorithm can be generalized to a nonlinear optimization problem:

- A general line search replaces the analytical formula for the step length.
- A general gradient computation replaces the analytical formula for a new gradient.

What stands out as an essential feature of the CG approach for nonlinear programming is the orthogonality properties of the search directions, which are H-orthogonal with regard to the Hessian matrix of the Taylor series of the objective function, which is notably never explicitly determined.

The intuitively clear message of the CG approach is to look into a new direction in each iteration step of the optimization process which has not been looked at before. In addition, H orthogonality allows us to adapt these "looks" to the shape of the quadratic part of the objective.

6 Constrained optimization

We have seen how to solve an unconstrained optimization problem with a quadratic objective function. We will extend this to solve a so-called quadratic optimization problem with a quadratic optimization objective and linear constraints. This will be done in two steps. First, linear equality constraints will be treated. A so-called coordinate transformation will transform the problem into a subspace without constraints. There, we have a quadratic objective and can use the solution methods from the previous chapter. Second, linear inequality constraints will be treated. Here, the active-set method will be introduced that maps certain inequality constraints into equality constraints (and back to inequality constraints.) And how to deal with equality constraints has been introduced before. Then, we will formulate the general nonlinear optimization problem. We will see how to transform it into a sequence of quadratic optimization problems (whose solution has been explained just before) using the Newton approach to a Lagrange formulation of the problem.

6.1 Quadratic Programming (QP) – linear equality constraints

The general formulation of Quadratic Programming with linear equality constraints is the following:

$$\boxed{\begin{array}{l} \min f_q(\mathbf{x}) \quad \text{s.t. } \mathbf{A}\cdot\mathbf{x} = \mathbf{c} \\ f_q(\mathbf{x}) = \mathbf{b}^T\cdot\mathbf{x} + \frac{1}{2}\mathbf{x}^T\cdot\mathbf{H}\cdot\mathbf{x} \\ \mathbf{H}\text{: symmetric, positive definite} \\ \mathbf{A}_{<n_c\times n_x>}, n_c \leq n_x, \ \text{rank}(\mathbf{A}) = n_c \end{array}} \tag{182}$$

Here, we assume that the linear equation system of constraints is underdetermined and has full rank (Appendix F). Full rank means that the equations are independent and cannot be further reduced to a system with fewer equations. Underdetermined means that there is a manifold of solutions to the equation system and no unique solution. The degree of freedom in the solution of the equation system leaves space for minimizing the objective function.

Transformation of (182) **into an unconstrained optimization problem by coordinate transformation**

A generic formula to approach coordinate transformation without prior mathematical knowledge is the following linear combination of the right side of the constraint $\mathbf{c}$ and a vector of coordinates $\mathbf{y}$ that are unconstrained:

$$\mathbf{x} = \underbrace{\left[\, \mathbf{Y}_{<n_x \times n_c>} \,|\, \mathbf{Z} \right]_{<n_x \times n_x>}}_{\text{non-singular}} \cdot \begin{bmatrix} \mathbf{c}_{<n_c>} \\ \hline \mathbf{y} \end{bmatrix}_{<n_x>} = \underbrace{\mathbf{Y} \cdot \mathbf{c}}_{\substack{\text{feasible point} \\ \text{of } \mathbf{Ax} = \mathbf{c}}} + \underbrace{\mathbf{Z} \cdot \mathbf{y}}_{\substack{\text{degrees of} \\ \text{freedom in} \\ \mathbf{Ax} = \mathbf{c}}} \tag{183}$$

Please note that a product of a matrix and a vector can be interpreted as a linear combination of the columns of the matrix with the vector elements as coefficients. Hence, in $\mathbf{Y} \cdot \mathbf{c}$ the components of $\mathbf{c}$ are coordinates in a coordinate system where the columns of $\mathbf{Y}$ describe the coordinate axes in the Cartesian coordinate system. Analogously, in $\mathbf{Z} \cdot \mathbf{y}$ the components of $\mathbf{y}$ are coordinates in a coordinate system, where the columns of $\mathbf{Z}$ describe the coordinate axes in the Cartesian coordinate system. We will come back to this soon. If we insert (183) into the constraint equation system, we obtain:

$$\mathbf{A} \cdot \mathbf{x} = \underbrace{\mathbf{A} \cdot \mathbf{Y}}_{(185)} \cdot \mathbf{c} + \underbrace{\mathbf{A} \cdot \mathbf{Z}}_{(186)} \cdot \mathbf{y} = \mathbf{c} \tag{184}$$

The constraint equation system is fulfilled by the following conditions on $\mathbf{Y}$ and $\mathbf{Z}$:

$$\mathbf{A} \cdot \mathbf{Y} = \mathbf{I}_{<n_c \times n_c>} \tag{185}$$

$$\mathbf{A} \cdot \mathbf{Z} = \mathbf{0}_{<n_c \times n_x - n_c>} \tag{186}$$

With these conditions on $\mathbf{Y}$, $\mathbf{Z}$, any vector $\mathbf{y}$ fulfills the constraint equation system. We now insert (183) into the optimization problem (182),

$$\begin{aligned} \phi(\mathbf{y}) &= f_q(\mathbf{x}(\mathbf{y})) && (187) \\ &= \mathbf{b}^T \cdot (\mathbf{Y} \cdot \mathbf{c} + \mathbf{Z} \cdot \mathbf{y}) + \frac{1}{2} (\mathbf{Y} \cdot \mathbf{c} + \mathbf{Z} \cdot \mathbf{y})^T \cdot \mathbf{H} \cdot (\mathbf{Y} \cdot \mathbf{c} + \mathbf{Z} \cdot \mathbf{y}) && (188) \\ &= \left(\mathbf{b} + \frac{1}{2} \mathbf{H} \cdot \mathbf{Y} \cdot \mathbf{c} \right)^T \cdot \mathbf{Y} \cdot \mathbf{c} && (189) \\ &\quad + (\mathbf{b} + \mathbf{H} \cdot \mathbf{Y} \cdot \mathbf{c})^T \cdot \mathbf{Z} \cdot \mathbf{y} \\ &\quad + \frac{1}{2} \mathbf{y}^T \cdot \mathbf{Z}^T \cdot \mathbf{H} \cdot \mathbf{Z} \cdot \mathbf{y} \end{aligned}$$

and obtain an unconstrained optimization problem in $\phi(\mathbf{y})$ that is equivalent to the original problem:

$$\min f_q(\mathbf{x}) \;\; \text{s.t.} \;\; \mathbf{A} \cdot \mathbf{x} = \mathbf{c} \;\; \equiv \;\; \min \phi(\mathbf{y}) \tag{190}$$

The stationary point $\mathbf{y}^\star$ of the unconstrained optimization problem (190) is computed through $\nabla \phi(\mathbf{y}) = \mathbf{0}$, and transformed back to the constrained solution $\mathbf{x}^\star$:

$$\underbrace{\left(\mathbf{Z}^T \cdot \mathbf{H} \cdot \mathbf{Z}\right)}_{\text{reduced Hessian}} \cdot \mathbf{y} = - \underbrace{\mathbf{Z}^T \cdot (\mathbf{b} + \mathbf{H} \cdot \mathbf{Y} \cdot \mathbf{c})}_{\text{reduced gradient}} \;\; \rightarrow \;\; \mathbf{y}^\star \;\; \rightarrow \;\; \mathbf{x}^\star = \mathbf{Y} \cdot \mathbf{c} + \mathbf{Z} \cdot \mathbf{y}^\star \tag{191}$$

Computation of Y, Z by QR-decomposition

An example of how the matrices $\mathbf{Y}$ and $\mathbf{Z}$ can be determined is the QR-decomposition of the matrix $\mathbf{A}^T = \mathbf{Q} \cdot \mathbf{R}$. Please see App. F for more details on QR-decomposition. Inserting the QR-decomposition of $\mathbf{A}$ into the conditions (185) and (186) yields the following formulas for $\mathbf{Y}$ and $\mathbf{Z}$:

$$\text{from (185): } \mathbf{R}^T \cdot \mathbf{Q}^T \cdot \mathbf{Y} = \mathbf{I} \;\rightarrow\; \boxed{\mathbf{Y} = \mathbf{Q} \cdot \mathbf{R}^{-T}} \tag{192}$$

$$\text{from (185): } \mathbf{R}^T \cdot \mathbf{Q}^T \cdot \mathbf{Z} = \mathbf{0} \;\rightarrow\; \boxed{\mathbf{Z} = \mathbf{Q}^{\perp}} \tag{193}$$

Figure 24 illustrates the coordinate transformation obtained for three parameters x_1 to x_3 using a QR-decomposition.

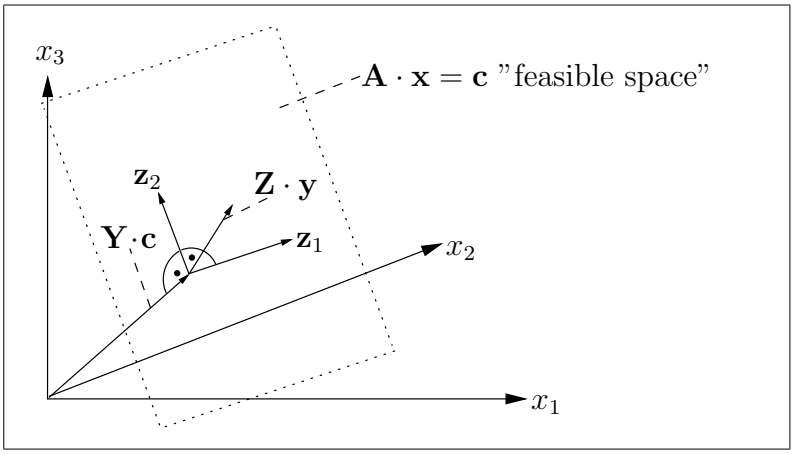


Figure 24. Illustration of coordinate transformation with a QR-decomposition for three parameters.

The plane in the three-dimensional space that is indicated by the dotted rectangle represents all parameter vectors $\mathbf{x}$ that satisfy the constraint equation, i.e., the feasible space of $\mathbf{x}$: any parameter set on this plane is a feasible solution. The optimization objective determines the specific parameter set on this plane that minimizes the quadratic objective. In the case of a QR-decomposition, the vector $\mathbf{Y} \cdot \mathbf{c}$ is the shortest vector that satisfies the constraint equation. It is orthogonal to the plane and represents the minimum-length solution of the underdetermined equation system. $\mathbf{Z} \cdot \mathbf{y}$ lies in the kernel of $\mathbf{A}$. The column vectors $\mathbf{z}_i, i = 1, \cdots, n_x - n_c$ are an orthonormal basis of the kernel (null-space) of $\mathbf{A}$, i.e., it holds $\mathbf{A} \cdot (\mathbf{Z} \cdot \mathbf{y}) = \mathbf{0}$. These column vectors represent an orthogonal coordinate system in the feasible space; the components of $\mathbf{y}$ are the coordinates of points within that feasible space using the coordinate system spanned by the columns of $\mathbf{Z}$.

Computation of Lagrange factors $\boldsymbol{\lambda}$

For the active set method for inequality constraints in the next section, we need to be able to determine the value of the Lagrange factors of the corresponding Lagrange function of optimization problem (182):

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f_q(\mathbf{x}) - \boldsymbol{\lambda}^T \cdot (\mathbf{A} \cdot \mathbf{x} - \mathbf{c}) \tag{194}$$

The stationary point $\nabla \mathcal{L}(\mathbf{x}) = 0$ yields an overdetermined equation system that can be solved using (185)

$$\begin{aligned} \mathbf{A}^T \cdot \boldsymbol{\lambda} &= \nabla f_q(\mathbf{x}) = \mathbf{b} + \mathbf{H} \cdot \mathbf{x} \\ \boldsymbol{\lambda} &\overset{(185)}{=} \mathbf{Y}^T \cdot \nabla f_q(\mathbf{x}^\star) \\ &= \mathbf{Y}^T \cdot (\mathbf{b} + \mathbf{H} \cdot \mathbf{x}^\star) \end{aligned} \tag{195}$$

In case of a QR-decomposition we replace $\mathbf{A}^T$ by $\mathbf{Q} \cdot \mathbf{R}$ in the first part of (195) to obtain:

$$\mathbf{Q} \cdot \mathbf{R} \cdot \boldsymbol{\lambda} = \mathbf{b} + \mathbf{H} \cdot \mathbf{x} \tag{196}$$

$$\boxed{\mathbf{R} \cdot \boldsymbol{\lambda} = \mathbf{Q}^T \cdot (\mathbf{b} + \mathbf{H} \cdot \mathbf{x}) \rightarrow \boldsymbol{\lambda}} \tag{197}$$

Solving (197) is possible by backward substitution, as $\mathbf{R}$ is an upper triangular matrix. As a result, we obtain the values of $\boldsymbol{\lambda}$ at any point of the optimization process, either immediately or at the optimum.

Computation of $Y \cdot \mathbf{c}$ in case of a QR-decomposition

By replacing the QR-decomposition for the system matrix $\mathbf{A}$ in the equality constraint, the product $Y \cdot \mathbf{c}$ can be computed in two steps.

$$\begin{aligned} \mathbf{A} \cdot \mathbf{x} &= \mathbf{c} \\ \overset{(461)}{\Longleftrightarrow} \mathbf{R}^T \cdot \underbrace{\mathbf{Q}^T \cdot \mathbf{x}}_{\mathbf{u}} &= \mathbf{c} \\ & \qquad \overset{\text{forward substitution}}{\longrightarrow} \\ \mathbf{R}^T \cdot \mathbf{u} &= \mathbf{c} \qquad \mathbf{u} \\ \mathbf{Y} \cdot \mathbf{c} &\overset{(192)}{=} \mathbf{Q} \cdot \mathbf{R}^{-T} \cdot \mathbf{c} = \mathbf{Q} \cdot \mathbf{u} \text{ (insert } \mathbf{u}) \end{aligned} \tag{198}$$

First, the intermediate vector $\mathbf{u}$ is computed by forward substitution (please note that $\mathbf{R}^T$ is a lower triangular matrix). Then, $Y \cdot \mathbf{c}$ is obtained as described above.

6.2 Quadratic Programming – inequality constraints

The general formulation of Quadratic Programming with linear inequality constraints and with linear equality constraints is the following:

$$\boxed{\begin{aligned} \min f_q(\mathbf{x}) \quad \text{s.t.} \quad & \mathbf{a}_i^T \cdot \mathbf{x} = c_i, \quad i \in E \\ & \mathbf{a}_j^T \cdot \mathbf{x} \geq c_j, \quad j \in I, \quad E \cap I = \emptyset \\ & f_q(\mathbf{x}) = \mathbf{b}^T \cdot \mathbf{x} + \tfrac{1}{2}\mathbf{x}^T \cdot \mathbf{H} \cdot \mathbf{x} \\ & \mathbf{H}\text{: symmetric, positive definite} \\ & \mathbf{A}_{<n_c \times n_x>}, n_c \leq n_x, \ \operatorname{rank}(\mathbf{A}) = n_c \end{aligned}} \tag{199}$$

$$\mathbf{A} = \begin{bmatrix} \vdots \\ \mathbf{a}_i^T, i \in E \\ \vdots \\ \cdots\cdots\cdots \\ \vdots \\ \mathbf{a}_j^T, j \in I \\ \vdots \end{bmatrix}$$

Here, we have written the equations and inequations individually. The index i denotes equations out of the set E of equality constraints. The index j denotes inequations out of the set I of inequality constraints. Combining all constraint equations and inequations in a matrix $\mathbf{A}$, we assume that this matrix has full rank and that the feasible region is not empty.

An approach to handle inequalities, this holds not only for Quadratic Programming but for nonlinear programming in general, is to iteratively make them to equality constraints when appropriate and address a resulting problem with only equality constraints by coordinate transformation, as described in the previous section. Equality constraints (if they are initially inequality constraints) are turned back to inequality constraints whenever appropriate.

When are inequality constraints made to equality constraints? The moment they are threatened to be violated, i.e., when they reach the bound 0. An iteration step calculates a search direction for the respective set of equality constraints. In this computation, some inequality constraints are ignored. Therefore, the resulting search direction may violate one or more inequality constraints. This is checked for; if it happens, the search direction is limited so that the constraints are not violated but only reach their bounds. They are said to become "active", hence the name of the approach: **active set method**. As an inequality constraint becomes active, it is made an equality constraint, and the computation restarts with a new set of active constraints.

In the further course of optimization, it may turn out that the optimization wants to move away from the bound of an active inequality constraint back into the feasible region. How do we know if an active inequality constraint should become inactive? We compute the

Lagrange multiplier: if the Lagrange multiplier is negative, this means that this condition is not necessary for an optimum, and we can set it inactive again. Then, the active set of constraints changes, and we can search for a new optimization direction.

This process is repeated until all active constraints have positive Lagrange multipliers, i.e., satisfy the first-order optimality conditions, and a stationary solution of the Quadratic Programming problem (199) has been reached.

The active set method is sketched in the following pseudo-code and illustrated with an example afterward. Before going through the example, we will go through the algorithmic principle.

We start with the first line of the pseudo-code. The starting point has to be a point that is inside the feasible region. Usually, it is a corner point of the feasible region, which has, for instance, been computed with linear programming. As a result of the computation of the starting point, a particular set of active constraints is also given.

Then, a repeat loop is started. At the beginning of this loop, Quadratic Programming with equality constraint with the current set of active constraints is performed. The solution approach using coordinate transformation has been described in the previous section. As a result, the computed search direction $\delta^{(\kappa)}$ may be zero (**case I**) or non-zero (**case II**):

In **case I**, a zero search direction means no further step is possible with the current set of active constraints. There are now two possibilities:

In **case Ia**, all Lagrange multipliers of active inequality constraints are non-negative. That means the first-order optimality conditions are fulfilled, and the optimum solution has been reached. We can stop.

In **case Ib**, at least one negative Lagrange multiplier of active inequality constraints exists. This means that the first-order optimality conditions are not fulfilled, and the optimum solution has not been reached. An inequality constraint with a negative Lagrange multiplier should be inactive, and the repeat loop should continue. Which constraint should be set inactive if there are several with negative Lagrange multipliers? To answer this question, please recall what has been revealed when interpreting the Lagrange multiplier: the value of the Lagrange multiplier is the sensitivity of the solution concerning a change in the constraint. Therefore, the inequality constraint with the most negative value should be inactive. This will enable the largest possible progress towards the minimum among all active inequality constraints with negative Lagrange multipliers.

Now to **case II**. We can do a step, but as we have ignored all inactive inequality constraints in the computation of this step, we have to check if we surpass one of the ignored constraints. This can be done by computing the safety margin of an inequality constraint to its bound and dividing it by how much we would consume of it with a complete step along the search direction. This gives the step length to reach the border defined by the considered constraint. This is done for all inactive constraints for which the search direction moves towards the bound. If the computed step length is greater than one, no new constraint becomes active, and the entire search direction length can be taken (**case IIa**). If the computed step length is less or equal to one, one of the inequality constraints becomes active (**case IIb**). The calculated step length is taken, and the corresponding constraint is added to the set of active constraints before continuing the repeat loop.

compute initial solution $\mathbf{x}^{(0)}$ that satisfies all constraints, e.g., by linear programming

$\kappa := 0$

repeat

/* problem formulation at current iteration point $\mathbf{x}^{(\kappa)} + \boldsymbol{\delta}$
for active constraints $\mathcal{A}^{(\kappa)}$ (active set method) */

$\min f_q(\boldsymbol{\delta})$ s.t. $\mathbf{a}_i^T \cdot \boldsymbol{\delta} = \mathbf{0}, i \in \mathcal{A}^{(\kappa)} \;\rightarrow\; \boldsymbol{\delta}^{(\kappa)}$/* (search direction) */

case I: $\boldsymbol{\delta}^{(\kappa)} = \mathbf{0}$ /* i.e., $\mathbf{x}^{(\kappa)}$ optimal for current $\mathcal{A}^{(\kappa)}$ */

compute $\boldsymbol{\lambda}^{(\kappa)}$ e.g., by (195) or (197)

case Ia: $\underset{i \in \mathcal{A}^{(\kappa)} \cap I}{\forall} \lambda_i^{(\kappa)} \geq 0$, /* i.e., first-order optimality conditions satisfied */

$\mathbf{x}^\star = \mathbf{x}^{(\kappa)}$; stop

case Ib: $\underset{i \in \mathcal{A}^{(\kappa)} \cap I}{\exists} \lambda_i^{(\kappa)} < 0$,

/* i.e., constraint becomes inactive, further improvement of f possible */

$q = \underset{i \in \mathcal{A}^{(\kappa)} \cap I}{\arg\min} \lambda_i^{(\kappa)}$,

/* i.e., q is most sensitive constraint to become inactive */

$\mathcal{A}^{(\kappa+1)} = \mathcal{A}^{(\kappa)} \setminus \{q\}$

$\mathbf{x}^{(\kappa+1)} = \mathbf{x}^{(\kappa)}$

case II: $\boldsymbol{\delta}^{(\kappa)} \neq \mathbf{0}$, /* i.e., f can be reduced for current $\mathcal{A}^{(\kappa)}$; step-length computation */

$$\alpha^{(\kappa)} = \underbrace{\min_{i \notin \mathcal{A}^{(\kappa)}, \mathbf{a}_i^T \cdot \boldsymbol{\delta}^{(\kappa)} < 0}}_{\substack{\text{inactive constraints that} \\ \text{approach their lower bound}}} \frac{\overbrace{c_i - \mathbf{a}_i^T \cdot \mathbf{x}^{(\kappa)}}^{\text{safety margin bound}}}{\underbrace{\mathbf{a}_i^T \cdot \boldsymbol{\delta}^{(\kappa)}}_{\substack{\text{consumed safety margin} \\ \text{at } \alpha^{(\kappa)} = 1}}}$$

$q' = \text{argmin}$ (–"–)

case IIa: $\alpha^{(\kappa)} > 1$ /* no constraint is violated, go full length of $\delta^{(\kappa)}$ */

$\mathcal{A}^{(\kappa+1)} = \mathcal{A}^{(\kappa)}$

$\alpha^{(\kappa)} = 1$

case IIb: $\alpha^{(\kappa)} \leq 1$ /* closest constraint is violated or hit, set it active, take this $\alpha^{(\kappa)}$ */

$\mathcal{A}^{(\kappa+1)} = \mathcal{A}^{(\kappa)} \cup (q')$

$\mathbf{x}^{(\kappa+1)} = \mathbf{x}^{(\kappa)} + \alpha^{(\kappa)} \cdot \boldsymbol{\delta}^{(\kappa)}$

$\kappa := \kappa + 1$

Example

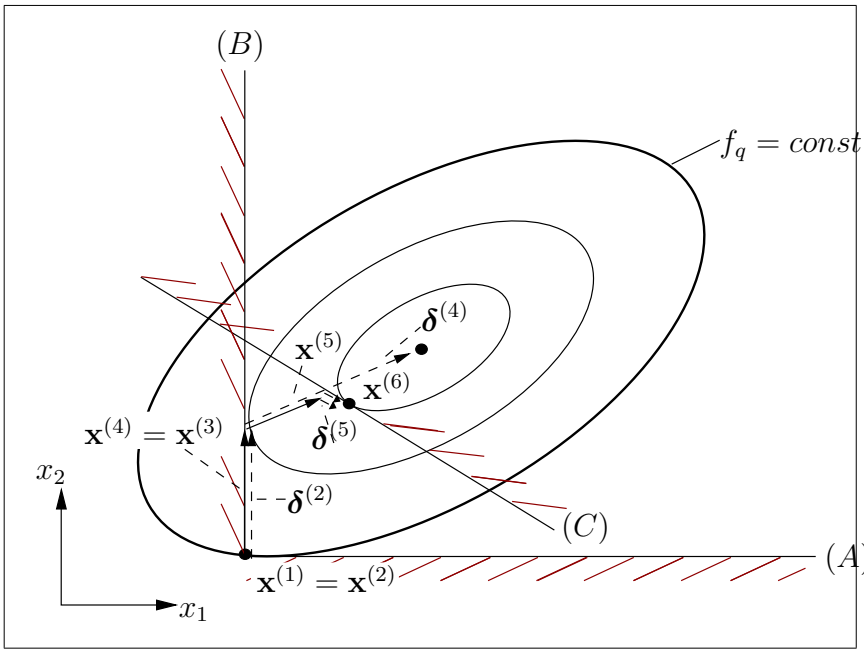


Figure 25. Example of active set process of Quadratic Programming

(κ)	$A^{(\kappa)}$	case
(1)	$\{A, B\}$	Ib
(2)	$\{B\}$	IIa
(3)	$\{B\}$	Ib
(4)	$\{\,\}$	IIb
(5)	$\{C\}$	IIa
(6)	$\{C\}$	Ia

Fig. 25 illustrates the active set process with an example. We have two parameters x_1, x_2. The level contours of the quadratic objective function are given with three ellipses. There are three inequality constraints indicated by three lines and indexes (A), (B), (C). No equality constraints are given in this example. The hatched sides of the lines are the "forbidden" half-spaces. (A) represents the constraint $x_2 \geq 0$, (B) represents the constraint $x_1 \geq 0$, (C) represents that a linear combination of x_1 and x_2 should be less than or equal to some positive value. We can see that the minimum of the objective function is outside the feasible region. We can also immediately identify that the minimum is where constraint boundary (C) touches the most inner ellipsis. But that is not what the algorithm sees.

The algorithm starts at $\mathbf{x}^{(1)}$, i.e., the coordinate system's origin, with an active set of $\{A, B\}$. For instance, this point and active set results if the objective $x_1 + x_2$ has been minimized over this set of inequalities with linear programming.

For this set of active constraints, which says to stay both on the boundary lines of (A) and (B), we cannot move whatever an objective is; the search direction is zero, and we have case I. The question now is if we can set a constraint inactive. Please note that the

algorithm only sets one constraint, active or inactive, at a time. This is for control of convergence. Either we compute the Lagrange multiplier, or, in this case, we can conduct a visual inspection. We draw the gradients of active constraints and objective in the current point $\mathbf{x}_1$ for that purpose. This results in Fig. 26.

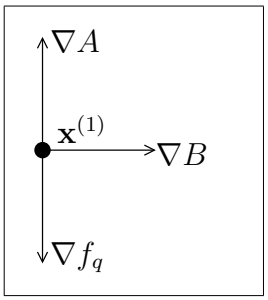


Figure 26. Check first-order optimality condition in iteration step (1).

The gradients point in the direction of the largest function increase. That means upwards for constraint (A), to the right for constraint (B). As constraints are linear, this holds constantly. The gradient of the objective is perpendicular to the respective ellipsis and points away from the objective minimum. From Fig. 26, we can derive for the first-order optimality equation, which is in the first step (1)

$$\nabla f_q^{(1)} = \lambda_A^{(1)} \cdot \nabla A^{(1)} + \lambda_B^{(1)} \cdot \nabla B^{(1)} \ ,$$

the following about the signs of the Lagrange multipliers:

$$\lambda_A^{(1)} < 0, \lambda_B^{(1)} = 0$$

That means the Lagrange multiplier for constraint (A) is negative. We have case Ib, set constraint (A) inactive, set $\mathbf{x}^{(2)} = \mathbf{x}^{(1)}$, and continue with the repeat-loop.

Please note that the lengths of gradients should be normalized to one for this consideration (see tutorial).

In step (2), the search direction from QP with active constraint (B) yields the search direction $\boldsymbol{\delta}^{(2)}$, which is along the boundary line of (B) till it touches the drawn middle contour ellipsis of the objective where the minimum objective value along this line is reached. So we have case II. As no constraint is along the way of $\boldsymbol{\delta}^{(2)}$, we have case IIa and can take the full step to point $\mathbf{x}^{(3)}$.

In step (3), there is no new direction for the same set of active constraints; this is again case I. Can we set the constraint (B) active? Do not look at the full picture; imitate the algorithm and look at the gradient of the constraint and the objective function in $\mathbf{x}^{(3)}$. The gradient of constraint (B) points to the right, and the gradient of the objective in this point goes to the left. Hence objective gradient is obtained by multiplying the constraint with a negative factor, i.e., a negative Lagrange multiplier. Hence, we have case Ib, set $\mathbf{x}^{(4)} = \mathbf{x}^{(3)}$ and set constraint (B) inactive.

Now the set of active constraints is empty. That means the next search direction of step 4 does not see any constraint and yields the complete Newton direction $\boldsymbol{\delta}^{(4)}$. We have case II here. From visual inspection, we see that constraint (C) lies in the way of $\boldsymbol{\delta}^{(4)}$.

That means we have case IIb and can only go till the boundary of (C), stop at $\mathbf{x}^{(5)}$, and set constraint (C) active.

In step (5), with active constraint (C), the Quadratic Programming yields the small step $\boldsymbol{\delta}^{(5)}$ along the boundary of constraint (C) to the next touching ellipsis. We reach point $\mathbf{x}^{(6)}$, as no constraint is on the way; it is the complete step length, hence case IIa.

In the next step (6), a non-zero step with this active constraint set is impossible. Looking at the gradients of the constraint (C) and the objective, we recognize that both point perpendicular to the constraint boundary to the lower left. Hence the Lagrange multiplier is positive; the first-order optimality condition is fulfilled, we have case Ia and reached the optimum.

6.3 Sequential Quadratic Programming (SQP), Lagrange–Newton

Sequential Quadratic Programming – equality constraints

A general nonlinear optimization problem with equality constraints is formulated as follows.

$$\boxed{\begin{array}{lll} \min f(\mathbf{x}) & \text{s.t.} & \mathbf{c}(\mathbf{x}) = \mathbf{0} \\ & & \mathbf{c}_{<n_c>}, n_c \leq n_x \end{array}} \tag{200}$$

Both objective and constraints are nonlinear. The number of constraints is smaller than the number of parameters, and the feasible region is not empty. In the first step, we write the Lagrange formulation of this optimization problem:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\lambda}^T \cdot \mathbf{c}(\mathbf{x}) = f(\mathbf{x}) - \sum_i \lambda_i \cdot c_i(\mathbf{x}) \tag{201}$$

Then, we write the first-order derivatives of the Lagrange function, both with regard to the parameters $\mathbf{x}$ and to the Lagrange multipliers $\boldsymbol{\lambda}$:

$$\begin{aligned} \nabla\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla\mathcal{L}(\mathbf{x}) \\ \\ \nabla\mathcal{L}(\boldsymbol{\lambda}) \end{bmatrix} &= \begin{bmatrix} \nabla f(\mathbf{x}) - \sum_i \lambda_i \cdot \nabla c_i(\mathbf{x}) \\ -\mathbf{c}(\mathbf{x}) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{g}(\mathbf{x}) - \overbrace{\left[\cdots \nabla c_i(\mathbf{x}) \cdots\right]}^{\mathbf{A}^T(\mathbf{x})} \cdot \boldsymbol{\lambda} \\ -\mathbf{c}(\mathbf{x}) \end{bmatrix} \end{aligned} \tag{202}$$

Please note that we have used the previously mentioned property that a matrix-vector product is the linear combination of the matrix columns with vector elements as combination factors and have combined the gradients of constraints as column vectors of a matrix $\mathbf{A}^T(\mathbf{x})$.

Then, we write the second-order derivative of the Lagrange function, again both with regard to the parameters $\mathbf{x}$ and to the Lagrange multipliers $\boldsymbol{\lambda}$:

$$\begin{aligned} \nabla^2\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) &= \begin{bmatrix} \nabla^2\mathcal{L}(\mathbf{x}) & \frac{\partial^2\mathcal{L}}{\partial\mathbf{x}\cdot\partial\boldsymbol{\lambda}^T} \\ \frac{\partial^2\mathcal{L}}{\partial\boldsymbol{\lambda}\cdot\partial\mathbf{x}^T} & \nabla^2\mathcal{L}(\boldsymbol{\lambda}) \end{bmatrix} \\ &= \begin{bmatrix} \nabla^2 f(\mathbf{x}) - \sum_i \lambda_i \cdot \nabla^2 c_i(\mathbf{x}) & -\mathbf{A}^T(\mathbf{x}) \\ \underbrace{\begin{bmatrix} \vdots \\ -\nabla c_i(\mathbf{x})^T \\ \vdots \end{bmatrix}}_{-\mathbf{A}(\mathbf{x})} & \mathbf{0} \end{bmatrix} \end{aligned} \tag{203}$$

With the first-order derivative and the second-order derivative of the Lagrange function, we can formulate a Newton approach by setting the model of the gradient of the Lagrange

function, using the Hessian of the Lagrange function, at a new value of parameter and Lagrange multiplier, i.e., $\mathbf{x}^{(\kappa+1)}$, $\boldsymbol{\lambda}^{(\kappa+1)}$, equal to zero:

$$\nabla\mathcal{L}(\underbrace{\mathbf{x}^{(\kappa)}+\boldsymbol{\delta}^{(\kappa)}}_{\mathbf{x}^{(\kappa+1)}},\underbrace{\boldsymbol{\lambda}^{(\kappa)}+\Delta\boldsymbol{\lambda}^{(\kappa)}}_{\boldsymbol{\lambda}^{(\kappa+1)}}) \approx \nabla\underbrace{\mathcal{L}\left(\mathbf{x}^{(\kappa)},\boldsymbol{\lambda}^{(\kappa)}\right)}_{\nabla\mathcal{L}^{(\kappa)}} + \underbrace{\nabla^2\mathcal{L}\left(\mathbf{x}^{(\kappa)},\boldsymbol{\lambda}^{(\kappa)}\right)}_{\nabla^2\mathcal{L}^{(\kappa)}}\cdot\begin{bmatrix}\boldsymbol{\delta}^{(\kappa)}\\ \Delta\boldsymbol{\lambda}^{(\kappa)}\end{bmatrix} \stackrel{!}{=} \mathbf{0} \tag{204}$$

Here, we have defined the change from $\mathbf{x}^{(\kappa)}$ to $\mathbf{x}^{(\kappa+1)}$ as $\boldsymbol{\delta}^{(\kappa)}$, and the change from $\boldsymbol{\lambda}^{(\kappa)}$ to $\boldsymbol{\lambda}^{(\kappa+1)}$ as $\Delta\boldsymbol{\lambda}^{(\kappa)}$.

Please note that this is a Newton approach to the Lagrange function (including the gradient with regard to the Lagrange multipliers). Hence the overall approach is also called a **Lagrange-Newton approach**.

Inserting (202) and (203) in (204) results in:

$$\boxed{\begin{bmatrix}\overbrace{\nabla^2 f(\mathbf{x}^{(\kappa)})-\sum_i \lambda_i^{(\kappa)}\cdot\nabla^2 c_i\left(\mathbf{x}^{(\kappa)}\right)}^{\mathbf{W}^{(\kappa)}} & -\mathbf{A}^{(\kappa)^T}\\ -\mathbf{A}^{(\kappa)} & \mathbf{0}\end{bmatrix}\cdot\begin{bmatrix}\boldsymbol{\delta}^{(\kappa)}\\ \Delta\boldsymbol{\lambda}^{(\kappa)}\end{bmatrix} = \begin{bmatrix}-\mathbf{g}^{(\kappa)}+\mathbf{A}^{(\kappa)^T}\cdot\boldsymbol{\lambda}^{(\kappa)}\\ \mathbf{c}^{(\kappa)}\end{bmatrix}} \tag{205}$$

With the abbreviation $\mathbf{W}^{(\kappa)}$ and using that $\Delta\boldsymbol{\lambda}^{(\kappa)}$ has a zero coefficient in the second row but appears on the right side of the first row, we obtain the following formulation to iteratively compute new values for $\mathbf{x}^{(\kappa+1)}$ and $\boldsymbol{\lambda}^{(\kappa+1)}$ from the first- and second-order derivatives of objective and constraints in step (κ):

$$\boxed{\begin{bmatrix}\mathbf{W}^{(\kappa)} & -\mathbf{A}^{(\kappa)^T}\\ -\mathbf{A}^{(\kappa)} & \mathbf{0}\end{bmatrix}\cdot\begin{bmatrix}\boldsymbol{\delta}^{(\kappa)}\\ \boldsymbol{\lambda}^{(\kappa+1)}\end{bmatrix} = \begin{bmatrix}-\mathbf{g}^{(\kappa)}\\ \mathbf{c}^{(\kappa)}\end{bmatrix}} \tag{206}$$

$$\Uparrow$$

$$\mathcal{L}(\boldsymbol{\delta},\boldsymbol{\lambda}) = \mathbf{g}^{(\kappa)^T}\cdot\boldsymbol{\delta}+\frac{1}{2}\boldsymbol{\delta}^T\cdot\mathbf{W}^{(\kappa)}\cdot\boldsymbol{\delta}-\boldsymbol{\lambda}^T\cdot\left(\mathbf{A}^{(\kappa)}\cdot\boldsymbol{\delta}+\mathbf{c}^{(\kappa)}\right) \tag{207}$$

$$\Uparrow$$

$$\boxed{\min \mathbf{g}^{(\kappa)^T}\cdot\boldsymbol{\delta}+\frac{1}{2}\boldsymbol{\delta}^T\cdot\mathbf{W}^{(\kappa)}\cdot\boldsymbol{\delta} \text{ s.t. } \mathbf{A}^{(\kappa)}\cdot\boldsymbol{\delta}=-\mathbf{c}^{(\kappa)}} \tag{208}$$

As can be seen, the result of the Lagrange-Newton approach (206) can also be derived over the Lagrange function (207) of a Quadratic Programming problem (208). As this is iteratively repeated, we obtain a sequence of Quadratic Programming problems, hence the name **Sequential Quadratic Programming (SQP)**.

A fascinating feature, which enables the whole process, is that the quadratic information of objective and constraints, i.e., the Hessian matrix of both, all move to the SQP objective function in (208). In contrast, the SQP constraints only contain the linear parts of the original constraints.

Please note that if the second-order derivatives of objective and constraints are unavailable, the quasi-Newton approach is applied to make SQP work.

Please also note that inequality constraints, i.e., $\mathbf{A}^{(\kappa)} \cdot \boldsymbol{\delta} \geq -\mathbf{c}_I^{(\kappa)}$, are treated with an active set method as described for Quadratic Programming. The nonlinearity of objectives and constraints requires methods to keep the constraints satisfied at any time of the optimization process (Feasible SQP, FSQP).

Penalty function

As a side note, instead of an active set method, the constraints can be combined with the objective function into a scalar objective function (penalty function) of an unconstrained optimization problem. The constraints are given a penalty parameter $\mu > 0$ There are different types of penalty functions, e.g.,

quadratic:

$$P_{\text{quad}}(\mathbf{x}, \mu) = f(\mathbf{x}) + \frac{1}{2\mu} \mathbf{c}^T(\mathbf{x}) \cdot \mathbf{c}(\mathbf{x}) \tag{209}$$

logarithmic:

$$P_{\log}(\mathbf{x}, \mu) = f(\mathbf{x}) - \mu \cdot \sum_i \log c_i(\mathbf{x}) \tag{210}$$

l_1-exact:

$$P_{l_1}(\mathbf{x}, \mu) = \mu \cdot f(\mathbf{x}) - \sum_{i \in E} |c_i(\mathbf{x})| + \sum_{i \in I} \max(-c_i(\mathbf{x}), 0) \tag{211}$$

Concluding remark

The basics of nonlinear optimization have been presented in this and the preceding chapter. This was suggested as we discovered that the circuit design task of general worst-case analysis leads to the mathematical formulation of a nonlinear optimization problem. Other circuit/system design tasks also lead to mathematical tasks of nonlinear optimization problems.

Parameter tolerances are an essential design aspect. Before formulating the corresponding design tasks mathematically, the following two chapters will introduce the basics of statistics. Then, we will formulate the tolerance design objectives and their computation in Chap. 9. The general circuit optimization tasks based on nominal and tolerance design objectives will be given in Chap. 10.

7 Statistical parameter tolerances

Up to now, we have learned about basic notations of mathematical optimization, what optimality conditions for unconstrained and constrained optimization are, and how to apply them to formulate the classical, realistic, and general worst-case analysis. For a general worst-case analysis, being a nonlinear optimization problem, we made a tour d'horizon through iterative solution approaches to unconstrained and constrained optimization problems. Now, we will switch to take a look at the tolerance design of circuits/systems. In the following, the basics of statistical parameter tolerances will be sketched. The main idea is to show notations of multivariate statistics with statistical parameter vectors and how to transform one statistical distribution into another.

Let us first recall some terms using a simple picture. For more background on probability space, please look at App. G. Fig. 27 illustrates frequency in statistics. Throwing the

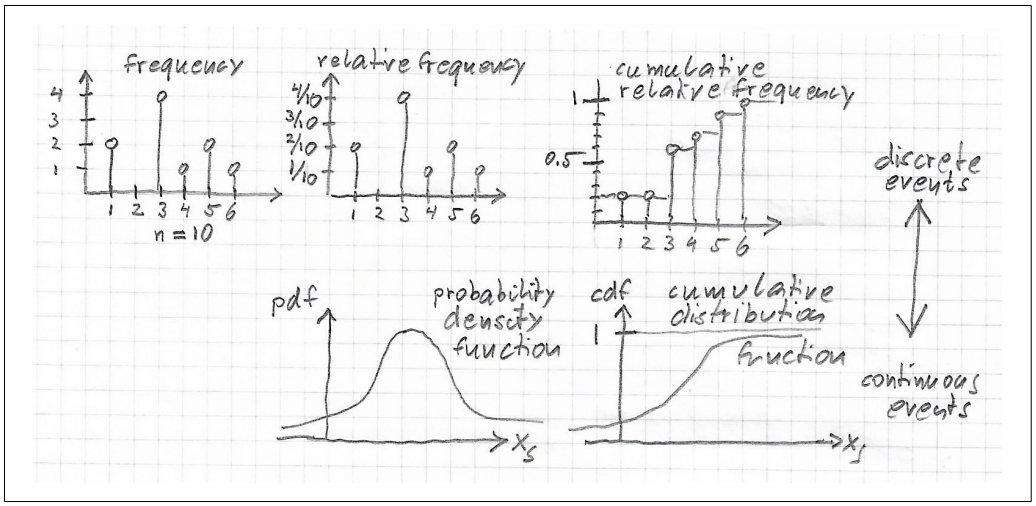


Figure 27. Top: throwing the dice ten times and showing the result. Bottom: function counterparts in case of continuous events

dice is a discrete statistical event space where one of the numbers one to six can happen. The frequency function counts how often each number occurred in a sample of, here, 10 sample elements. If we want to eliminate the absolute number, we divide by the sample size, here, 10, to obtain the relative frequency. We can also denote how often we received a number up to one, up to two, and so on, till up to 6. "Up to zero" represents the impossible event, and "up to six" represents the certain event. The resulting function is called cumulative relative frequency, increasing monotonously from zero to one. On the lower part, the counterparts in the case of continuous events are illustrated. In the continuous case, the relative frequency becomes the probability density function (*pdf*), and the cumulative relative frequency becomes the cumulative distribution function (*cdf*).

We will denote specifically the statistical parameters in the following, as the statistical distributions specifically refer to them. The manufacturing variations are modeled through a multivariate continuous distribution function of these statistical parameters $\mathbf{x}_s$. The

multivariate cumulative distribution function (*cdf*) is a multiple integral over the probability density function of the statistical parameters:

$$cdf\left(\mathbf{x}_s\right) = \int_{-\infty}^{x_{s,1}} \cdots \int_{-\infty}^{x_{s,n_{xs}}} pdf\left(\mathbf{t}\right) \cdot d\mathbf{t} \tag{212}$$

with the abbreviation:

$$d\mathbf{t} = dt_1 \cdot dt_2 \cdot \ldots \cdot dt_{n_{xs}}$$

The **multivariate probability density function** (*pdf*) is obtained by partially deriving the cumulative distribution function with regard to all statistical parameters:

$$pdf\left(\mathbf{x}_s\right) = \frac{\partial^{n_{xs}}}{\partial_{x_{s,1}} \cdots \partial_{x_{s,n_{xs}}}} cdf\left(\mathbf{x}_s\right) \tag{213}$$

In the following, we will first recall the univariate normal distribution and then describe the multivariate normal distribution.

7.1 Univariate Gaussian distribution (univariate normal distribution)

A single random variable x_s that is normally distributed with mean value $x_{s,0}$ and variance σ^2 is denoted as:

$$x_s \sim N\left(x_{s,0}, \sigma^2\right) \tag{214}$$

with

$$\begin{aligned} x_{s,0}: &\quad \text{mean value} \\ \sigma^2: &\quad \text{variance} \\ \sigma: &\quad \text{standard deviation} \end{aligned}$$

The probability density function of the univariate normal distribution is as follows:

$$pdf_N\left(x_s, x_{s,0}, \sigma^2\right) = \frac{1}{\sqrt{2\pi}\cdot\sigma} \cdot e^{-\frac{1}{2}\left(\frac{x_s - x_{s,0}}{\sigma}\right)^2} \tag{215}$$

Fig. 28 shows the probability density function and the cumulative distribution function of a univariate normal distribution.

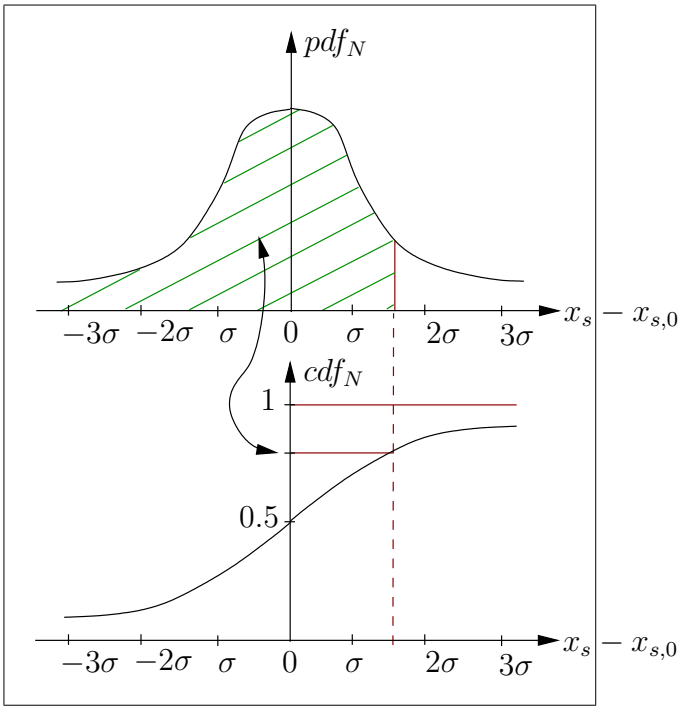


Figure 28. Probability density function, pdf, and corresponding cumulative distribution function, cdf, of a univariate Gaussian distribution. The area of the shaded region under the pdf is the value of the cdf, as shown.

The probability density function has a bell shape around the mean value and is symmetrical. Due to the symmetry, the mean value is also the median, and the cumulative distribution function is 0.5 at the mean value: half of the pdf is on each side.

Typical values of the cumulative distribution function are given in Table 2.

$x_s - x_{s,0}$	-3σ	-2σ	$-\sigma$	0	σ	2σ	3σ	4σ
$cdf\,(x_s - x_{s,0})$	0.1%	2.2%	15.8%	50%	84.2%	97.8%	99.9%	99.99%

Table 2: Typical values of the cumulative distribution function of the univariate normal distribution.

Whenever people talk about "x-sigma" they think of a univariate normal distribution and the percentage within $\pm x\sigma$ or from $-\infty$ to $x\sigma$. 3σ for instance refers to 99.9% for all random variable values below 3σ.

7.2 Multivariate Gaussian distribution (multivariate normal distribution)

A random variable vector $\mathbf{x}_s$ that is normally distributed with mean value vector $\mathbf{x}_{s,0}$ and covariance matrix $\mathbf{C}$ is denoted as

$$\mathbf{x}_s \sim N\left(\mathbf{x}_{s,0}, \mathbf{C}\right) \tag{216}$$

with the abbreviations of $\mathbf{x}_{s,0}$ as vector of mean values of statistical parameters $\mathbf{x}_s$ and $\mathbf{C}$ as symmetric, positive definite covariance matrix of statistical parameters $\mathbf{x}_s$.

The probability density function of the multivariate normal distribution is given as

$$pdf_N\left(\mathbf{x}_s, \mathbf{x}_{s,0}, \mathbf{C}\right) = \frac{1}{\sqrt{2\pi}^{n_{xs}} \cdot \sqrt{\det\left(\mathbf{C}\right)}} \cdot e^{-\frac{1}{2}\beta^2\left(\mathbf{x}_s, \mathbf{x}_{s,0}, \mathbf{C}\right)} \tag{217}$$

$$\beta^2\left(\mathbf{x}_s, \mathbf{x}_{s,0}, \mathbf{C}\right) = \left(\mathbf{x}_s - \mathbf{x}_{s,0}\right)^T \cdot \mathbf{C}^{-1} \cdot \left(\mathbf{x}_s - \mathbf{x}_{s,0}\right) \tag{218}$$

$$\mathbf{C} = \mathbf{\Sigma} \cdot \mathbf{R} \cdot \mathbf{\Sigma} \tag{219}$$

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \sigma_{n_{xs}} \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 1 & \rho_{1,2} & \cdots & \rho_{1,n_{xs}} \\ \rho_{1,2} & 1 & \cdots & \rho_{2,n_{xs}} \\ \vdots & & \ddots & \vdots \\ \rho_{1,n_{xs}} & \rho_{2,n_{xs}} & \cdots & 1 \end{bmatrix} \tag{220}$$

$$\mathbf{C} = \begin{bmatrix} \sigma_1^2 & \sigma_1\rho_{1,2}\sigma_2 & \cdots & \sigma_1\rho_{1,n_{xs}}\sigma_{n_{xs}} \\ \sigma_1\rho_{1,2}\sigma_2 & \sigma_2^2 & \cdots & \sigma_2\rho_{2,n_{xs}}\sigma_{n_{xs}} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_1\rho_{1,n_{xs}}\sigma_{n_{xs}} & \cdots & \cdots & \sigma_{n_{xs}}^2 \end{bmatrix} \tag{221}$$

with the following abbreviations:

$\mathbf{R}$:	correlation matrix of statistical parameters
$\mathbf{C}$:	matrix of variances and covariances
σ_k:	standard deviation of component $x_{s,k}$, $\sigma_k > 0$
σ_k^2:	variance of component $x_{s,k}$
$\sigma_k\rho_{k,l}\sigma_l$:	covariance of components $x_{s,k}$, $x_{s,l}$
$\rho_{k,l}$:	correlation coefficient of components $x_{s,k}$ and $x_{s,l}$, $-1 < \rho_{k,l} < 1$
$\rho_{k,l} = 0$:	uncorrelated and also independent if jointly normal
$\lvert\rho_{k,l}\rvert = 1$:	strongly correlated components

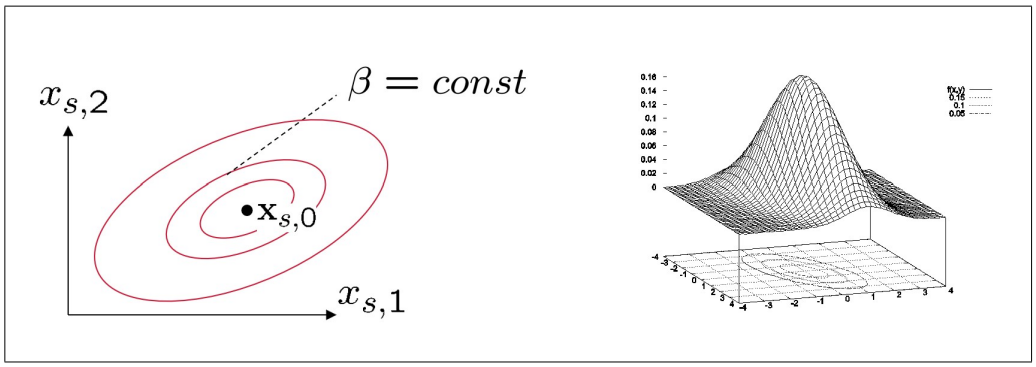


Figure 29. Probability density function of a normal distribution of two statistical parameters (right) and corresponding level contours (left).

Fig. 29 illustrates the probability density function of a normal distribution of two statistical parameters.

You can see the typical Gaussian bell curve of the probability density function (217). The level contours are hyperellipsoids in general, which are described with the quadratic form (218), as we had already anticipated in the realistic and general worst-case analysis. The covariance matrix $\mathbf{C}$ gathers the individual standard deviations σ_k of the statistical parameters and the mutual correlation factors $\rho_{k,l}$ to the variances σ_k^2 in the diagonal and the covariances $\sigma_k \rho_{k,l} \sigma_l$ in the off-diagonal elements.

Let us look at how the shape of the hyperellipsoidal level contours changes with changing correlation and standard deviation values. Fig. 30 shows that the half-axes of the ellipsoids coincide with the coordinate axes in case of uncorrelated parameters.

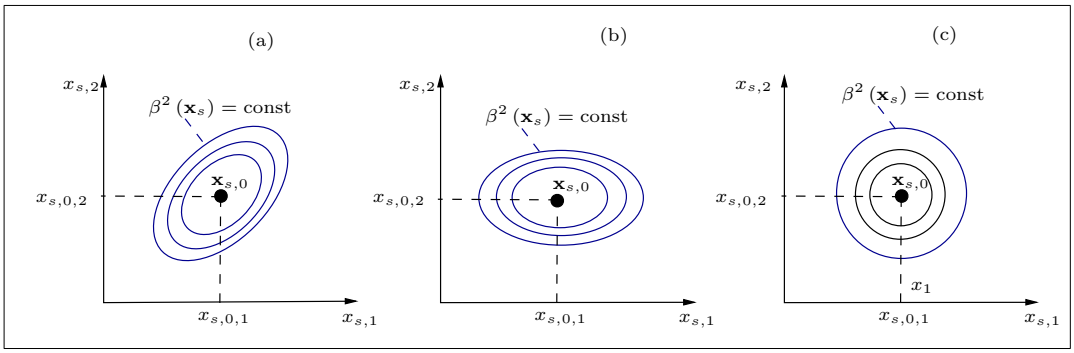


Figure 30. Level sets of a two-dimensional normal pdf with general covariance matrix $\mathbf{C}$, (a), with uncorrelated components, $\mathbf{R} = \mathbf{I}$, (b), and uncorrelated components of equal spread, $\mathbf{C} = \sigma^2 \cdot \mathbf{I}$, (c).

This corresponds to a diagonal covariance matrix. If, in addition, the variances of the parameters have the same variance, we obtain concentric circles around the mean value vector, which refers to the covariance matrix as an identity matrix multiplied by the common variance value. Another property of the shape of the matrix results from taking a specific value of β and then sweeping the parameter's correlation. Fig. 31 illustrates this for two statistical parameters.

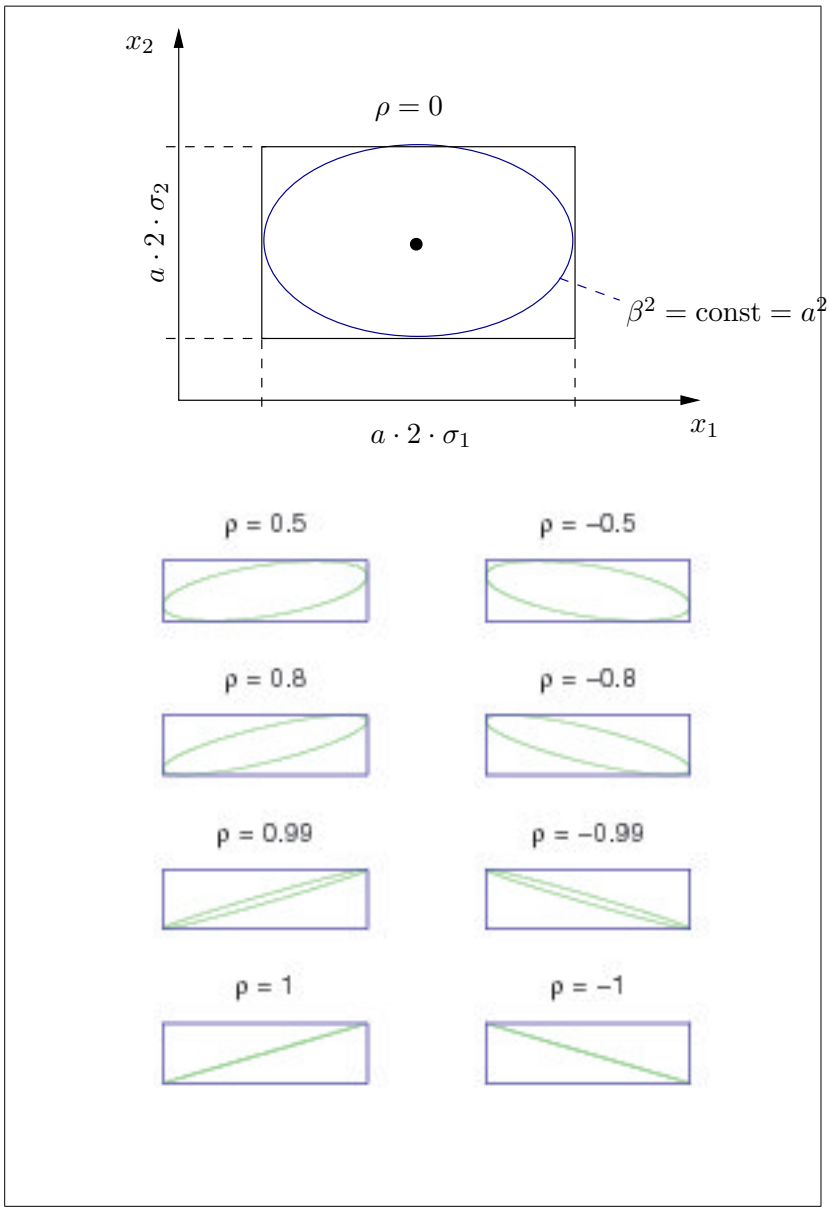


Figure 31. Level set with $\beta^2 = a^2$ of a two-dimensional normal pdf for different correlation coefficient values, ρ.

We can see that the shape of the hyperellipsoid becomes more and more narrow towards the north-east and south-west in case of a positive correlation and north-west and south-east for a negative correlation. This corresponds to a positive (negative) correlation that biases the distribution towards more similar values of the two parameters of equal (opposite) sign. The exciting feature is that the hyperellipsoid always stays within the same hyperrectangle. This can be interpreted as the hyperrectangle being the worst-case level contour to catch any possible value of an unknown correlation.

7.3 Transformation of statistical distributions

We have seen that the level contours of the multivariate normal distribution are hyperellipsoids described by a quadratic function. We also know from our excursion into the optimization theory that a quadratic function is advantageous for optimization. We will therefore describe and develop our methods for yield analysis and yield optimization/design centering based on a multivariate normal distribution. In practice, however, the parameter distribution may be of any other type. For instance, if physical parameters cannot be negative, like oxide thickness, then a normal distribution that attributes probabilities to the complete range of real numbers is not suitable. In such cases, we use the fact that any statistical distribution type can be transformed into another type. The approach for transforming statistical distributions will be explained in the following.

Assume two random variable vectors, $\mathbf{y} \in \mathbb{R}^{n_y}$ and $\mathbf{z} \in \mathbb{R}^{n_z}$, which have the same number of random variables $n_y = n_z$. Assume further that there is a bijective and smooth mapping between the two random variables, such that $\mathbf{z} = \mathbf{z}(\mathbf{y})$ and $\mathbf{y} = \mathbf{y}(\mathbf{z})$ (more precisely $\mathbf{z} = \boldsymbol{\rho}(\mathbf{y})$, $\mathbf{y} = \boldsymbol{\rho}^{-1}(\mathbf{z})$).

Then, we can transform the cumulative distribution function $cdf_y(\mathbf{y})$ of type y of the random variable vector $\mathbf{y}$ into the cumulative distribution function $cdf_z(\mathbf{z})$ of type z of the random variable vector $\mathbf{z}$ by means of **variable substitution in multiple integration** as follows:

$$\begin{aligned} cdf_y(\mathbf{y}) &= \int_{-\infty}^{\mathbf{y}} \cdots \int pdf_y(\mathbf{y}') \cdot d\mathbf{y}' = \int_{-\infty}^{\mathbf{z}(\mathbf{y})} \cdots \int pdf_y(\mathbf{y}'(\mathbf{z}')) \cdot \left| \det\left(\frac{\partial \mathbf{y}}{\partial \mathbf{z}^T}\right) \right| \cdot d\mathbf{z}' \\ &= \int_{-\infty}^{\mathbf{z}} \cdots \int pdf_z(\mathbf{z}') \cdot d\mathbf{z}' = cdf_z(\mathbf{z}) \end{aligned} \tag{222}$$

We obtain two formulas for transforming statistical distributions:

$$\boxed{\int_{-\infty}^{\mathbf{y}} \cdots \int pdf_y(\mathbf{y}') \cdot d\mathbf{y}' = \int_{-\infty}^{\mathbf{z}} \cdots \int pdf_z(\mathbf{z}') \cdot d\mathbf{z}'} \tag{223}$$

$$\boxed{pdf_z(\mathbf{z}) = pdf_y(\mathbf{y}(\mathbf{z})) \cdot \left| \det\left(\frac{\partial \mathbf{y}}{\partial \mathbf{z}^T}\right) \right|} \tag{224}$$

$$\text{univariate case:} \quad pdf_z(z) = pdf_y(y(z)) \cdot \left| \frac{\partial y}{\partial z} \right| \tag{225}$$

The first formula is suitable if two probability density functions are given, and we are searching for the mapping function between these two probability density functions. The second formula is suitable if one probability density function and the mapping function are given, and we are searching for the probability density function that results from the mapping.

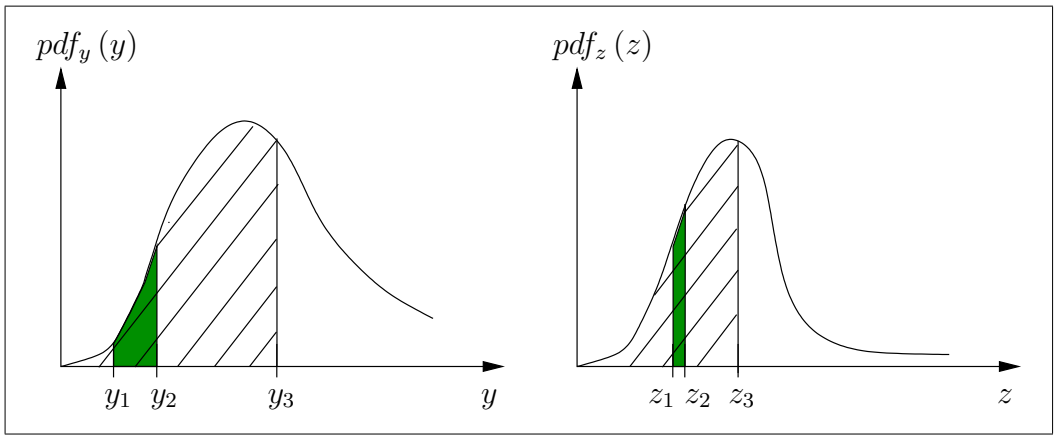


Figure 32. Univariate pdf with random number y is transformed to a new pdf of new random number $z = z(y)$. According to (223), the shaded and hatched areas under the curve are equal.

In the univariate case, the function pdf_z has a domain that is a scaled version of the domain of pdf_y. $\left|\frac{\partial y}{\partial z}\right|$ determines the scaling factor. This is illustrated in Fig. 32. The three values y_1, y_2, y_3 are mapped onto the values of z with the same index, i.e., z_1, z_2, z_3, and vice versa. (223) says that the area under the pdf from $-\infty$ to the respective point is identical for both pdf_y and pdf_z. Hence, the hatched areas are identical. Building differences from the cdf values, the two shaded areas are also identical. Making the shaded area infinitesimally small, we have the two pdf values scaled by $\left|\frac{\partial y}{\partial z}\right|$.

In high-order cases, the random variable space is scaled and rotated with the Jacobian matrix $\left|\det\left(\frac{\partial \mathbf{y}}{\partial \mathbf{z}^T}\right)\right|$ determining the scaling and rotation.

We will now use (223) to transform a univariate uniform distribution into an arbitrary distribution and then extend this to create a multivariate normal distribution sample. This is, e.g., happening in a Monte-Carlo analysis or any other statistical method that has to simulate samples of given distributions.

Transforming a univariate uniform distribution into arbitrary univariate distribution

Given are the probability density function $pdf_U(z)$ of a uniform distribution

$$pdf_U(z) = \begin{cases} 1 \text{ for } 0 < z < 1 \\ 0 \text{ otherwise} \end{cases} \tag{226}$$

and another probability density function $pdf_y(y)$, $y \in \mathbb{R}$.

What is the mapping function to obtain a random number z that is distributed according to (223) from a uniformly distributed random number z distributed according to (226)?

We first clarify the domain bounds of the two random variables to be mapped onto each other. These are:

z	y
0	$-\infty$
1	$+\infty$

From (223), we obtain:

$$\int_0^z \underbrace{pdf_z(z')}_{\substack{1 \text{ for } 0 \le z \le 1 \\ \text{from (226)}}} \cdot dz' = \int_{-\infty}^{y} pdf_y(y') \cdot dy' \tag{227}$$

and hence:

$$z = \int_{-\infty}^{y} pdf_y(y') \cdot dy' = cdf_y(y) \tag{228}$$

The overall mapping function $y = \rho^{-1}(z)$ then is

$$\boxed{y = cdf_y^{-1}(z)} \tag{229}$$

The transformation process now is:

- Beforehand:
  - Insert $pdf_y(y)$ in (228).
  - Compute cdf_y by integration.
  - Compute the inverse cdf_y^{-1}.
- Creating sample element y:
  - Create uniform random number z according to the distribution given by $pdf_z(z)$.
  - Insert obtained z into (229) to get sample value, y, distributed according to $pdf_y(y)$.

Generating multivariate normally distributed sample from univariate uniform distribution

We can use the approach just outlined and adapt it in the following way:

- First, we create a set of statistically independent, uniformly distributed random numbers.
- Second, the computation of the *cdf* and the inverse cdf^{-1} are already available for a standard normal distribution, i.e., a normal distribution with mean value 0 and standard deviation 1. In any book on statistics, there are tables for these functions.

- Third, anticipating derivations done in the subsequent chapter, a multivariate standard normal distribution (with zero mean values and standard deviations all one) can be transformed into a normal distribution with arbitrary mean value vector and covariance matrix $\mathbf{C}$ by a linear transformation.

First, we generate $n_{\mathbf{x}_s}$ independent uniformly distributed random numbers z_k, $k = 1, \ldots, n_{\mathbf{x}_s}$: $\mathbf{z} = [\ldots z_k \ldots]^T_{<n_{\mathbf{x}_s}>}$. This can be done, e.g., in hardware by a linear feedback shift register or in software by arithmetic random number generators. Corresponding uniform random functions are available in most software libraries. Please note that we do not create random numbers in this way, but pseudorandom numbers. It is a research area of its own to create pseudorandom number generators that are as random as possible.

Second, we map each z_k onto a univariate standard normally distributed y_k with the help of the respective statistical tables:

$$z_k \sim U(0,1) \quad \rightarrow \quad y_k \sim N(0,1) \tag{230}$$

For instance, $z_k = 0.6950$ from a uniform distribution maps onto $y_k = 0.51$ from a standard normal distribution, $z_k = 0.9406$ from a uniform distribution maps onto $y_k = 1.56$. For more details, please see the tutorial.

This is done for each component of the standard normally distributed random number vector $\mathbf{y} = [\ldots y_k \ldots]^T_{<n_{\mathbf{x}_s}>} \sim N(\mathbf{0}, \mathbf{I})$.

Third, we map the standard normally distributed random number vector $\mathbf{y}$ on the normally distributed random number vector $\mathbf{x}_s \sim N(\mathbf{x}_{s,0}, \mathbf{C})$ by

$$\mathbf{x}_s = \mathbf{A} \cdot \mathbf{y} + \mathbf{x}_{s,0} \quad , \quad \mathbf{C} = \mathbf{A} \cdot \mathbf{A}^T \tag{231}$$

This formula is derived in the next chapter. The matrix decomposition $\mathbf{A} \cdot \mathbf{A}^T$ of the covariance matrix $\mathbf{C}$ is preferably computed through an eigenvalue decomposition.

8 Expectation values and their estimation

The definition of yield and the methods for yield analysis and yield optimization will be based on the concept of expectation values and their estimation. This chapter is dedicated to describing all the necessary formulas. In particular, it will present multivariate versions of expectation values and estimation.

8.1 Expectation values

8.1.1 Definitions

We introduce $\mathbf{h}(\mathbf{z})$ as a deterministic multivariate function of a random variable vector $\mathbf{z}$ that is distributed according to the probability density function $pdf(\mathbf{z})$.

Expectation value

The expectation value $E\{\cdot\}$ of the deterministic function $\mathbf{h}(\mathbf{z})$ of the random variable vector $\mathbf{z}$ with regard to the probability density function $pdf(\mathbf{z})$ is defined as the multiple integral of the function times the probability density function:

$$\underset{pdf(\mathbf{z})}{E}\{\mathbf{h}(\mathbf{z})\} = E\{\mathbf{h}(\mathbf{z})\} = \int_{-\infty}^{+\infty} \cdots \int \mathbf{h}(\mathbf{z}) \cdot pdf(\mathbf{z}) \cdot d\mathbf{z} \tag{232}$$

The expectation value (or: expected value) can be seen as an operator that consists of a specific integration function. There are particular functions whose expectation leads to statistical key figures. Examples follow.

Moment of order κ

The moment of order κ of the univariate random variable z is the expectation value of z^κ:

$$m^{(\kappa)} = E\{z^\kappa\} \tag{233}$$

Mean value (first-order moment)

The first-order moment is obtained for $\kappa = 1$ and defines the mean value m

$$m^{(1)} = m = E\{z\} \tag{234}$$

Instead of m, the mean value is often denoted with the letter μ.

The mean value vector $\mathbf{m}$ is defined as the vector of the individual mean values of its components:

$$\mathbf{m} = E\{\mathbf{z}\} = \begin{bmatrix} E\{z_1\} \\ \vdots \\ E\{z_{n_z}\} \end{bmatrix} \tag{235}$$

Central moment of order κ

The central moment of order κ of the univariate random variable z is the moment of order κ of z subtracted by its mean value. The central moment of order 1 is zero:

$$\begin{aligned} c^{(\kappa)} &= E\left\{(z-m)^{\kappa}\right\} \\ c^{(1)} &= 0 \end{aligned} \tag{236}$$

Variance (second-order central moment)

The second-order central moment yields the variance of the univariate random number z. It is also denoted as $V\{\cdot\}$:

$$c^{(2)} = E\left\{(z-m)^2\right\} = \sigma^2 = V\{z\} \tag{237}$$

The square root of the variance is the standard deviation σ. It is important to note that the standard deviation has the same physical unit as the random variable. Therefore, It is the standard deviation used for a technical interpretation.

Covariance

The covariance between two random variables z_i and z_j is defined as:

$$cov\left\{z_i, z_j\right\} = E\left\{\left(z_i - m_i\right)\left(z_j - m_j\right)\right\} \tag{238}$$

Regarding quadratic models, the variance corresponds to the quadratic part in one variable, and the covariance corresponds to the mixed quadratic part in two variables. Analogous to the Hessian matrix, we can combine all variances and covariances of a vector of random variables in one matrix, which is denoted as variance/covariance matrix and usually abbreviated as covariance matrix:

Variance/covariance matrix

$$\begin{aligned} \mathbf{C} &= V\{\mathbf{z}\} = E\left\{(\mathbf{z}-\mathbf{m})\cdot(\mathbf{z}-\mathbf{m})^T\right\} \\ &= \begin{bmatrix} V\{z_1\} & cov\{z_1,z_2\} & \cdots & cov\{z_1,z_{n_z}\} \\ cov\{z_2,z_1\} & V\{z_2\} & \cdots & cov\{z_2,z_{n_z}\} \\ \vdots & & \ddots & \vdots \\ cov\{z_{n_z},z_1\} & cov\{z_{n_z},z_2\} & \cdots & V\{z_{n_z}\} \end{bmatrix} \end{aligned} \tag{239}$$

We cannot only write the covariance matrix of the multivariate random variable vector but also the covariance matrix $V\{\mathbf{h}(\mathbf{z})\}$ of the multivariate deterministic function $\mathbf{h}(\mathbf{z})$ of a random variable vector:

$$V\{\mathbf{h}(\mathbf{z})\} = E\left\{\left(\mathbf{h}(\mathbf{z}) - E\{\mathbf{h}(\mathbf{z})\}\right)\cdot\left(\mathbf{h}(\mathbf{z}) - E\{\mathbf{h}(\mathbf{z})\}\right)^T\right\} \tag{240}$$

8.1.2 Expectation value of linear transformation

The expectation value of a linear mapping can be reformulated according to the following rule:

$$\boxed{E\left\{\mathbf{A}\cdot\mathbf{h}\left(\mathbf{z}\right)+\mathbf{b}\right\}=\mathbf{A}\cdot E\left\{\mathbf{h}\left(\mathbf{z}\right)\right\}+\mathbf{b}} \tag{241}$$

The proofs of this and the following two rules are given in the tutorial. The proof uses the fact that the expectation operator is an integral, from which constants can be removed.

There are the following special cases of this general multivariate rule:

$$\begin{aligned} E\left\{c\right\} &= c, \text{ c is a constant} \\ E\left\{c\cdot\mathbf{h}\left(\mathbf{z}\right)\right\} &= c\cdot E\left\{\mathbf{h}\left(\mathbf{z}\right)\right\} \\ E\left\{h_1\left(\mathbf{z}\right)+h_2\left(\mathbf{z}\right)\right\} &= E\left\{h_1\left(\mathbf{z}\right)\right\}+E\left\{h_2\left(\mathbf{z}\right)\right\} \end{aligned}$$

8.1.3 Variance of linear transformation

The variance of a linear mapping can be reformulated according to the following rule:

$$\boxed{V\left\{\mathbf{A}\cdot\mathbf{h}\left(\mathbf{z}\right)+\mathbf{b}\right\}=\mathbf{A}\cdot V\left\{\mathbf{h}\left(\mathbf{z}\right)\right\}\cdot\mathbf{A}^T} \tag{242}$$

The result has to do with the formulation of the variance as an expectation of quadratic forms of the random variable. The multiplicating matrix will then go out of the integral twice, and the constant disappears because of the central moment forming.

There are the following special cases of this general multivariate law:

$$\begin{aligned} V\left\{\mathbf{a}^T\cdot\mathbf{h}\left(\mathbf{z}\right)+b\right\} &= \mathbf{a}^T\cdot V\left\{\mathbf{h}\left(\mathbf{z}\right)\right\}\cdot\mathbf{a} \\ V\left\{a\cdot h\left(z\right)+b\right\} &= a^2\cdot V\left\{h(z)\right\} \end{aligned}$$

The Gaussian error propagation, or, measurement error propagation, is another special case:

$$V\left\{\mathbf{a}^T\cdot\mathbf{z}+b\right\}=\mathbf{a}^T\cdot\mathbf{C}\cdot\mathbf{a}=\sum_{i,j}a_i a_j\sigma_i\rho_{i,j}\sigma_j \underset{\substack{\forall \\ i\neq j}}{=}{}^{\rho_{i,j}=0}\sum_i a_i^2\sigma_i^2$$

8.1.4 Translation law of variances

The translation law of variances reads as:

$$\boxed{V\left\{\mathbf{h}\left(\mathbf{z}\right)\right\} = E\left\{\left(\mathbf{h}\left(\mathbf{z}\right) - \mathbf{a}\right) \cdot \left(\mathbf{h}\left(\mathbf{z}\right) - \mathbf{a}\right)^T\right\} - \left(E\left\{\mathbf{h}\left(\mathbf{z}\right)\right\} - \mathbf{a}\right) \cdot \left(E\left\{\mathbf{h}\left(\mathbf{z}\right)\right\} - \mathbf{a}\right)^T} \tag{243}$$

It says that the variance does not have to be formulated as central moment, i.e., with regard to the mean values, but that it can be formulated with regard to any constant vector $\mathbf{a}$ if then the mean value vector minus the constant vector are subtracted as a dyadic product.

There are the following special cases of this general multivariate law:

$$\begin{aligned} V\left\{h\left(\mathbf{z}\right)\right\} &= E\left\{\left(h\left(\mathbf{z}\right) - a\right)^2\right\} - \left(E\left\{h\left(\mathbf{z}\right)\right\} - a\right)^2 \\ V\left\{\mathbf{h}\left(\mathbf{z}\right)\right\} &= E\left\{\mathbf{h}\left(\mathbf{z}\right) \cdot \mathbf{h}^T\left(\mathbf{z}\right)\right\} - E\left\{\mathbf{h}\left(\mathbf{z}\right)\right\} \cdot E\left\{\mathbf{h}^T\left(\mathbf{z}\right)\right\} \\ V\left\{h\left(\mathbf{z}\right)\right\} &= E\left\{\left(h^2\left(\mathbf{z}\right)\right)\right\} - \left(E\left\{h\left(\mathbf{z}\right)\right\}\right)^2 \end{aligned}$$

8.1.5 Normalizing a random variable

The goal of normalizing a random variable is to take out the mean value and the standard deviation and transform it into a random variable with zero mean and standard deviation one. The formula for this is to subtract the mean value and divide by the standard deviation:

$$z' = \frac{z - E\left\{z\right\}}{\sqrt{V\left\{z\right\}}} = \frac{z - m_z}{\sigma_z} \tag{244}$$

While this is intuitively suggesting itself, we can prove it using the laws just stated:

$$E\left\{z'\right\} = \frac{E\left\{z\right\} - m_z}{\sigma_z} = \frac{m_z - m_z}{\sigma_z} = 0 \tag{245}$$

$$V\left\{z'\right\} = E\left\{\left(z' - 0\right)^2\right\} = \frac{E\left\{\left(z - m_z\right)^2\right\}}{\sigma_z^2} = \frac{\sigma_z^2}{\sigma_z^2} = 1 \tag{246}$$

8.1.6 Linear transformation of a normal distribution

In the following, we leave out the index s that usually denotes statistical parameters.

Assume that a parameter vector $\mathbf{x}$ is normally distributed,

$$\mathbf{x} \sim N(\mathbf{x}_0, \mathbf{C})$$

and is linearly mapped onto a performance $\overline{\varphi}$:

$$\boxed{\overline{\varphi}(\mathbf{x}) = \varphi_a + \mathbf{g}^T \cdot (\mathbf{x} - \mathbf{x}_a)} \tag{247}$$

The mean value $\mu_{\overline{\varphi}}$ of the linearized performance $\overline{\varphi}$ can be calculated based on the expectation value of a linear transformation, or "on foot":

$$\begin{aligned} \mu_{\overline{\varphi}} &= E\{\overline{\varphi}\} = E\left\{\varphi_a + \mathbf{g}^T \cdot (\mathbf{x} - \mathbf{x}_a)\right\} \\ &= E\{\varphi_a\} + \mathbf{g}^T \cdot (E\{\mathbf{x}\} - E\{\mathbf{x}_a\}) \end{aligned}$$

This finally leads to:

$$\boxed{\mu_{\overline{\varphi}} = \varphi_a + \mathbf{g}^T \cdot (\mathbf{x}_0 - \mathbf{x}_a)} \tag{248}$$

Correspondingly, the variance $\sigma^2_{\overline{\varphi}}$ of the linearized performance $\overline{\varphi}$ can be calculated based on the variance of a linear transformation, or "on foot":

$$\begin{aligned} \sigma^2_{\overline{\varphi}} &= E\left\{(\overline{\varphi} - \mu_{\overline{\varphi}})^2\right\} = E\left\{\left(\mathbf{g}^T(\mathbf{x} - \mathbf{x}_0)\right)^2\right\} \\ &= E\left\{\mathbf{g}^T \cdot (\mathbf{x} - \mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0)^T \cdot \mathbf{g}\right\} \\ &= \mathbf{g}^T \cdot E\left\{(\mathbf{x} - \mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0)^T\right\} \cdot \mathbf{g} \end{aligned}$$

Here, we have transformed a product of two scalar products of vectors into an inner dyadic product, multiplied to a scalar by multiplication of a row vector from left and a column vector from right. The internal expectation value is the definition of the covariance matrix of parameters, so we finally get:

$$\boxed{\sigma^2_{\overline{\varphi}} = \mathbf{g}^T \cdot \mathbf{C} \cdot \mathbf{g}} \tag{249}$$

This is precisely the formula we have used in the realistic worst-case analysis to interpret the level contour size β of the tolerance ellipsoid as a multiple of the standard deviation of the linearized performance.

Please note that this does not prove that the linearized performance is normally distributed. The proof will be given in the tutorial.

8.2 Estimation of expectation values

The formulation of the expectation values is part of probability theory. In practice, these values often have to be estimated based on a sample, and the corresponding formulas are part of statistics. In the following, estimators for the expectation value and the variance will be given. Then, the accuracy of estimating an expectation value will be discussed. Finally, the three laws from the previous section will be formulated for estimations.

The estimation is based on a sample of the population with n_{MC} sample elements $\mathbf{x}^{(\mu)}$, $\mu = 1, \ldots, n_{MC}$ (i.e., a sample of sample size n_{MC}), that are distributed according to the distribution $\mathcal{D}$ defined by the probability density function:

$$\mathbf{x}^{(\mu)} \sim \mathcal{D}\left(pdf\left(\mathbf{x}\right)\right), \quad \mu = 1, \ldots, n_{MC}$$

The sample elements $\mathbf{x}^{(\mu)}, \mu = 1, \cdots, n_{MC}$, are **independently and identically distributed**. It follows that all sample elements have the same variance and covariance, and the covariance among sample elements is zero:

$$E\left\{\mathbf{h}\left(\mathbf{x}^{(\mu)}\right)\right\} = E\left\{\mathbf{h}\left(\mathbf{x}\right)\right\} = \mathbf{m}_h \tag{250}$$

$$V\left\{\mathbf{h}\left(\mathbf{x}^{(\mu)}\right)\right\} = V\left\{\mathbf{h}\left(\mathbf{x}\right)\right\}, \ \operatorname{cov}\left\{\mathbf{h}\left(\mathbf{x}^{(\mu)}\right), \mathbf{h}\left(\mathbf{x}^{(\nu)}\right)\right\} = \mathbf{0}, \mu \neq \nu \tag{251}$$

We can in general call $\hat{\boldsymbol{\phi}}\left(\mathbf{x}\right) = \hat{\boldsymbol{\phi}}\left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n_{MC})}\right)$ an **estimator function** for $\boldsymbol{\phi}\left(\mathbf{x}\right)$.

The **estimator bias** describes how much the expectation value of the estimator deviates from the true function:

$$\mathbf{b}_{\hat{\phi}} = E\left\{\hat{\boldsymbol{\phi}}\left(\mathbf{x}\right)\right\} - \boldsymbol{\phi}\left(\mathbf{x}\right) \tag{252}$$

An **unbiased estimator** has a bias of zero:

$$E\left\{\hat{\boldsymbol{\phi}}\left(\mathbf{x}\right)\right\} = \boldsymbol{\phi}\left(\mathbf{x}\right) \Leftrightarrow \mathbf{b}_{\hat{\phi}} = \mathbf{0} \tag{253}$$

A **consistent estimator** is obtained if the deviation of the estimator from the true function is small enough for large sample size:

$$\lim_{n_{MC} \to \infty} P\left\{\left\|\hat{\boldsymbol{\phi}} - \boldsymbol{\phi}\right\| < |\epsilon|\right\} = 1 \tag{254}$$

A **strongly consistent estimator** is obtained if this deviation becomes zero:

$$\epsilon \to 0 \tag{255}$$

The integration becomes a summation in the discrete case of an estimation. Hence, we obtain the following formulas for the estimators of expectation value and variance, which correspond to the original expectation and variance formulas.

Expectation value estimator

$$\boxed{\hat{E}\left\{\mathbf{h}\left(\mathbf{x}\right)\right\} = \hat{\mathbf{m}}_h = \frac{1}{n_{MC}} \cdot \sum_{\mu=1}^{n_{MC}} \mathbf{h}\left(\mathbf{x}^{(\mu)}\right)} \tag{256}$$

Variance estimator

$$\boxed{\hat{V}\left\{\mathbf{h}\left(\mathbf{x}\right)\right\} = \frac{1}{n_{MC}-1} \sum_{\mu=1}^{n_{MC}} \left(\mathbf{h}\left(\mathbf{x}^{(\mu)}\right) - \hat{\mathbf{m}}_h\right) \cdot \left(\mathbf{h}\left(\mathbf{x}^{(\mu)}\right) - \hat{\mathbf{m}}_h\right)^T} \tag{257}$$

This formula provides an unbiased estimator in case an estimated mean value $\hat{\mathbf{m}}_h$ is available.

If the real mean value $\mathbf{m}_h$ was available instead of the estimated mean value $\hat{\mathbf{m}}_h$, the unbiased estimator would be:

$$\hat{\hat{V}}\left\{\mathbf{h}\left(\mathbf{x}\right)\right\} = \frac{1}{n_{MC}} \sum_{\mu=1}^{n_{MC}} \left(\mathbf{h}\left(\mathbf{x}^{(\mu)}\right) - \mathbf{m}_h\right) \cdot \left(\mathbf{h}\left(\mathbf{x}^{(\mu)}\right) - \mathbf{m}_h\right)^T \tag{258}$$

8.2.1 Variance of the expectation value estimator

Please note that an estimation is done for a specific sample. If the estimation is repeated, another sample with different size and/or sample elements will be used, leading to a different result. The estimation is, therefore, a statistical process, and the first-order characterization of the estimation quality is the estimator's variance. A large variance means that the spread of possible outcomes of the estimation process is large, which means a large interval of confidence, which in turn means that the true value may be in a large interval of values, which is not what we want. A high quality of an estimation corresponds to a small variance of the expectation value estimator.

Please note that the variance in the difference between the estimator function and the true function also depends on the bias. If the bias is zero, as is the case in the formulas given above, it is just the variance of the estimator itself we have to look at:

$$E\left\{\left(\hat{\boldsymbol{\phi}}-\boldsymbol{\phi}\right)\cdot\left(\hat{\boldsymbol{\phi}}-\boldsymbol{\phi}\right)^{T}\right\}=V\left\{\hat{\boldsymbol{\phi}}\right\}+\mathbf{b}_{\hat{\phi}}\cdot\mathbf{b}_{\hat{\phi}}^{T}\overset{\mathbf{b}_{\hat{\phi}}=\mathbf{0}}{=}V\left\{\hat{\boldsymbol{\phi}}\right\}\tag{259}$$

In the following, the variance of the estimated expectation of a deterministic function of a random variable $\mathbf{h}(\mathbf{x})$ is derived:

$$V\left\{\hat{\mathbf{m}}_h\right\}=V\left\{\hat{E}\left\{\mathbf{h}(\mathbf{x})\right\}\right\}=V\left\{\frac{1}{n_{MC}}\cdot\sum_{\mu=1}^{n_{MC}}\mathbf{h}\left(\mathbf{x}^{(\mu)}\right)\right\}\overset{(242)}{=}\frac{1}{n_{MC}^2}\cdot V\left\{\sum_{\mu=1}^{n_{MC}}\mathbf{I}_{\langle n_h,n_h\rangle}\cdot\mathbf{h}^{(\mu)}\right\}$$

Here, $\mathbf{h}^{(\mu)}=\mathbf{h}\left(\mathbf{x}^{(\mu)}\right)$, and $\mathbf{I}_{\langle n_h,n_h\rangle}$ is the identity matrix of size n_h of $\mathbf{h}^{(\mu)}$. To continue,

$$\begin{aligned}
V\left\{\hat{\mathbf{m}}_h\right\} &= \frac{1}{n_{MC}^2}\cdot V\left\{\left[\mathbf{I}_{\langle n_h,n_h\rangle}\,\mathbf{I}_{\langle n_h,n_h\rangle}\,\ldots\,\mathbf{I}_{\langle n_h,n_h\rangle}\right]\cdot\begin{bmatrix}\mathbf{h}^{(1)}\\ \mathbf{h}^{(2)}\\ \vdots\\ \mathbf{h}^{(n_{MC})}\end{bmatrix}\right\}\\
&\overset{(242)}{=} \frac{1}{n_{MC}^2}\cdot\left[\mathbf{I}_{\langle n_h,n_h\rangle}\,\mathbf{I}_{\langle n_h,n_h\rangle}\,\ldots\,\mathbf{I}_{\langle n_h,n_h\rangle}\right]\cdot V\left\{\begin{bmatrix}\mathbf{h}^{(1)}\\ \mathbf{h}^{(2)}\\ \vdots\\ \mathbf{h}^{(n_{MC})}\end{bmatrix}\right\}\cdot\begin{bmatrix}\mathbf{I}_{\langle n_h,n_h\rangle}\\ \mathbf{I}_{\langle n_h,n_h\rangle}\\ \vdots\\ \mathbf{I}_{\langle n_h,n_h\rangle}\end{bmatrix}\\
&\overset{(251)}{=} \frac{1}{n_{MC}^2}\cdot\left[\mathbf{I}_{\langle n_h,n_h\rangle}\,\mathbf{I}_{\langle n_h,n_h\rangle}\,\ldots\,\mathbf{I}_{\langle n_h,n_h\rangle}\right]\cdot\begin{bmatrix}V\{\mathbf{h}\} & & \mathbf{0}\\ & \ddots & \\ \mathbf{0} & & V\{\mathbf{h}\}\end{bmatrix}\cdot\begin{bmatrix}\mathbf{I}_{\langle n_h,n_h\rangle}\\ \mathbf{I}_{\langle n_h,n_h\rangle}\\ \vdots\\ \mathbf{I}_{\langle n_h,n_h\rangle}\end{bmatrix}\\
&= \frac{1}{n_{MC}^2}\cdot n_{MC}\cdot V\{\mathbf{h}\}
\end{aligned}$$

We finally obtain for the **standard deviation of the estimated expectation value** of the **deterministic function of the random variable $\mathbf{h}(\mathbf{x})$**:

$$\boxed{\sqrt{V\left\{\hat{\mathbf{m}}_h\right\}} = \frac{1}{\sqrt{n_{MC}}} \cdot \sqrt{V\left\{\mathbf{h}(\mathbf{x})\right\}}} \qquad \boxed{\sqrt{\hat{V}\left\{\hat{\mathbf{m}}_h\right\}} = \frac{1}{\sqrt{n_{MC}}} \cdot \sqrt{\hat{V}\left\{\mathbf{h}(\mathbf{x})\right\}}} \tag{260}$$

The second part holds for the **estimated standard deviation of the estimated expectation value** of the **deterministic function of the random variable $\mathbf{h}(\mathbf{x})$**. It is obtained by replacing $V\{\mathbf{h}\}$ by $\hat{V}\{\mathbf{h}\}$ and (242) by (262) in the derivation.

The formulas should be written with regard to the standard deviation, as this is the variable with a physical unit and a technical interpretation, as mentioned before. (260) tells us that a problem-inherent component of the estimation standard deviation is the standard deviation of the original function. This is outside the influence of the estimation.

The original standard deviation is **divided by the square root of the sample size** to obtain the standard deviation of the estimation.

The good news is that the estimation quality by sampling does NOT depend on the dimension of the parameter space. It does not matter if we estimate a space with ten parameters or a space with 10,000 parameters, the estimation accuracy is the same.

The bad news is that to obtain a ten times better accuracy, we need 100 times more sample elements. This is the disadvantage of estimation by sampling; we will see that it leads to dramatic sample size in case of estimation of rare events, which is the case if very high robustness of circuits/systems shall be evaluated.

In the following, the corresponding transformation formulas in the case of estimation will be given. The proofs are provided in the tutorial.

8.2.2 Estimated expectation value of linear transformation

$$\boxed{\hat{E}\left\{\mathbf{A} \cdot \mathbf{h}\left(\mathbf{z}\right) + \mathbf{b}\right\} = \mathbf{A} \cdot \hat{E}\left\{\mathbf{h}\left(\mathbf{z}\right)\right\} + \mathbf{b}} \tag{261}$$

8.2.3 Estimated variance of linear transformation

$$\boxed{\hat{V}\left\{\mathbf{A} \cdot \mathbf{h}\left(\mathbf{z}\right) + \mathbf{b}\right\} = \mathbf{A} \cdot \hat{V}\left\{\mathbf{h}\left(\mathbf{z}\right)\right\} \cdot \mathbf{A}^T} \tag{262}$$

8.2.4 Translation law of estimated variance

$$\boxed{\hat{V}\left\{\mathbf{h}\left(\mathbf{z}\right)\right\} = \frac{n_{MC}}{n_{MC}-1}\left[\hat{E}\left\{\mathbf{h}\left(\mathbf{z}\right) \cdot \mathbf{h}^T\left(\mathbf{z}\right)\right\} - \hat{E}\left\{\mathbf{h}\left(\mathbf{z}\right)\right\} \cdot \hat{E}\left\{\mathbf{h}^T\left(\mathbf{z}\right)\right\}\right]} \tag{263}$$

9 Yield analysis

9.1 Problem formulation

In circuit/system design, it is roughly distinguished between catastrophic yield and parametric yield. Examples of catastrophic yield are spot defects on wafers during integrated circuit manufacturing, which lead to complete circuit failure. Parametric yield refers to variations of circuit parameters that cause parametric variation in the performance, such that the performance specification is sometimes violated. In the following, we will deal with parametric yield, leaving out the annotation of "parametric".

The yield is defined as the portion (between 0.0 and 1.0), **or, percentage** (between 0% and 100%), **of circuits that fulfill the performance specification:**

$$Y = \int_{\mathbf{x}_s \in A_s} \cdots \int pdf\left(\mathbf{x}_s\right) \cdot d\mathbf{x}_s \tag{264}$$

with the **acceptance region in the parameter space**, A_s,

$$A_s = \left\{\mathbf{x}_s \mid \boldsymbol{\varphi}\left(\mathbf{x}_s\right) \in A_\varphi\right\} \tag{265}$$

the **acceptance region in the performance space**, A_φ,

$$A_\varphi = \left\{\boldsymbol{\varphi} \mid \boldsymbol{\varphi}_L \leq \boldsymbol{\varphi} \leq \boldsymbol{\varphi}_U\right\} \tag{266}$$

and the probability density function in the parameter space given as or transformed into a normal distribution,

$$pdf_s\left(\mathbf{x}_s\right) \triangleq pdf_N\left(\mathbf{x}_s\right) = \frac{1}{\sqrt{2\pi}^{n_{xs}} \cdot \sqrt{det\left(\mathbf{C}\right)}} \cdot e^{-\frac{1}{2}\beta^2\left(\mathbf{x}_s\right)} \tag{267}$$

$$\beta^2\left(\mathbf{x}_s\right) = \left(\mathbf{x}_s - \mathbf{x}_{s,0}\right)^T \cdot \mathbf{C}^{-1} \cdot \left(\mathbf{x}_s - \mathbf{x}_{s,0}\right) \tag{268}$$

The statistical parameter distribution is due to the manufacturing process and describes its inevitable variability. The performance specification is due to the customer, in-house or outside the company, and what has been promised as performance.

To compute the yield, a multiple integral has to be solved. An additional problem arises because the integrand is defined in the statistical parameter space, but the integration bounds are defined in the performance space. As mapping between parameter space and performance space, only an expensive, point-wise, numerical simulation is available. Fig. 33 illustrates the situation with two statistical parameters on the left and two performance features on the right. The green ellipses on the left are the level contours of the known parameter distribution. The blue lines marked with one to four slashes on the right illustrate the four known bounds of the performance specification. The known items are shown with solid lines in Fig. 33, the unknown items, i.e., the specification bounds in the parameter space and the statistical performance distribution in the performance space, are shown with dashed lines. The yield analysis requires to model the statistical distribution in the performance space or the specification bounds in the parameter space.

A statistical approach for yield estimation and a deterministic approach for yield approximation will be presented in the following.

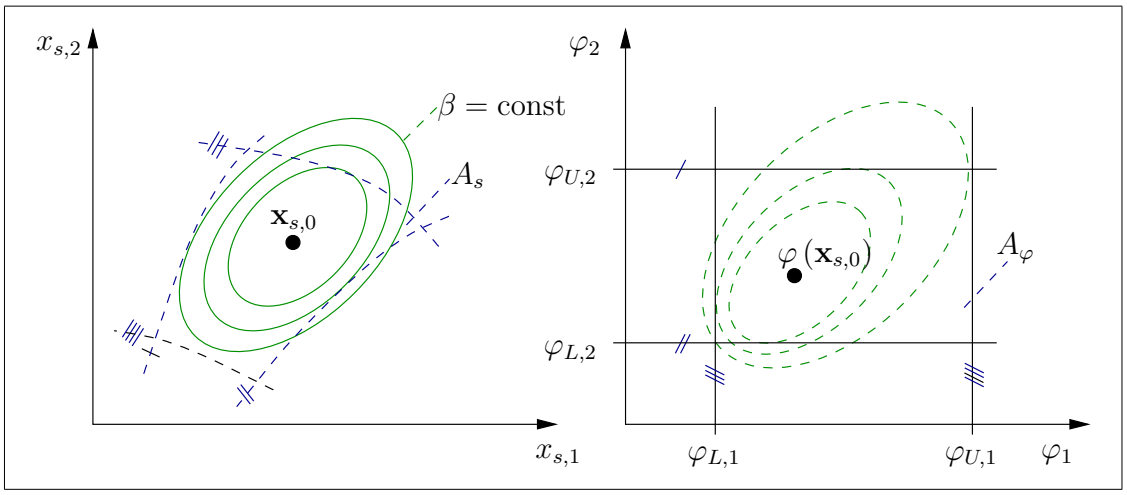


Figure 33. Statistical distributions and specifications in the parameter and performance space.

9.2 Statistical yield estimation/Monte-Carlo analysis

The statistical approach for yield estimation uses a co-called **acceptance function**:

$$\delta(\mathbf{x}_s) = \begin{cases} 1, & \boldsymbol{\varphi}(\mathbf{x}_s) \in A_\varphi \\ 0, & \boldsymbol{\varphi}(\mathbf{x}_s) \notin A_\varphi \end{cases} = \begin{cases} 1, & \mathbf{x}_s \in A_s \quad \text{"circuit functions"} \\ 0, & \mathbf{x}_s \notin A_s \quad \text{"circuit malfunctions"} \end{cases} \tag{269}$$

The crux with the definition of an acceptance function is that the **integration bounds are transformed into a function inside the integral of yield estimation** (264), with the effect that the outer integration can be formulated over the complete space of real numbers,

$$Y = \int_{A_s} \cdots \int pdf(\mathbf{x}_s) \cdot d\mathbf{x}_s \tag{270}$$

$$= \int_{-\infty}^{\infty} \cdots \int \delta(\mathbf{x}_s) \cdot pdf(\mathbf{x}_s) \cdot d\mathbf{x}_s \tag{271}$$

$$= E\{\delta(\mathbf{x}_s)\} \tag{272}$$

which leads to the result that the yield can be formulated as the **expectation value of the acceptance function**. This opens the opportunity for a statistical yield estimation, whose process is summarized in the following. This method is also called **Monte-Carlo analysis**.

- Create a sample of statistical parameter vectors according to the given distribution:

$$\mathbf{x}_s^{(\mu)} \sim N(\mathbf{x}_{s,0}, \mathbf{C}), \quad \mu = 1, \ldots, n_{MC} \tag{273}$$

- Perform (numerical) circuit simulation of each sample element (this can be interpreted as a simulation of the stochastic manufacturing process on the circuit/system level):

$$\mathbf{x}_s^{(\mu)} \mapsto \boldsymbol{\varphi}^{(\mu)} = \boldsymbol{\varphi}\left(\mathbf{x}_s^{(\mu)}\right) \ , \ \mu = 1, \ldots, n_{MC} \tag{274}$$

- Evaluate the acceptance function for each sample element (this can be interpreted as a simulation of the production test on the circuit/system level):

$$\delta^{(\mu)} = \delta\left(\mathbf{x}_s^{(\mu)}\right) = \begin{cases} 1, & \boldsymbol{\varphi}^{(\mu)} \in A_\varphi \\ 0, & \boldsymbol{\varphi}^{(\mu)} \notin A_\varphi \end{cases} \ , \ \mu = 1, \ldots, n_{MC} \tag{275}$$

- Estimate the yield using (256):

$$\hat{Y} = \hat{E}\left\{\delta\left(\mathbf{x}_s\right)\right\} = \frac{1}{n_{MC}} \sum_{\mu=1}^{n_{MC}} \delta^{(\mu)} \tag{276}$$

$$= \frac{\text{number of functioning circuits}}{\text{sample size}} \tag{277}$$

$$= \frac{n_+}{n_{MC}} = \frac{\#\{\text{"+"}\}}{\#\{\text{"+"}\} + \#\{\text{"-"}\}} \quad \text{(Fig. 34)} \tag{278}$$

Fig. 34 illustrates the process. The sample elements drawn from the given normal distribution will reflect this distribution, i.e., the density of sample elements is high in the center. It lowers as the distance to the center grows according to the level contours. After simulation, whether each sample element fulfills or violates the performance specification is known. The "good" ones can be marked with a "+", and the bad ones with a "-". The estimated yield is the percentage of "good" samples in the sample size according to (256) and (278).

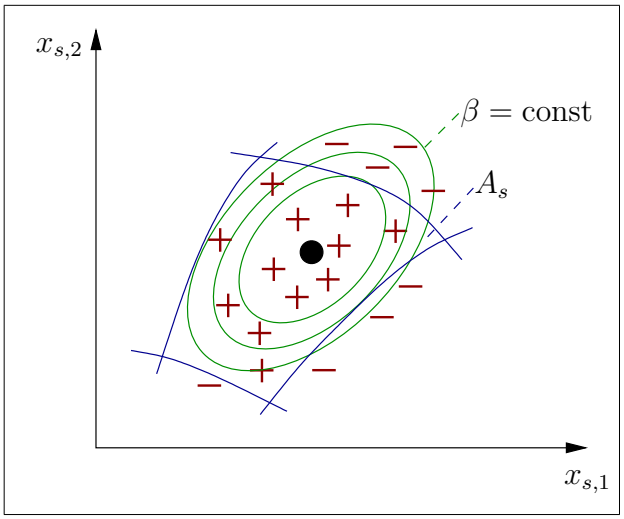


Figure 34. Monte Carlo analysis.

We will now investigate the accuracy of the statistical yield estimation and will derive a formula to determine the sample size to reach the required estimation accuracy.

9.3 Accuracy of statistical yield estimation

As described earlier, the accuracy of an estimator can be characterized by the variance of the estimation. We can derive the following formula for the **variance of the yield estimator** $\sigma^2_{\hat{Y}}$:

$$\begin{aligned}\sigma^2_{\hat{Y}} &= V\left\{\hat{Y}\right\} \overset{(276)}{=} V\left\{\hat{E}\left\{\delta\left(\mathbf{x}_s\right)\right\}\right\} \overset{(260)}{=} \frac{1}{n_{MC}} V\left\{\delta\left(\mathbf{x}_s\right)\right\} \\ &= \frac{1}{n_{MC}}[\underbrace{E\{\underbrace{\delta^2\left(\mathbf{x}_s\right)}_{\delta(\mathbf{x}_s)}\}}_{Y} - \underbrace{(\underbrace{E\left\{\delta\left(\mathbf{x}_s\right)\right\}}_{Y})^2}_{Y^2}]\end{aligned}$$

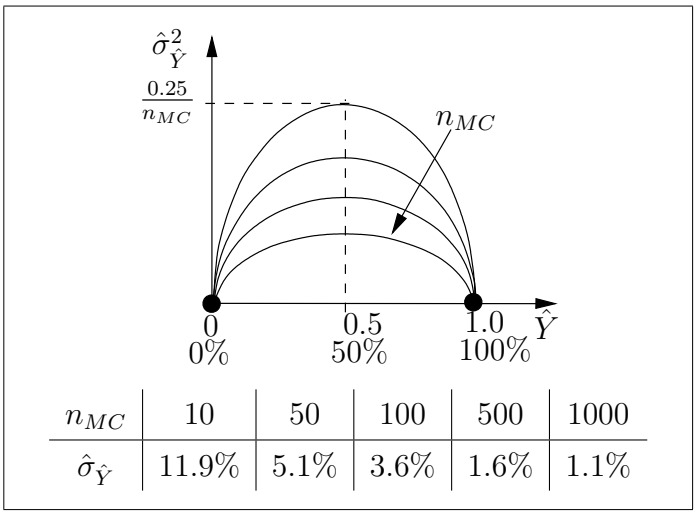
$$\boxed{\sigma^2_{\hat{Y}} = \frac{Y\left(1-Y\right)}{n_{MC}}} \tag{279}$$

Analogously, the following formula for the **estimated variance of the yield estimator** $\hat{\sigma}^2_{\hat{Y}}$ can be derived:

$$\begin{aligned}\hat{\sigma}^2_{\hat{Y}} &= \hat{V}\left\{\hat{Y}\right\} \overset{(276)}{=} \hat{V}\left\{\hat{E}\left\{\delta\left(\mathbf{x}_s\right)\right\}\right\} \overset{(260)}{=} \frac{1}{n_{MC}} \hat{V}\left\{\delta\left(\mathbf{x}_s\right)\right\} \\ &\overset{(263)}{=} \frac{1}{n_{MC}} \cdot \frac{n_{MC}}{n_{MC}-1}[\underbrace{\hat{E}\{\underbrace{\delta^2\left(\mathbf{x}_s\right)}_{\delta(\mathbf{x}_s)}\}}_{\hat{Y}} - \underbrace{(\underbrace{\hat{E}\left\{\delta\left(\mathbf{x}_s\right)\right\}}_{\hat{Y}})^2}_{\hat{Y}^2}]\end{aligned}$$

$$\boxed{\hat{\sigma}^2_{\hat{Y}} = \frac{\hat{Y}\left(1-\hat{Y}\right)}{n_{MC}-1}} \tag{280}$$

Fig. 35 illustrates (280) for an estimated yield of $\hat{Y} = 85\%$.



n_{MC}	10	50	100	500	1000
$\hat{\sigma}_{\hat{Y}}$	11.9%	5.1%	3.6%	1.6%	1.1%

Figure 35. Estimated variance of yield estimator $\hat{Y} = 85\%$.

The quadratic form of (280) leads to a maximum of the estimation's standard deviation at a yield value of 50%. This corresponds to the cumulative distribution function of a normal distribution with maximum sensitivity at that value. The estimation's standard deviation is decreasing by a factor of $\frac{1}{\sqrt{n_{MC}-1}}$. The table in Fig. 35 shows the resulting estimation's standard deviations of an estimated yield of $\hat{Y} = 85\%$ for different sample sizes. For a sample size of 1,000, the estimation's standard deviation is 1.1%.

The yield estimator $\hat{Y}$ is actually binomially distributed: it is the probability that n_+ of n_{MC} circuits are functioning. According to the central limit theorem, $\hat{Y}$ becomes normally distributed for $n_{MC} \to \infty$. In practice, a normal distribution can already be assumed for $n_+ > 4$, $n_{MC} - n_+ > 4$ and $n_{MC} > 10$.

Let us from now on assume that these conditions are fulfilled and **assume that $\hat{Y}$ is normally distributed**.

If the estimation is normally distributed, the probability ζ that an estimated value is within a certain interval, i.e., **confidence interval**, with a certain probability, i.e., **confidence level**, can be calculated from the standard normal distribution:

$$\zeta = P(Y \in \underbrace{[\hat{Y} - k_\zeta \cdot \hat{\sigma}_{\hat{Y}}, \hat{Y} + k_\zeta \cdot \hat{\sigma}_{\hat{Y}}]}_{\text{confidence interval}}) = \underbrace{\int_{-k_\zeta}^{k_\zeta} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt}_{\text{confidence level}} \tag{281}$$

Here, the confidence interval is given as k_ζ-multiple of the standard deviation. The **confidence level (i.e., the probability of being within the confidence interval)** is obtained from the cumulative distribution function of the standard normal distribution.

For an estimation's standard deviation of $\hat{\sigma}_{\hat{Y}} = 1.1\%$ for an estimated yield of $\hat{Y} = 85\%$ with sample size of $n_{MC} = 1,000$, this means that the probability to be within an interval of $\hat{Y} \pm 3\hat{\sigma}_{\hat{Y}}$, i.e., $k_\zeta = 3$, is 99.7%: $P(Y \in [81.7\%, 88.3\%]) = 99.7\%$. We can, therefore, be sure with a confidence level of 99.7% that the yield will be between 81.7% and 88.3%, which is quite a broad range for $1,000$ expensive circuit simulations.

(281) can be reorganized to calculate the **required sample size n_{MC} to have an estimated yield $\hat{Y}$ with confidence level ζ [%] in a confidence interval** $Y \in [\hat{Y} \pm \Delta Y]$:

From the confidence level, we calculate the interval width in k_ζ of the standard distribution:

$$\zeta = cdf_N\left(\hat{Y} + k_\zeta \cdot \hat{\sigma}_{\hat{Y}}\right) - cdf_N\left(\hat{Y} - k_\zeta \cdot \hat{\sigma}_{\hat{Y}}\right) \to k_\zeta \tag{282}$$

Equating the interval width in ΔY with the interval width in k_ζ times, i.e., $\Delta Y = k_\zeta \cdot \hat{\sigma}_{\hat{Y}}$, and using (280) we obtain:

$$\boxed{n_{MC} = 1 + \frac{\hat{Y} \cdot \left(1 - \hat{Y}\right) \cdot (k_\zeta)^2}{\Delta Y^2}} \tag{283}$$

Table 3 shows the resulting sample sizes n_{MC} for some confidence levels ζ and confidence intervals/estimated yield values.

$\hat{Y} \pm \Delta Y$	ζ	90%	95%	99%	99.9%
	k_ζ	1.645	1.960	2.576	3.291
85% ± 10%		36	50	86	140
85% ± 5%		140	197	340	554
85% ± 1%		3,452	4,900	8,462	13,811
99.99% ± 0.01%				66,352	

Table 3: Sample size n_{MC} required for confidence interval $\hat{Y} \pm \Delta Y$ and confidence level ζ.

It requires more than 60,000 circuit/system simulations to verify that a circuit/system has a yield of 99.99% ± 0.01% with 99% confidence! This shows that verifying a robust design with a large yield value by statistical estimation with a Monte Carlo analysis requires a huge number of simulations!

Another approach to determining the required sample size that does not need the assumption of a normal distribution is hypothesis testing:

We require that the yield should be larger than a certain value $\hat{Y} \overset{!}{>} Y_{min}$ with a significance level α [%]. The null hypothesis H_0, $\hat{Y} < Y_{min}$, is rejected if all circuits are functioning, i.e., if $n_+ = n_{MC}$. Assuming that H_0 holds, the probability of falsely (i.e., $\hat{Y} < Y_{min}$) rejecting H_0 is

$$P(\text{"rejection"}) \overset{\left(\substack{\text{test}\\\text{definition}}\right)}{=} P\left(\text{"}n_+ = n_{MC}\text{"}\right) \tag{284}$$

$$\overset{\left(\substack{\text{binominal}\\\text{distribution}}\right)}{=} \hat{Y}^{n_{MC}} \overset{\text{(falsely)}}{<} Y_{min}^{n_{MC}} \overset{!}{<} \alpha \tag{285}$$

From that, we obtain

$$\boxed{n_{MC} > \frac{\log \alpha}{\log Y_{min}}} \tag{286}$$

Table 4 shows some numbers from (286). We obtain a number of $n_{MC} = 46,000$ to verify a yield of 99.99% with 1% significance level.

Y_{min}	$\alpha = 5\%$	$\alpha = 1\%$
95%	60	92
99%	300	460
99.9%	3,000	4,600
99.99%	30,000	46,000

Table 4: Sample size n_{MC} to verify a minimum yield Y_{min} with significance level α.

Importance sampling

Importance sampling aims at determining a specific sampling distribution pdf_{IS} to reduce the estimator's variance. The yield estimation can be reformulated as:

$$\begin{aligned} Y &= \int_{-\infty}^{\infty} \cdots \int \delta\left(\mathbf{x}_s\right) \cdot pdf\left(\mathbf{x}_s\right) \cdot d\mathbf{x}_s && (287) \\ &= \int_{-\infty}^{\infty} \cdots \int \delta\left(\mathbf{x}_s\right) \cdot \frac{pdf\left(\mathbf{x}_s\right)}{pdf_{IS}\left(\mathbf{x}_s\right)} \cdot pdf_{IS}\left(\mathbf{x}_s\right) \cdot d\mathbf{x}_s && (288) \\ &= \underset{pdf_{IS}}{E}\left\{\delta\left(\mathbf{x}_s\right) \cdot \frac{pdf\left(\mathbf{x}_s\right)}{pdf_{IS}\left(\mathbf{x}_s\right)}\right\} = E\left\{\delta(\mathbf{x}_s) \cdot w(\mathbf{x}_s)\right\} && (289) \end{aligned}$$

The sample uses a specific importance sampling distribution pdf_{IS}.

Please note that any integration problem can be solved using a Monte-Carlo analysis and by moving the integration bounds inside of the integral and introducing a sampling distribution:

$$\begin{aligned} \int_{\mathbf{x}\in IR} \cdots \int f_{int}(\mathbf{x}) d\mathbf{x} &= \int_{-\infty}^{\infty} \cdots \int \delta(\mathbf{x}) f_{int}(\mathbf{x}) d\mathbf{x} \quad \text{with } \delta(\mathbf{x}) = \begin{cases} 1, \mathbf{x} \in IR \\ 0, \text{else} \end{cases} \\ &= \int_{-\infty}^{\infty} \cdots \int \frac{\delta(\mathbf{x}) f_{int}(\mathbf{x})}{pdf_{IS}(\mathbf{x})} pdf_{IS}(\mathbf{x}) d\mathbf{x} \\ &= \underset{pdf_{IS}}{E}\left\{\frac{\delta(\mathbf{x}) f_{int}(\mathbf{x})}{pdf_{IS}(\mathbf{x})}\right\} && (290) \end{aligned}$$

9.4 Geometric yield analysis for linearized performance feature

We will present a deterministic yield analysis method based on a geometric concept. In the first step, we will use a linearized performance model. As described earlier, the statistical parameters have eventually been transformed into a normal distribution.

Analogously to the realistic worst-case analysis formulated for a linear performance model, we can name the geometric yield analysis with a linearized performance feature **realistic geometric yield analysis**.

Please note that we will partition the yield analysis problem by investigating individual performance features. We will denote the resulting yield values for individual performance features as **yield partition**s. Later on, we will collect the yield partitions to the overall yield, following a "divide-and-conquer" principle. No index i to indicate a performance feature φ_i will be given in the following.

The derivation will be given for a **lower performance bound** $\varphi \geq \varphi_L$, and for a linear performance model around a point $\mathbf{x}_{s,a}$, for which the lower performance bound is obtained, i.e., $\varphi(\mathbf{x}_{s,a}) = \varphi_L$:

$$\overline{\varphi} = \varphi_L + \mathbf{g}^T \cdot (\mathbf{x}_s - \mathbf{x}_{s,a}) \tag{291}$$

The acceptance region A_s for one performance bound is a half of $\mathbb{R}^{n_s}$, in the case of a linear performance model, the boundary is a hyperplane defined by $\overline{\varphi}(\mathbf{x}_s) = \varphi_L$. The yield partition, then, is the percentage of circuits in the half-space linearly bounded by the hyperplane.

Fig. 36 illustrates the situation for two statistical parameters.

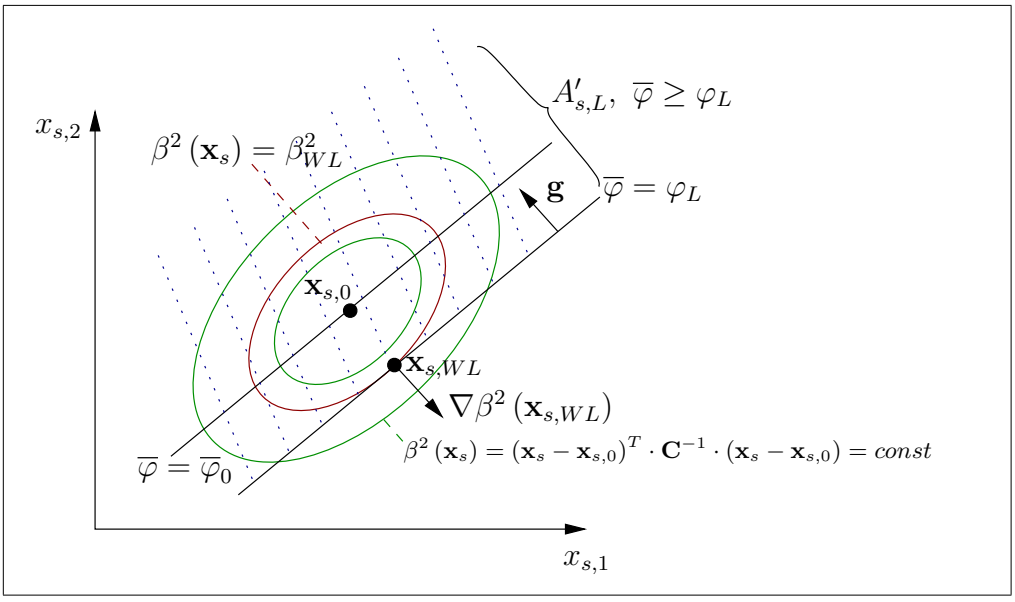


Figure 36. Yield partition of a linearized performance feature is the integral of the parameters' probability density function over the hatched area.

In realistic worst-case analysis, a distinct parameter hyperellipsoid is selected, and the tangential hyperplane of the performance in the parameter space to that parameter hyperellipsoid is searched for.

In realistic geometric yield analysis, we vice versa select a specific performance value, which refers to a particular performance hyperplane in the parameter space, and search for the tangential parameter hyperellipsoid.

In realistic worst-case analysis, the size of the parameter hyperellipsoid was selected, i.e., the worst-case distance β_{WL}, and we found out that the worst-case distance was also the multiple of the performance standard deviation.

In the following realistic geometric yield analysis, we will, vice versa, define the worst-case distance β_{WL} in the performance space and find out that it refers to the size of the tangential parameter hyperellipsoid. That means a reverse path to the same yield relation and yield approximation will be presented in the following, established while dealing with worst-case analysis.

Step 1:

We will start by using the fact that a linear function maps a normal distribution into a normal distribution (248), (249). Hence, we can formulate the mean value $\overline{\varphi}_0$ and the variance $\sigma^2_{\overline{\varphi}}$ of the linearized performance, and we can formulate the yield partition $\overline{Y}_L$ for a lower bound of a single performance feature in the performance space:

$$\overline{\varphi} \sim N\left(\underbrace{\varphi_L + \mathbf{g}^T \cdot (\mathbf{x}_{s,0} - \mathbf{x}_{s,WL})}_{\overline{\varphi}_0 = \overline{\varphi}(\mathbf{x}_{s,0})},\ \underbrace{\mathbf{g}^T \cdot \mathbf{C} \cdot \mathbf{g}}_{\sigma^2_{\overline{\varphi}}}\right) \tag{292}$$

$$\overline{Y}_L = \int_{\mathbf{x}_s \in A'_{s,L}} \cdots \int pdf_N(\mathbf{x}_s) \cdot d\mathbf{x}_s = \int_{\overline{\varphi} \geq \varphi_L} pdf_{\overline{\varphi}}(\overline{\varphi}) \cdot d\overline{\varphi} = \int_{\varphi_L}^{\infty} \frac{1}{\sqrt{2\pi} \cdot \sigma_{\overline{\varphi}}} \cdot e^{-\frac{1}{2}\left(\frac{\overline{\varphi} - \overline{\varphi}_0}{\sigma_{\overline{\varphi}}}\right)^2} \cdot d\overline{\varphi} \tag{293}$$

This formula says that the yield partition $\overline{Y}_L$ is obtained from integration over the (analytically calculatable) Gaussian probability density function $pdf_{\overline{\varphi}}$ of the linearized performance feature $\overline{\varphi}$ from φ_L to ∞. This is illustrated in Fig. 37.

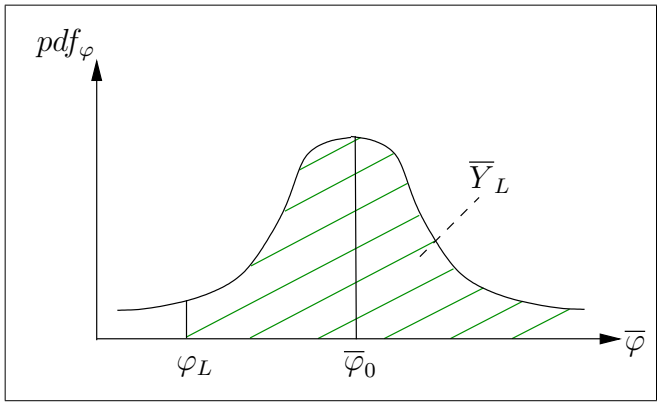


Figure 37. Yield partition of linearized performance feature $\overline{\varphi}$.

Step 2:

We now **define the worst-case distance β_{WL} as the difference between nominal performance and specification bound as multiple of standard deviation $\sigma_{\overline{\varphi}}$ of the linearized performance feature**:

$$\boxed{\overline{\varphi}_0 - \varphi_L \;=\; \begin{cases} +\beta_{WL} \cdot \sigma_{\overline{\varphi}}, & \overline{\varphi}_0 > \varphi_L \quad \text{"circuit functions"} \\ -\beta_{WL} \cdot \sigma_{\overline{\varphi}}, & \overline{\varphi}_0 < \varphi_L \quad \text{"circuit malfunctions"} \end{cases}} \tag{294}$$

Step 3:

The worst-case distance defined in this way is used for a variable substitution in (293):

$$\begin{aligned} t &= \frac{\overline{\varphi} - \overline{\varphi}_0}{\sigma_{\overline{\varphi}}}, \frac{d\overline{\varphi}}{dt} = \sigma_{\overline{\varphi}} \\ t_L &= \frac{\varphi_L - \overline{\varphi}_0}{\sigma_{\overline{\varphi}}} = \begin{cases} -\beta_{WL}, & \overline{\varphi}_0 > \varphi_L \\ +\beta_{WL}, & \overline{\varphi}_0 < \varphi_L \end{cases} \end{aligned}$$

$$\overline{Y}_L = \int_{\mp\beta_{WL}}^{\infty} \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{t^2}{2}} \cdot dt = -\int_{\pm\beta_{WL}}^{-\infty} \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{(-t')^2}{2}} \cdot dt' \quad \left(t' = -t, \frac{dt}{dt'} = -1\right) \tag{295}$$

This means that the **yield partition can be calculated as a function of the worst-case distance** β_{WL} and the standard normal distribution:

$$\boxed{\begin{aligned} &\overline{Y}_L = \textstyle\int_{-\infty}^{\pm\beta_{WL}} \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{t'^2}{2}} \cdot dt' \\ &\overline{\varphi}_0 > \varphi_L : +\beta_{WL} \\ &\overline{\varphi}_0 < \varphi_L : -\beta_{WL} \end{aligned}} \tag{296}$$

(296) says that the **yield partition** of a single performance specification that depends on a **multivariate statistical distribution of parameters** can be determined by a univariate standard normal distribution without any Monte-Carlo analysis with the **accuracy of the underlying statistical table of the standard normal distribution** with β_{WL} computed by (294), and using $\overline{\varphi}_0$ and $\sigma_{\overline{\varphi}}^2$ from (292).

(296) also says that the **yield partition** is **determined by the worst-case distance** in the sense of a **β_{WL}-sigma design**.

Step 4:

We will now look at the worst-case distance in the statistical parameter space. Let us insert a specific point of the level contour $\overline{\varphi} = \varphi_L$ into the linear performance model. This specific point is $\mathbf{x}_{s,WL}$ in Fig. 36. We obtain

$$\overline{\varphi}\left(\mathbf{x}_{s,WL}\right) = \varphi_L \overset{(291)}{\Longleftrightarrow} \mathbf{g}^T \cdot \left(\mathbf{x}_{s,WL} - \mathbf{x}_a\right) = 0 \tag{297}$$

As the right term in (297) is zero, it can be added to the linear model to obtain:

$$\begin{aligned} \overline{\varphi}\left(\mathbf{x}_s\right) &= \varphi_L + \mathbf{g}^T \cdot \left(\mathbf{x}_s - \mathbf{x}_{s,a}\right) - \mathbf{g}^T \cdot \left(\mathbf{x}_{s,WL} - \mathbf{x}_{s,a}\right) \\ &= \varphi_L + \mathbf{g}^T \cdot \left(\mathbf{x}_s - \mathbf{x}_{s,WL}\right) \end{aligned} \tag{298}$$

That means that $\mathbf{x}_{s,a}$ in (291) can be replaced with any point of level set $\overline{\varphi} = \varphi_L$.

From visual inspection of Fig. 36, we can see that the gradient of the tolerance ellipsoid and the gradient of the linearized performance in $\mathbf{x}_{s,WL}$ have the same or opposite direction, depending on whether the nominal parameter set $\mathbf{x}_{s,0}$ is inside or outside the acceptance region[1]:

$$\nabla\beta^2\left(\mathbf{x}_{s,WL}\right) = 2 \cdot \mathbf{C}^{-1} \cdot \left(\mathbf{x}_{s,WL} - \mathbf{x}_{s,0}\right) = \begin{cases} -\lambda_{WL} \cdot \mathbf{g}, & \overline{\varphi}_0 > \varphi_L \\ +\lambda_{WL} \cdot \mathbf{g}, & \overline{\varphi}_0 < \varphi_L \end{cases}, \quad \lambda_{WL} > 0 \tag{299}$$

From this, we obtain

$$\left(\mathbf{x}_{s,WL} - \mathbf{x}_{s,0}\right) = \mp\frac{\lambda_{WL}}{2} \cdot \mathbf{C} \cdot \mathbf{g} \tag{300}$$

$$\Longrightarrow \mathbf{g}^T \cdot \left(\mathbf{x}_{s,WL} - \mathbf{x}_{s,0}\right) = \mp\frac{\lambda_{WL}}{2} \cdot \sigma_{\overline{\varphi}}^2 \overset{(298)}{=} \varphi_L - \overline{\varphi}_0 \overset{(294)}{=} \mp\beta_{WL} \cdot \sigma_{\overline{\varphi}} \tag{301}$$

which gives the following relation

$$\frac{\lambda_{WL}}{2} = \frac{\beta_{WL}}{\sigma_{\overline{\varphi}}} \tag{302}$$

Substituting (302) in (300) leads to

$$\left(\mathbf{x}_{s,WL} - \mathbf{x}_{s,0}\right) = \mp\frac{\beta_{WL}}{\sigma_{\overline{\varphi}}} \cdot \mathbf{C} \cdot \mathbf{g} \tag{303}$$

$$\left(\mathbf{x}_{s,WL} - \mathbf{x}_{s,0}\right)^T \cdot \mathbf{C}^{-1} \cdot \left(\mathbf{x}_{s,WL} - \mathbf{x}_{s,0}\right) = \beta_{WL}^2 \cdot \underbrace{\frac{1}{\sigma_{\overline{\varphi}^2}^2} \cdot \mathbf{g}^T \cdot \mathbf{C} \cdot \mathbf{g}}_{1} \tag{304}$$

and finally to:

$$\boxed{\left(\mathbf{x}_{s,WL} - \mathbf{x}_{s,0}\right)^T \cdot \mathbf{C}^{-1} \cdot \left(\mathbf{x}_{s,WL} - \mathbf{x}_{s,0}\right) = \beta_{WL}^2} \tag{305}$$

This means the **worst-case distance** β_{WL}, which has been defined in the performance space as a performance safety margin in multiples of the standard deviation of the linearized performance, at the same time **defines the hyperellipsoid that touches the performance level hyperplane** $\overline{\varphi} = \varphi_L$.

[1] This corresponds to the KKT condition of a related optimization problem.

9.4.1 Upper performance bound $\overline{\varphi} \leq \varphi_U$

In case of an upper performance bound, (294) becomes

$$\boxed{\varphi_U - \overline{\varphi}_0 \;=\; \begin{cases} +\beta_{WU} \cdot \sigma_{\overline{\varphi}}, & \overline{\varphi}_0 < \varphi_U \quad \text{"circuit functions"} \\ -\beta_{WU} \cdot \sigma_{\overline{\varphi}}, & \overline{\varphi}_0 > \varphi_U \quad \text{"circuit malfunctions"} \end{cases}} \tag{306}$$

(296) becomes

$$\boxed{\begin{array}{l} \overline{Y}_U = \int_{-\infty}^{\pm\beta_{WU}} \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{t'^2}{2}} \cdot dt' \\ \overline{\varphi}_0 < \varphi_U : +\beta_{WU} \\ \overline{\varphi}_0 > \varphi_U : -\beta_{WU} \end{array}} \tag{307}$$

9.5 General geometric yield analysis

In general, the geometric yield analysis deals with a nonlinear performance feature. The level contour that divides the parameter space into "good" and "bad" circuits/systems is nonlinear. Fig. 38 illustrates the situation for a lower bound. The nonlinear level contour of parameter sets with $\varphi = \varphi_L$ is curved.

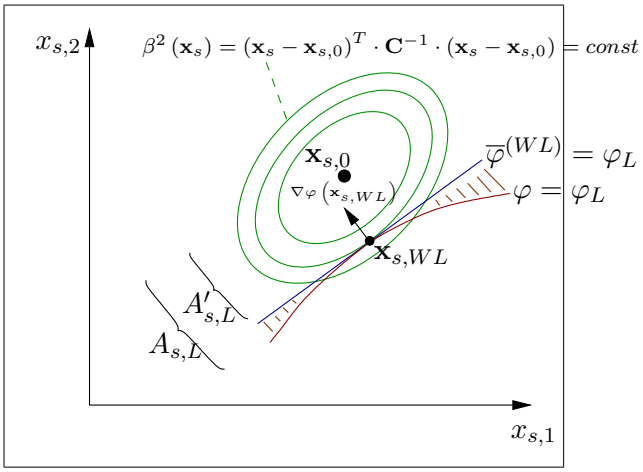


Figure 38. Geometric yield analysis with general nonlinear performance feature, which has a lower bound.

Suppose you imagine the probability density function, i.e., the Gaussian bell curve, growing from the figure towards you. In that case, there is a characteristic point, $\mathbf{x}_{s,WL}$, among all points on the lower right half of the space, i.e., the side with "bad" circuits. It is characterized in that it has the highest probability (density) among all "bad" points.

This is a general deterministic approach to the statistical problem of discriminant analysis: The worst-case parameter set is the point with the highest probability (density) to fail the specification. The linear model of the performance at the worst-case parameter set gives a linear approximation of the corresponding discriminant analysis problem.

In geometric yield analysis, we have to distinguish four cases arising from having a lower or upper bound that is either fulfilled or violated in the nominal point. We can summarize the four cases into two cases, described as **find the point of highest probability on the other side of the separating contour curve from the nominal point.** This leads to the following two optimization problems.

For a lower bound that is nominally fulfilled or for an upper bound that is nominally violated, the geometric yield analysis is the solution to this optimization problem:

$$\varphi(\mathbf{x}_{s,0}) > \varphi_{L/U} \; : \; \max_{\mathbf{x}_s} \; pdf_N(\mathbf{x}_s) \;\; \text{s.t.} \;\; \varphi(\mathbf{x}_s) \leq \varphi_{L/U} \tag{308}$$

For a lower bound that is nominally violated or for an upper bound that is nominally fulfilled, the geometric yield analysis is the solution to this optimization problem:

$$\varphi(\mathbf{x}_{s,0}) < \varphi_{L/U} \; : \; \max_{\mathbf{x}_s} \; pdf_N(\mathbf{x}_s) \;\; \text{s.t.} \;\; \varphi(\mathbf{x}_s) \geq \varphi_{L/U} \tag{309}$$

Considering the quadratic level contours of the probability density function with decreasing density values for increasing size of the level hyperellipsoids, we can reformulate the geometric yield analysis as the point with the smallest distance to the level contour. The distance is weighted according to the quadratic form of the tolerance hyperellipsoids ("Mahalanobis norm").

For a lower bound that is nominally fulfilled or for an upper bound that is nominally violated, the geometric yield analysis is the solution of this minimization problem:

$$\boxed{\varphi\left(\mathbf{x}_{s,0}\right) > \varphi_{L/U} \;:\; \min_{\mathbf{x}_s} \; \beta^2\left(\mathbf{x}_s\right) \;\text{ s.t. }\; \varphi\left(\mathbf{x}_s\right) \leq \varphi_{L/U}} \tag{310}$$

For a lower bound that is nominally violated or for an upper bound that is nominally fulfilled, the geometric yield analysis is the solution of this minimization problem:

$$\boxed{\varphi\left(\mathbf{x}_{s,0}\right) < \varphi_{L/U} \;:\; \min_{\mathbf{x}_s} \; \beta^2\left(\mathbf{x}_s\right) \;\text{ s.t. }\; \varphi\left(\mathbf{x}_s\right) \geq \varphi_{L/U}} \tag{311}$$

Geometric yield analysis according to (310) and (311) is a specific form of a nonlinear programming problem with a quadratic objective function and one nonlinear inequality constraint.

It can be visualized as "blowing into a balloon of the shape of an ellipsoid until it touches the level contour". It can also be imagined as "pulling in (pulling is skewed according to the ellipsoids) a fishing line (the other side is the water) until it hits the bank".

Please compare the problem formulations of geometric yield analysis and general worst-case analysis. They have the same structure with the difference that objective and constraint have been exchanged. Fig. 39 illustrates it.

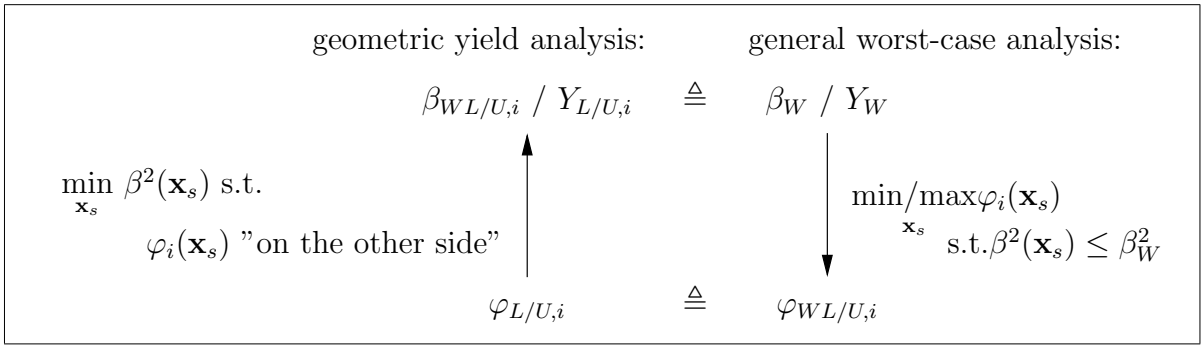


Figure 39. Geometric yield analysis and worst-case analysis in comparison.

9.6 Accuracy of geometric yield approximation

The yield value obtained by a general geometric yield analysis is based on the worst-case distance to the general worst-case parameter vector obtained by solving the nonlinear minimization problems (310) and (311). The obtained yield is exact for the linear hyperplane in the worst-case parameter vector. The error is systematic and depends on the difference from the linear hyperplane to the exact separating hypercurve. The (red) hatched area in Fig. 38 is where the linear model misses adding the probability function's volume. In this case, the yield is underestimated. The yield would have been overestimated if the curvature had been to the other side.

Here are a few considerations why this error is negligible.

First, the error in the yield estimation for a given infinitesimal dx is larger, the larger the corresponding value of the probability density function is. Looking at the definition of the worst-case parameter set, the separating hyperplane is exact where the pdf value is maximum. The more the deviation of the linear hyperplane to the exact separating curve, the smaller the pdf value and the smaller the error is. The geometric yield estimation, therefore, inherently fits the accuracy of the separation approximation to the required accuracy of the yield estimation.

Second, let us consider the duality principle in minimum-norm problems. It says that the minimum distance between a point and a convex set equals the maximum distance between the point and any separating hyperplane. Fig. 40 illustrates this. The convex set indicated by the hatches represents the failure/non-acceptance region. The worst-case point is the point with the smallest distance to the acceptance region. And the one with the maximum distance to any hyperplane that separates the nominal point from the convex set. As we determine the largest possible worst-case distance in this case, we also obtain the largest possible yield value for any separating hyperplane. At the same time, we underestimate the true yield. Hence, we get the greatest lower bound for all possible separating hyperplanes.

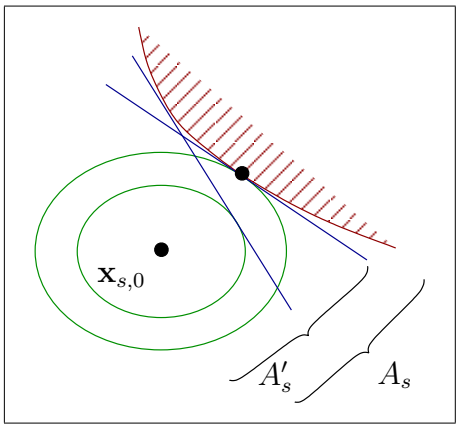


Figure 40. Yield estimation $\overline{Y}\,(A'_s)$ is the greatest lower bound of $Y\,(A_s)$ concerning any tangent hyperplane of the acceptance region.

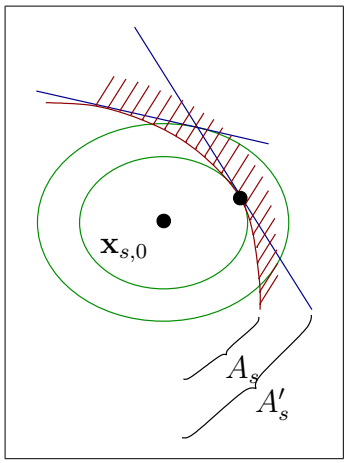


Figure 41. Yield estimation $\overline{Y}\,(A'_s)$ is the least upper bound of $Y\,(A_s)$ concerning any tangent hyperplane of the acceptance region.

Another situation is that the curvature of the separating curve is towards the nominal point. Here, we obtain the smallest possible worst-case distance to all tangential hyperplanes of the acceptance region while overestimating the true yield value. Hence, we get the least upper bound.

Overall, we obtain the best solution of all possible tangential hyperplanes.

Third, and most importantly, we should know that yield analysis in technical problems aims for a robust system with worst-case distances beyond 3, 6, 9, or more. The yield value in these cases is beyond 99.9%. Neither does it change much anymore, so the area approximation error is negligible. Nor is the yield value of any relevance anymore, as its sensitivity to changes in the design is close to zero. It is the worst-case distance that gives a sensitive robustness measure with regard to design changes and design centering.

9.7 Overall yield from geometric yield analysis

The overall yield is obtained from collecting the worst-case points, the linearizations in those points, and the corresponding worst-case distances for all performance specifications. Fig. 42 illustrates this for two parameters and four specification bounds, e.g., a lower and an upper bound for each performance feature.

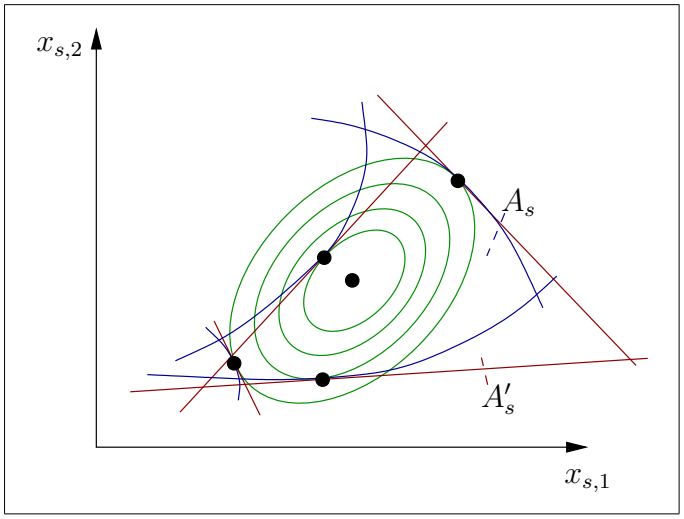


Figure 42. Combine the yield analysis results of four specification bounds.

With the four linear performance models in the four respective worst-case points, an approximation A'_s of the acceptance region in the parameter space is available. Using A'_s, a Monte-Carlo analysis can be conducted without any expensive circuit/system simulation, just on the linear equations. These can be evaluated at very low computational cost so that thousands or millions of function evaluations can be conducted.

An approximation of the overall yield can be made solely based on the worst-case distances without considering worst-case points and linear performance models in the worst-case points:

The smallest worst-case distance gives an upper bound of the overall yield:

$$\overline{Y} \leq \min_i \overline{Y}_{L/U,i} \tag{312}$$

The overall yield cannot be larger, as every further specification leads to cutting away another piece of the "production cake" represented by the normal probability density function.

On the other hand, an upper bound of the overall yield is obtained by summing up all yield losses from individual specification bounds and subtracting that from 100%:

$$\overline{Y} \geq 1 - \sum_i \left(1 - \overline{Y}_{L/U,i}\right) \tag{313}$$

It is an upper bound as we count overlapping regions of cuts of individual specification bounds several times, adding up too much yield loss.

9.8 Consideration of range parameters

Up to here, we have not considered range parameters in the yield estimation. Range parameters have no distribution. They are part of the specification in that the performance of a circuit/system has to be guaranteed for the complete range of those parameters.

Fig. 43 illustrates three sample points of statistical parameters, $\mathbf{x}_s^{(1)}$, $\mathbf{x}_s^{(2)}$, $\mathbf{x}_s^{(3)}$ and the interval of values for one performance feature φ that each of these points covers over the tolerance range T_r. It is clear that $\mathbf{x}_s^{(3)}$ fails the specification and $\mathbf{x}_s^{(1)}$ fulfills the specificaiton. But what about $\mathbf{x}_s^{(2)}$? It fulfills the specification for some sets of range parameters and fails it for others. The answer is clear if you take the example of a mobile phone that works in summer (high temperature) and does not work in winter (low temperature), and when bought, it was promised to work all year round.

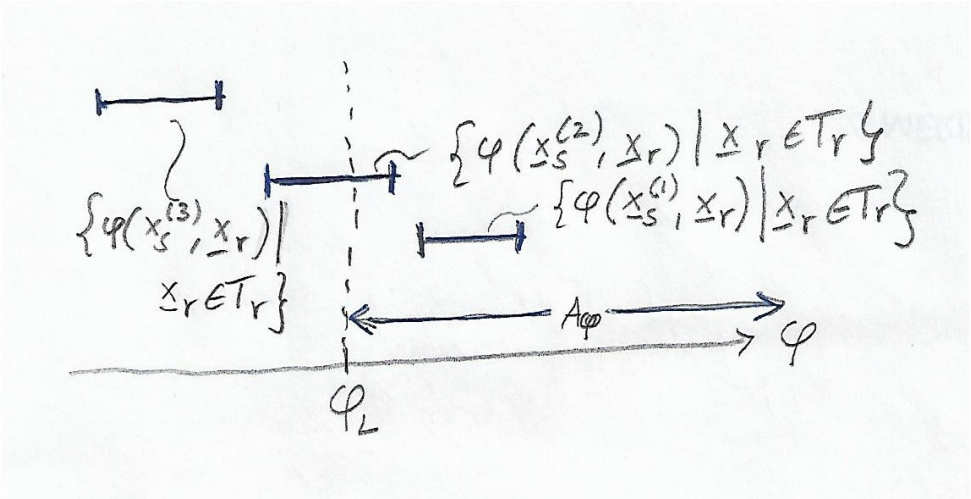


Figure 43. Three statistical sample points with performance interval due to range parameter tolerances.

An extended definition of the acceptance region A_s is the **set of statistical parameter sets that satisfy the specification for all range parameters in their given tolerance intervals**. The region of failure $\overline{A}_s$ is obtained when a range parameter vector exists at a statistical parameter sample element that fails the specification:

$$A_{s,L,i} = \left\{ \mathbf{x}_s \Big|_{\mathbf{x}_r \in T_r}^{\forall} \varphi_i(\mathbf{x}_s, \mathbf{x}_r) \geq \varphi_{L,i} \right\} \tag{314}$$

$$A_{s,U,i} = \left\{ \mathbf{x}_s \Big|_{\mathbf{x}_r \in T_r}^{\forall} \varphi_i(\mathbf{x}_s, \mathbf{x}_r) \leq \varphi_{U,i} \right\} \tag{315}$$

$$\overline{A}_{s,L,i} = \left\{ \mathbf{x}_s \Big|_{\mathbf{x}_r \in T_r}^{\exists} \varphi_i(\mathbf{x}_s, \mathbf{x}_r) < \varphi_{L,i} \right\} \tag{316}$$

$$\overline{A}_{s,U,i} = \left\{ \mathbf{x}_s \Big|_{\mathbf{x}_r \in T_r}^{\exists} \varphi_i(\mathbf{x}_s, \mathbf{x}_r) > \varphi_{U,i} \right\} \tag{317}$$

with:

$$A_{s,L,i} \cap \overline{A}_{s,L,i} = \phi \quad \text{and} \quad A_{s,L,i} \cup \overline{A}_{s,L,i} = \mathbb{R}^{n_s} \tag{318}$$

The problem formulation of geometric yield analysis now starts with this extended definition of acceptance regions:

$$\varphi(\mathbf{x}_{s,0}) \in A_{s,L,i} : \max_{\mathbf{x}_s} pdf_N(\mathbf{x}_s) \text{ s.t. } \mathbf{x}_s \in \overline{A}_{s,L,i} \tag{319}$$

$$\varphi(\mathbf{x}_{s,0}) \in \overline{A}_{s,L,i} : \max_{\mathbf{x}_s} pdf_N(\mathbf{x}_s) \text{ s.t. } \mathbf{x}_s \in A_{s,L,i} \tag{320}$$

From (314) to (317), we obtain the following definitions of statistical parameter set being inside or outside of acceptance:

$$\mathbf{x}_s \in A_{s,L,i} \iff \min_{\mathbf{x}_r \in T_r} \varphi_i(\mathbf{x}_s, \mathbf{x}_r) \geq \varphi_{L,i} \text{ "smallest value still inside"} \tag{321}$$

$$\mathbf{x}_s \in A_{s,U,i} \iff \max_{\mathbf{x}_r \in T_r} \varphi_i(\mathbf{x}_s, \mathbf{x}_r) \leq \varphi_{U,i} \text{ "largest value still inside"} \tag{322}$$

$$\mathbf{x}_s \in \overline{A}_{s,L,i} \iff \min_{\mathbf{x}_r \in T_r} \varphi_i(\mathbf{x}_s, \mathbf{x}_r) < \varphi_{L,i} \text{ "smallest value already outside"} \tag{323}$$

$$\mathbf{x}_s \in \overline{A}_{s,U,i} \iff \max_{\mathbf{x}_r \in T_r} \varphi_i(\mathbf{x}_s, \mathbf{x}_r) > \varphi_{U,i} \text{ "largest value already outside"} \tag{324}$$

This means we must look for the extreme performance values within the tolerance intervals of range parameters for each statistical parameter set. If this value is still acceptable, the corresponding statistical parameter set is "good" because it satisfies the specification for the whole tolerance range of range parameters. If the extreme value is not acceptable, the corresponding statistical parameter set is "bad" because it does not satisfy the specification for the whole tolerance range of range parameters.

For the geometric yield analysis, we insert this into (319), (320) to obtain:

$$\mathbf{x}_{s,0} \in A_{s,L} : \min_{\mathbf{x}_s} \beta^2(\mathbf{x}_s) \text{ s.t. } \min_{\mathbf{x}_r \in T_r} \varphi(\mathbf{x}_s, \mathbf{x}_r) \leq \varphi_L \text{ "nominal inside"} \tag{325}$$

$$\mathbf{x}_{s,0} \in \overline{A}_{s,L} : \min_{\mathbf{x}_s} \beta^2(\mathbf{x}_s) \text{ s.t. } \min_{\mathbf{x}_r \in T_r} \varphi(\mathbf{x}_s, \mathbf{x}_r) \geq \varphi_L \text{ "nominal outside"} \tag{326}$$

$$\mathbf{x}_{s,0} \in A_{s,U} : \min_{\mathbf{x}_s} \beta^2(\mathbf{x}_s) \text{ s.t. } \max_{\mathbf{x}_r \in T_r} \varphi(\mathbf{x}_s, \mathbf{x}_r) \geq \varphi_U \text{ "nominal inside"} \tag{327}$$

$$\mathbf{x}_{s,0} \in \overline{A}_{s,U} : \min_{\mathbf{x}_s} \beta^2(\mathbf{x}_s) \text{ s.t. } \max_{\mathbf{x}_r \in T_r} \varphi(\mathbf{x}_s, \mathbf{x}_r) \leq \varphi_U \text{ "nominal outside"} \tag{328}$$

Geometric yield analysis is now a constrained optimization problem with another constrained optimization problem, which is a worst-case analysis problem, in fact, a constraint.

In practice, worst-case range parameters often do not change over the relevant region of statistical parameters (and deterministic design parameters) so that they can be determined once at the beginning of the optimization process, then kept constant, and verified by another analysis at the end. Only if they change the optimization process has to be restarted.

9.9 Another interpretation of the worst-case distance and preparing its gradient

As done for the general worst-case analysis, a formula for the worst-case parameter set based on its linearization after solving the general yield analysis problem will be derived in the following. This formula will be used for a different formulation of the worst-case distance, leading to another interpretation of the worst-distance. It will also be used to get a gradient of the worst-case distance with regard to deterministic design parameters in the next chapter.

The Lagrange function and first-order optimality conditions of problem (310) are:

$$\mathcal{L}(\mathbf{x}_s, \lambda) = \beta^2(\mathbf{x}_s) - \lambda \cdot (\varphi_L - \varphi(\mathbf{x}_s)) \tag{329}$$

$$\nabla\mathcal{L}(\mathbf{x}_s) = 0: \; 2 \cdot \mathbf{C}^{-1} \cdot (\mathbf{x}_{s,WL} - \mathbf{x}_{s,0}) + \lambda_{WL} \cdot \nabla\varphi(\mathbf{x}_{s,WL}) = \mathbf{0} \tag{330}$$

$$\lambda_{WL} \cdot (\varphi_L - \varphi(\mathbf{x}_{s,WL})) = 0 \tag{331}$$

$$\beta_{WL}^2 = (\mathbf{x}_{s,WL} - \mathbf{x}_{s,0})^T \cdot \mathbf{C}^{-1} \cdot (\mathbf{x}_{s,WL} - \mathbf{x}_{s,0}) \tag{332}$$

If we assume that $\lambda_{WL} = 0$, the constraint is inactive or just active and $\varphi_L \geq \varphi(\mathbf{x}_{s,WL})$ according to (310). From (330) we would then obtain $\mathbf{x}_{s,WL} = \mathbf{x}_{s,0}$. But from (310), the nominal point was inside the acceptance region and $\varphi(\mathbf{x}_{s,WL}) = \varphi(\mathbf{x}_{s,0}) > \varphi_L$, which contradicts the assumption. Therefore, it holds:

$$\lambda_{WL} > 0 \tag{333}$$

$$\varphi(\mathbf{x}_{s,WL}) = \varphi_L \tag{334}$$

From (330), we get:

$$\mathbf{x}_{s,WL} - \mathbf{x}_{s,0} = -\frac{\lambda_{WL}}{2} \cdot \mathbf{C} \cdot \nabla\varphi(\mathbf{x}_{s,WL}) \tag{335}$$

Substituting (335) in (332) leads to:

$$\left(\frac{\lambda_{WL}}{2}\right)^2 \cdot \nabla\varphi(\mathbf{x}_{s,WL})^T \cdot \mathbf{C} \cdot \nabla\varphi(\mathbf{x}_{s,WL}) = \beta_{WL}^2 \tag{336}$$

$$\frac{\lambda_{WL}}{2} = \frac{\beta_{WL}}{\sqrt{\nabla\varphi(\mathbf{x}_{s,WL})^T \cdot \mathbf{C} \cdot \nabla\varphi(\mathbf{x}_{s,WL})}} \tag{337}$$

Substituting (337) in (335) finally results in:

$$\mathbf{x}_{s,WL} - \mathbf{x}_{s,0} = \frac{-\beta_{WL}}{\sqrt{\nabla\varphi(\mathbf{x}_{s,WL})^T \cdot \mathbf{C} \cdot \nabla\varphi(\mathbf{x}_{s,WL})}} \cdot \mathbf{C} \cdot \nabla\varphi(\mathbf{x}_{s,WL}) \tag{338}$$

(338) corresponds to (104) from worst-case analysis.

The second-order optimality condition of problem (310) is:

$$\nabla^2\mathcal{L}(\mathbf{x}_{s,WL}) = 2 \cdot \mathbf{C}^{-1} + \lambda_{WL} \cdot \nabla^2\varphi(\mathbf{x}_{s,WL}) \tag{339}$$

It describes that the curvature of β^2 has to be stronger than that of φ in the solution.

Similarly, we obtain for the problem (311)

$$\mathcal{L}(\mathbf{x}_s, \lambda) = \beta^2(\mathbf{x}_s) - \lambda \cdot (\varphi(\mathbf{x}_s) - \varphi_L) \tag{340}$$

Then, (338) becomes

$$\mathbf{x}_{s,WL} - \mathbf{x}_{s,0} = \frac{+\beta_{WL}}{\sqrt{\nabla\varphi(\mathbf{x}_{s,WL})^T \cdot \mathbf{C} \cdot \nabla\varphi(\mathbf{x}_{s,WL})}} \cdot \mathbf{C} \cdot \nabla\varphi(\mathbf{x}_{s,WL}) \tag{341}$$

Worst-case distance combines performance safety margin and performance sensitivity

Let us insert the nominal point of statistical parameters $\mathbf{x}_{s,0}$ into the performance linearization

$$\overline{\varphi}^{(WL/U)}(\mathbf{x}_s) = \varphi_{L/U} + \nabla\varphi\left(\mathbf{x}_{s,WL/U}\right)^T \cdot \left(\mathbf{x}_s - \mathbf{x}_{s,WL/U}\right) \tag{342}$$

at the worst-case parameter set $\mathbf{x}_{s,WL/U}$:

$$\overline{\varphi}^{(WL/U)}(\mathbf{x}_{s,0}) - \varphi_{L/U} = \nabla\varphi\left(\mathbf{x}_{s,WL/U}\right)^T \cdot \left(\mathbf{x}_{s,0} - \mathbf{x}_{s,WL/U}\right) \tag{343}$$

Substituting (338) and (341) in (343), we obtain the following formula for the worst-case distance for a nominally fulfilled lower bound or a nominally violated upper bound:

$$\boxed{\beta_{WL/U} = \frac{\overbrace{\overline{\varphi}^{(WL/U)}(\mathbf{x}_{s,0}) - \varphi_{L/U}}^{\text{"performance safety margin"}}}{\sqrt{\nabla\varphi\left(\mathbf{x}_{s,WL/U}\right)^T \cdot \mathbf{C} \cdot \nabla\varphi\left(\mathbf{x}_{s,WL/U}\right)}} = \frac{\nabla\varphi\left(\mathbf{x}_{s,WL/U}\right)^T \cdot \left(\mathbf{x}_{s,0} - \mathbf{x}_{s,WL/U}\right)}{\underbrace{\sigma_{\overline{\varphi}^{(WL/U)}}}_{\text{"performance variability"}}}} \tag{344}$$

Analogously for a nominally violated lower bound or a nominally fulfilled upper bound:

$$\boxed{\beta_{WL/U} = \frac{\overbrace{\varphi_{L/U} - \overline{\varphi}^{(WL/U)}(\mathbf{x}_{s,0})}^{\text{"performance safety margin"}}}{\sqrt{\nabla\varphi\left(\mathbf{x}_{s,WL/U}\right)^T \cdot \mathbf{C} \cdot \nabla\varphi\left(\mathbf{x}_{s,WL/U}\right)}} = \frac{\nabla\varphi\left(\mathbf{x}_{s,WL/U}\right)^T \cdot \left(\mathbf{x}_{s,WL/U} - \mathbf{x}_{s,0}\right)}{\underbrace{\sigma_{\overline{\varphi}^{(WL/U)}}}_{\text{"performance variability"}}}} \tag{345}$$

(344) and (344) show that the worst-case distance includes two aspects of improving the robustness. Improving the robustness, i.e., maximizing the yield, is obtained by maximizing the worst-case distances of a circuit/system. This includes maximizing the performance safety margin, which is visible in the numerator of the worst-case distance. This part refers to the difference in the performance values at the nominal and worst-case points. On the other hand, the robustness and yield are maximized by minimizing the sensitivity of performance with regard to statistical variation. This part is included in the denominator of the worst-case distance.

Fig. 44 illustrates this for a single performance feature. It shows the performance's initial probability density function with parts cut away left of the lower bound φ_L as yield loss. The yield can be increased by moving the nominal performance value away from the bound, thus increasing the performance safety margin. The green pdf illustrates this move to the right. The yield can also be increased by reducing the sensitivity with regard to statistical parameters. This reduces the performance variability, as seen in the worst-case distance's denominator and the figure's blue pdf. Both aspects are not independent of each other in a circuit and are considered in the worst-case distance objective.

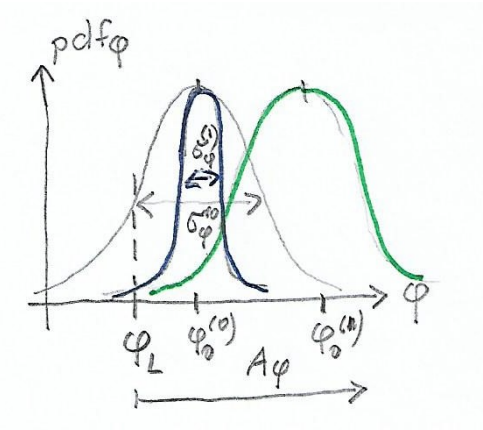


Figure 44. Moving the nominal point and reducing the standard deviation of a performance increase the yield partition.

10 Yield optimization/design centering/nominal design

10.1 Optimization objectives

While nominal design aims at optimizing the values of the performance features $\varphi_i, i = 1, \ldots, n_\varphi$, yield optimization and design centering (we use the two terms as synonyms) aims at optimizing either the yield Y, or the worst-case distances:

$$\beta_{WL/U,i} = \frac{|\overline{\varphi}^{(WL/U,i)}(\mathbf{x}_{s,0}) - \varphi_{L/U,i}|}{\sqrt{\nabla\varphi_i\left(\mathbf{x}_{s,WL/U,i}\right)^T \cdot \mathbf{C} \cdot \nabla\varphi_i\left(\mathbf{x}_{s,WL/U,i}\right)}} \tag{346}$$

$$= \frac{|\nabla\varphi_i\left(\mathbf{x}_{s,WL/U,i}\right)^T \cdot \left(\mathbf{x}_{s,0} - \mathbf{x}_{s,WL/U,i}\right) + \nabla\varphi_i\left(\mathbf{x}_{d,0}\right)^T \cdot \left(\mathbf{x}_d - \mathbf{x}_{d,0}\right)|}{\sigma_{\overline{\varphi}^{(WL/U,i)}}} \tag{347}$$

$$i = 1, \ldots, n_{PSF} \tag{348}$$

Fig. 45 illustrates the task of maximizing the yield. It consists in maximizing the volume of the probability density function that is not cut away by the performance specification. In Fig. 45, this is achieved by moving the mean value of the statistical parameters.

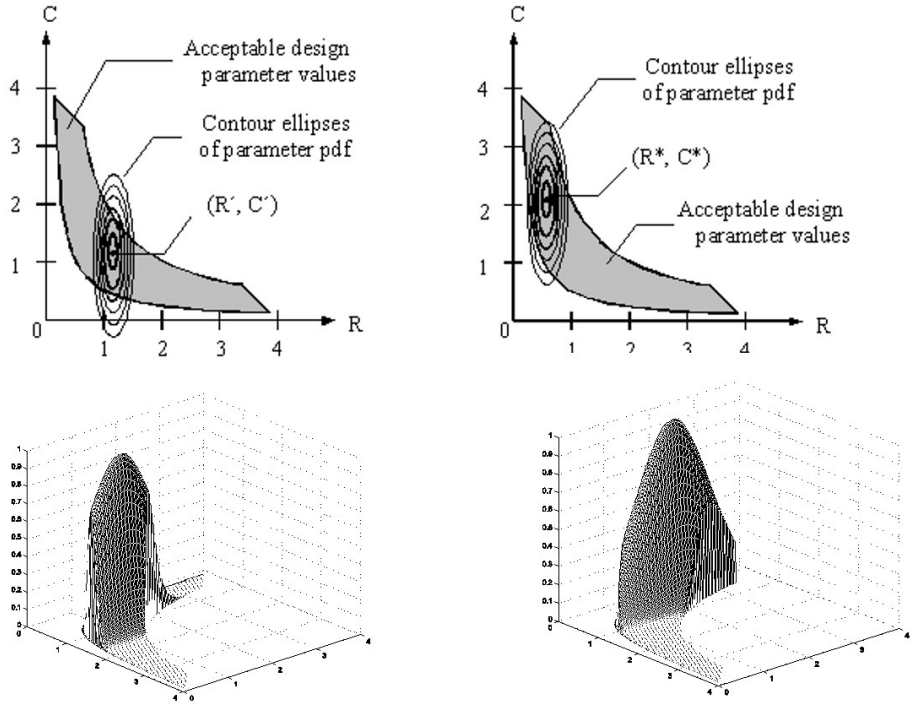


Figure 45. Maximizing the yield by moving the probability density function within the specification bounds.

Please note that we have extended the linear performance model in (347) to include a term for the deterministic parameters $\mathbf{x}_d$. The deterministic parameters are the design parameters, not the statistical ones. Designers are usually not tuning manufacturing process parameters, like threshold voltage or oxide thickness. This leads to another picture

in the statistical parameter space than the one in Fig. 45. As the mean values of statistical parameters are not the design parameters, the pdf volume maximizes by moving the bounds. Fig. 46 illustrates this. In addition to the two statistical parameters, a deterministic design parameter pointing to the reader represents the third dimension in the overall parameter space of this example. There is a gradient of the worst-case distance with regard to this deterministic parameter that guides how to move the specification bound in the statistical parameter space away from the center of the probability density function to maximize the remaining volume of the probability density function after having been truncated by the specification.

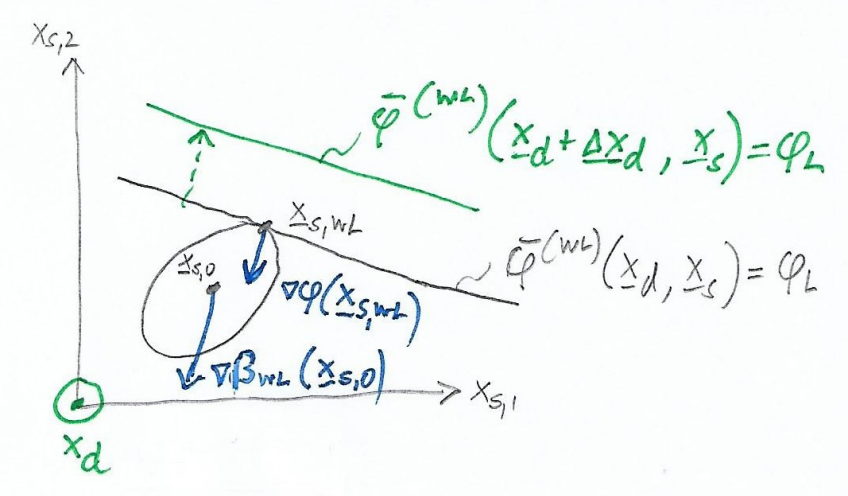


Figure 46. Maximizing the yield by moving the specification bound away from the center of the probability density function.

An illustration of yield optimization of five performance features of an operational amplifier is given in Fig. 47. The left axis shows the values of the worst-case distances, and on the right side, the corresponding yield partition values according to (296) and (307).

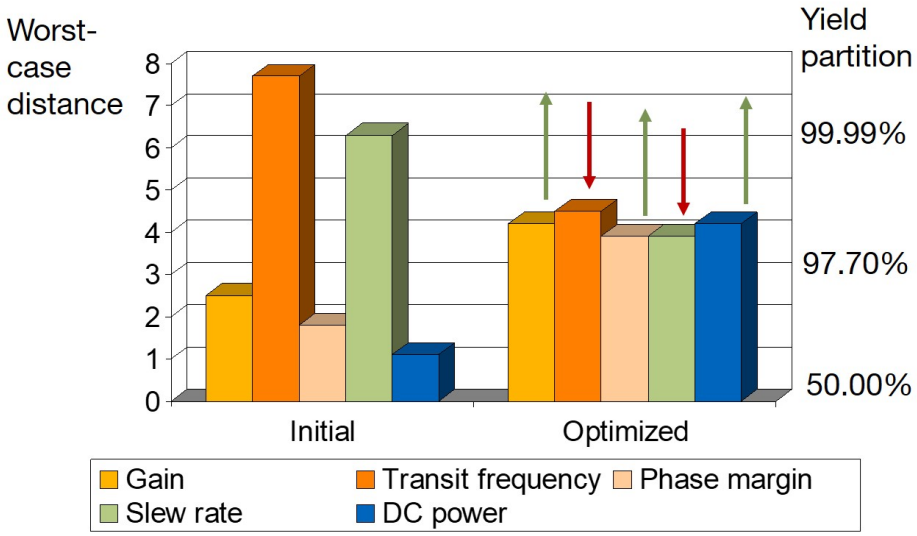


Figure 47. Maximizing five performance specification features of an operational amplifier.

The initial situation on the left shows strongly differing worst-case distances. The smallest worst-case distance is that of the power consumption with a value around 1. The initial

design is, therefore, not more than a one-sigma design. On the right, the optimized design is shown. The minimum worst-case distances are now close to 4. This means a four-sigma design has been achieved for the corresponding circuit structure and process technology. The minimum worst-case distances are obtained for phase margin and slew rate. A further improvement in one of these two worst-case distances would decrease the worst-case distance of the other one and hence reduce the overall robustness, which cannot be better than the minimum of all worst-case distances. The optimum can be seen as a center between competing worst-case distances and corresponding specification bounds. This explains the synonymous term "design centering" for yield optimization. In this example, the design center is a robustness tradeoff regarding speed (slew rate) and stability (phase margin).

As a side comment, please note that the **worst-case performance features** $\varphi_{WL/U,i},\ i = 1,\ldots,n_\varphi$ can also be objective of an optimization.

In case there are no statistical parameters, the concept of the worst-case distance as performance safety margin divided by performance variability can be used to define a corresponding objective for nominal design, called **performance distance**:

$$\alpha_{WL/U,i} = \frac{|\overline{\varphi_i}\left(\mathbf{x}_{d,0}\right) - \varphi_{L/U,i}|}{\sqrt{\nabla\varphi_i\left(\mathbf{x}_{d,0}\right)^T \cdot \mathbf{A}_{x_d}^2 \cdot \nabla\varphi_i\left(\mathbf{x}_{d,0}\right)}} \quad \begin{matrix}\text{"performance safety margin"} \\ \text{"sensitivity"}\end{matrix} \tag{349}$$

with

$$\overline{\varphi}_i\left(\mathbf{x}_d\right) = \varphi_i\left(\mathbf{x}_{d,0}\right) + \nabla\varphi_i\left(\mathbf{x}_{d,0}\right)^T \cdot \left(\mathbf{x}_d - \mathbf{x}_{d,0}\right) \tag{350}$$

$$\mathbf{A}_{x_d} = \begin{bmatrix} a_{xd,1} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & a_{xd,n_{x_d}} \end{bmatrix} \qquad \begin{pmatrix} \text{scaling of individual} \\ \text{design parameters} \end{pmatrix} \tag{351}$$

10.2 Derivatives of optimization objectives

Derivatives of the statistically estimated yield

For a yield estimated with a Monte-Carlo analysis, both the first-order derivative and the second-order derivative with regard to the mean values of statistical parameters can be calculated. They are given in the following. For the derivation, please see App. I.

$$\nabla Y\left(\mathbf{x}_{s,0}\right) = Y \cdot \mathbf{C}^{-1} \cdot \left(\mathbf{x}_{s,0,\delta} - \mathbf{x}_{s,0}\right) \tag{352}$$

$$\mathbf{x}_{s,0,\delta} = \underset{pdf_\delta}{E}\left\{\mathbf{x}_s\right\}, \quad pdf_\delta\left(\mathbf{x}_s\right) = \frac{1}{Y} \cdot \delta\left(\mathbf{x}_s\right) \cdot pdf\left(\mathbf{x}_s\right) \tag{353}$$

$$\nabla^2 Y\left(\mathbf{x}_{s,0}\right) = Y \cdot \mathbf{C}^{-1} \cdot \left[\mathbf{C}_\delta + \left(\mathbf{x}_{s,0,\delta} - \mathbf{x}_{s,0}\right) \cdot \left(\mathbf{x}_{s,0,\delta} - \mathbf{x}_{s,0}\right)^T - \mathbf{C}\right] \cdot \mathbf{C}^{-1} \tag{354}$$

$$\mathbf{C}_\delta = \underset{pdf_\delta}{E}\left\{\left(\mathbf{x}_s - \mathbf{x}_{s,0,\delta}\right) \cdot \left(\mathbf{x}_s - \mathbf{x}_{s,0,\delta}\right)\right\} \tag{355}$$

Please note the variables with an index δ. These are calculated for the probability density function that results after truncation by the performance specification. Fig. 48 illustrates the truncation of the original probability density function by the performance specification.

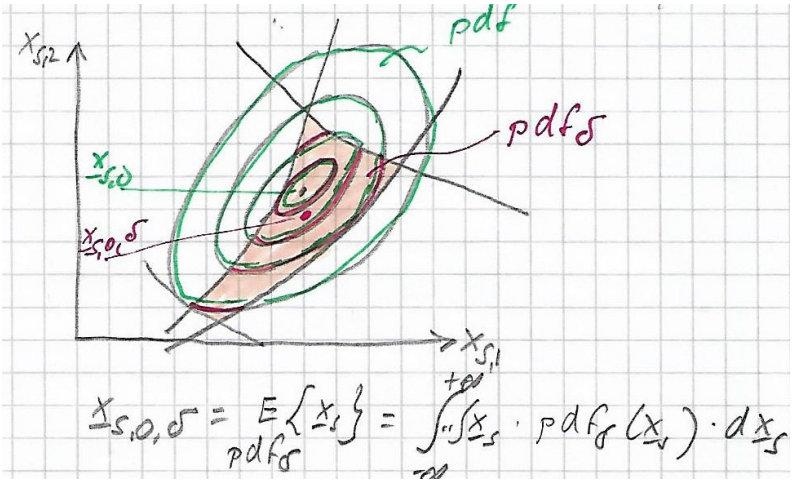


Figure 48. Center of gravity of truncated probability density function.

The truncated probability density function pdf_δ is the rest of the original pdf obtained over the area shaded in red. According to (352), the stationary point of the yield maximization problem is where the mean values of truncated pdf and original pdf coincide. The mean value of a pdf is also representing the center of gravity of the pdf. That means for a maximum yield, the centers of gravity of pdf and truncated pdf are identical, i.e., if you balance the optimal mean value of a two-dimensional pdf on the tip of your finger, it does not tilt if you truncate it according to the performance specification.

Statistical yield optimization methods are therefore also called "center-of-gravity" approaches.

Please note that a yield gradient $\nabla Y(\mathbf{x}_d)$ with regard to design parameters cannot be determined based on a Monte Carlo analysis. Some finite-difference approach with additional Monte-Carlo sampling is required in this case. This makes yield optimization based on Monte-Carlo analysis extremely expensive.

First-order derivative of the worst-case distance

In contrast to the statistical yield estimate, a second-order derivative of the worst-case distance is unknown. On the other hand, the first-order derivative of the worst-case distance can be determined for **both deterministic and statistical parameters**. Based on (347) we obtain:

$$\nabla\beta_{WL/U,i}(\mathbf{x}_{s,0}) = \frac{\pm 1}{\sigma_{\overline{\varphi}^{(WL/U,i)}}} \cdot \nabla\varphi_i\left(\mathbf{x}_{s,WL/U,i}\right) \tag{356}$$

$$\nabla\beta_{WL/U,i}\left(\mathbf{x}_d^{(\kappa)}\right) = \frac{\pm 1}{\sigma_{\overline{\varphi}^{(WL/U,i)}}} \cdot \nabla\varphi_i\left(\mathbf{x}_d^{(\kappa)}\right) \tag{357}$$

The worst-case distance considers both its change by moving the center $\mathbf{x}_{s,0}$ away from the specification curve in the parameter space and its change by moving away this specification curve from the center through changes in design parameters $\mathbf{x}_d$, as illustrated in Fig. 46.

As a side comment, the analogous derivative of the performance distance is:

$$\nabla\alpha_{WL/U,i}(\mathbf{x}_{d,0}) = \frac{\pm 1}{\sqrt{\nabla\varphi(\mathbf{x}_{d,0})^T \mathbf{A}_{x_d}^2 \nabla\varphi(\mathbf{x}_{d,0})}} \nabla\varphi(\mathbf{x}_{d,0}) \tag{358}$$

10.3 Problem formulations of analog optimization

Nominal design

The nominal design problem can be formulated as a constrained optimization problem with performance features or performance distances as objective:

$$\min_{\mathbf{x}_d} \pm\varphi_i\left(\mathbf{x}_d\right), i = 1, \cdots, n_\varphi \quad \text{s.t.} \quad \mathbf{c}\left(\mathbf{x}_d\right) \geq \mathbf{0} \tag{359}$$

$$\begin{aligned} \max_{\mathbf{x}_d} \pm\alpha_{WL/U,i}\left(\mathbf{x}_d\right) \quad & , i = 1, \cdots, n_{PSF} \quad \text{s.t.} \quad \mathbf{c}\left(\mathbf{x}_d\right) \geq \mathbf{0} \\ \text{“nominal inside”} \quad & : + \\ \text{“nominal outside”} \quad & : - \end{aligned} \tag{360}$$

This is a multicriteria optimization problem (MCO), which has to be scalarized to apply optimization methods as presented before.

Design centering/yield optimization

Yield optimization can be formulated as a constrained optimization problem with yield as an objective:

$$\max_{\mathbf{x}_d} Y\left(\mathbf{x}_d\right) \quad \text{s.t.} \quad \mathbf{c}\left(\mathbf{x}_d\right) \geq \mathbf{0} \tag{361}$$

or with worst-case distances as objectives (”geometric yield optimization”):

$$\begin{aligned} \max_{\mathbf{x}_d} \pm\beta_{WL/U,i}\left(\mathbf{x}_d\right) \quad & , i = 1, \ldots, n_{PSF} \quad \text{s.t.} \quad \mathbf{c}\left(\mathbf{x}_d\right) \geq \mathbf{0} \\ \text{“nominal inside”} \quad & : + \\ \text{“nominal outside”} \quad & : - \end{aligned} \tag{362}$$

Yield optimization based on worst-case distances is a multicriteria optimization problem (MCO), which has to be scalarized to apply optimization methods as presented before.

Scalarization of multicriteria optimization problems

The task is to map a multicriteria optimization problem with multiple objectives o_i, $i = 1, \ldots, n_o$ onto a scalar optimization problem with objective f:

$$\min o_i(\mathbf{x}_d), i = 1, \ldots, n_o \qquad \rightarrow \qquad \min f(\mathbf{x}_d) \tag{363}$$

The following typical scalarization approaches are based on vector norms:

$$\text{weighted sum: } f_{l1}(\mathbf{x}) = \sum_{i=1}^{n_o} w_i \cdot o_i(\mathbf{x}) \tag{364}$$

$$\text{weighted least squares: } f_{l2}(\mathbf{x}) = \sum_{i} w_i \cdot \left(o_i(\mathbf{x}) - o_{i,target}\right)^2 \tag{365}$$

$$\text{weighted min/max: } f_{l\infty}(\mathbf{x}) = \max_i w_i \cdot o_i \tag{366}$$

$$w_i > 0,\ i = 1, \ldots, n_o\ , \quad \sum_{i=1}^{n_o} w_i = 1 \tag{367}$$

The weighted sum approach corresponds to the l_1-norm, the least-squares approach corresponds to the l_2-norm, and the min/max approach corresponds to the l_∞-norm.

A popular geometric approach for yield optimization inscribes a maximum hyperellipsoid into the parameter acceptance region. If worst-case distances are applied, an inner iteration step κ of optimization can be to inscribe the largest hyperellipsoid into the linearized acceptance region by linear programming:

$$\max_{f_\beta, \mathbf{x}_d} f_\beta \text{ s.t. } \beta_{w,i}^{(\kappa)} + \nabla\beta_{w,i}^T(\mathbf{x}_d^{(\kappa)}) \cdot (\mathbf{x}_d - \mathbf{x}_d^{(\kappa)}) \geq f_\beta \ , i = 1, \ldots, n_{PSF} \qquad (368)$$
$$\left[\overline{\mathbf{c}}^{(\kappa)}(\mathbf{x}_d) \geq 0\right]$$

An exponential cost function over worst-case distances is also possible but adds strong nonlinearity to the objective function:

$$\max \pm\beta_{WL/U,i}(\mathbf{x}_d) \qquad \rightarrow \qquad \min \sum_i e^{-w_i\left(\pm\beta_{WL/U,i}(\mathbf{x}_d)\right)} \qquad (369)$$

10.4 Input/output of yield optimization with worst-case distances

As an overview, Fig. 49 shows how yield optimization based on worst-case distances calls geometric yield analysis. As yield optimization is a nonlinear optimization problem, it will call yield analysis in each of its $n_{it,yo}$ iteration steps to obtain the values and gradients of the n_{PSF} worst-case distances of the n_{PSF} performance specification features. That means it has to call a geometric yield analysis for each performance specification feature. In turn, getting the worst-case distance for one performance specification feature is its own nonlinear optimization problem, solved iteratively. In each of the $n_{it,gya}$ iteration steps of its iterative optimization process, a sensitivity analysis is called to get the performance sensitivity with regard to statistical parameters. Once the worst-case distance has been determined, a sensitivity analysis with regard to deterministic (design) parameters is performed to determine the gradient of the worst-case distance with regard to these deterministic (design) parameters. The worst-case range parameters are usually determined only once at the beginning of the overall optimization process and then kept for the rest of the yield optimization process. At the end of the optimization process, their validity is re-checked; the whole process is re-started only if they change (their worst-case parameter set).

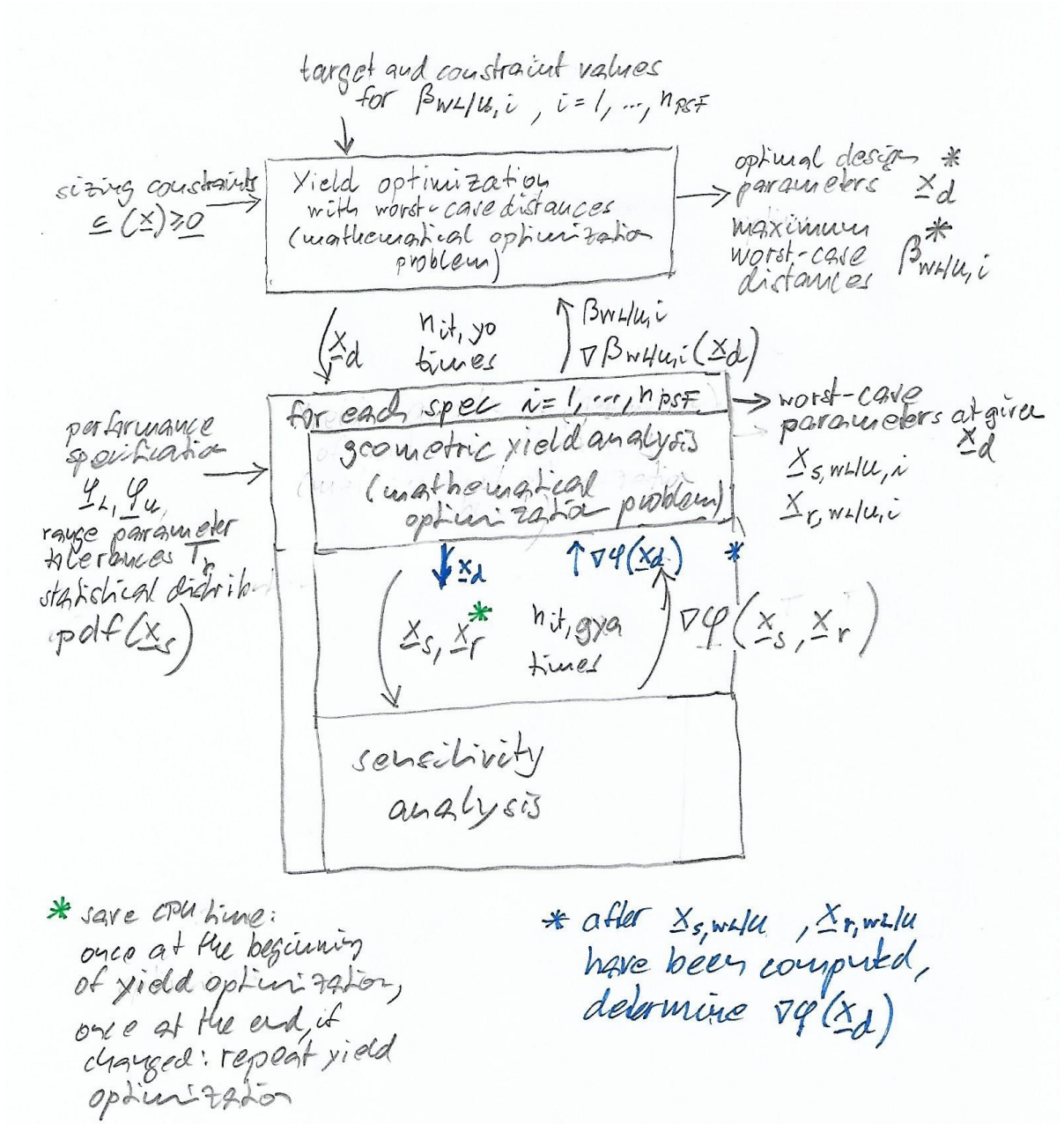


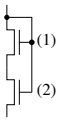
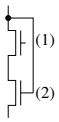
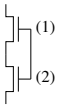
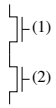
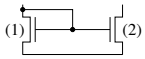
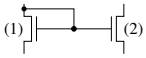
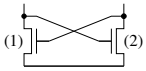
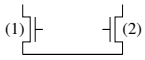
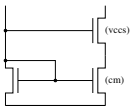
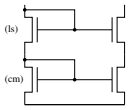
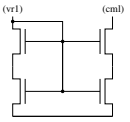
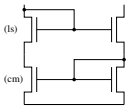
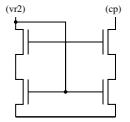
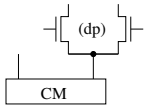
Figure 49. Yield optimization calls yield analysis for each performance specification feature in each iteration step of its iterative solution process. Yield analysis in turn calls sensitivity analysis for the performance features in each iteration step of its iterative solution process.

11 Sizing rules for analog circuit optimization

- Design constraints for geometries and currents/voltages of transistors
- Ensure function and robustness
- Automatic construction for given circuit netlist
  - hierarchical library of transistor groups
  - structural analysis of netlist

[Graeb, Zizala, Eckmueller, Antreich: *The Sizing Rules Method for Analog Integrated Circuit Design*, IEEE International Conference on Computer-Aided Design (ICCAD), 2001]
[Massier, Graeb, Schlichtmann: *The Sizing Rules Method for CMOS and Bipolar Analog Integrated Circuit Synthesis*, IEEE TCAD 2008]

11.1 Library of NMOS transistor groups

Function	Schematic	Sub-Library
Voltage-Controlled Resistor (vcres) Volt.-Contr. Current Source (vccs)		L_0
Voltage Reference 1 (vr1)		L_1
Voltage Reference 2 (vr2)		
Current Mirror Load (cml)		
Cascode Pair (cp)		
Simple Current Mirror (cm)		
Level Shifter (ls)		
Cross-Coupled Pair (cc)		
Differential Pair (dp)		
Wilson Current Mirror (WCM)		L_2
Cascode Current Mirror (CCM)		
4-Transistor Current Mirror (4TCM)		
Improved Wilson Current Mirror (IWCM)		
Wide Swing Cascode Current Mirror (WSCCM)		
Differential Stage (DS)		L_3

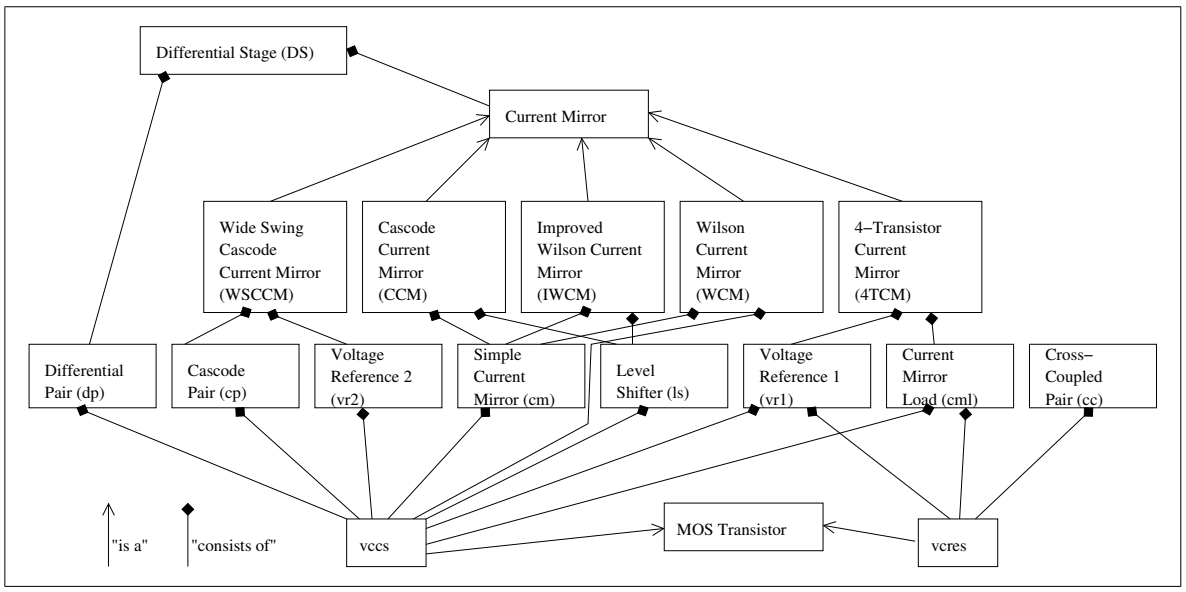


Figure 50.

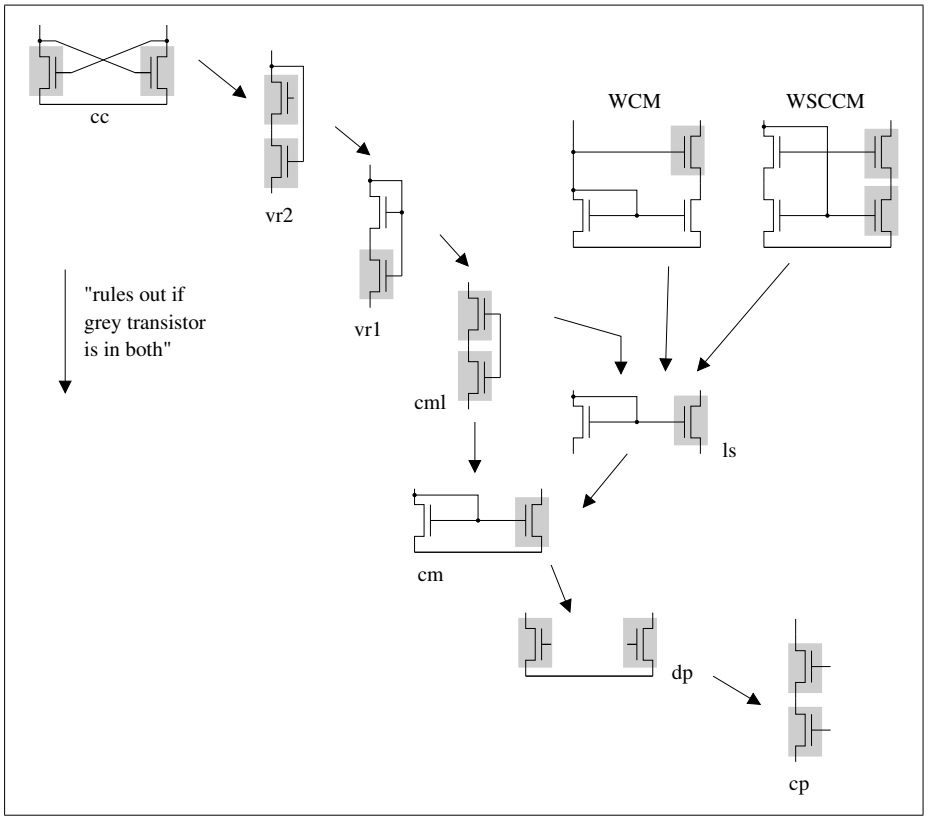


Figure 51. Ambiguity Arbitration Rules

11.2 Single (NMOS) transistor

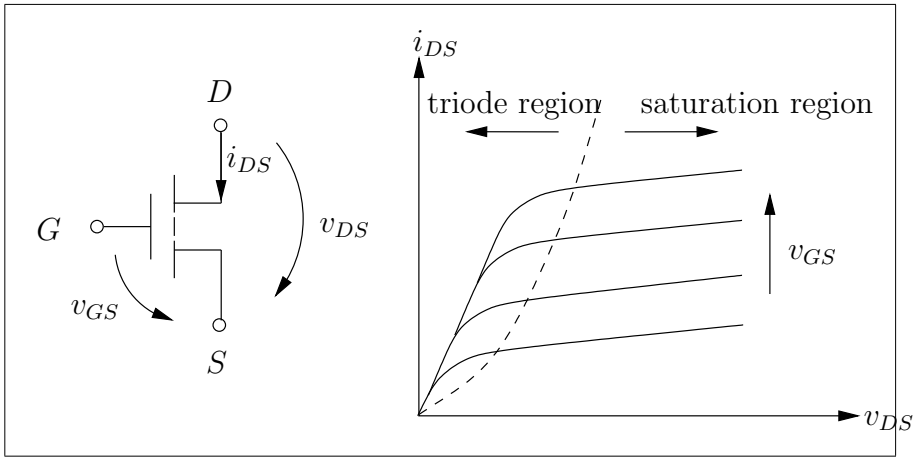


Figure 52.

drain-source current (simple model)

$$i_{DS} = \begin{cases} 0, & v_{DS} < 0 \\ \mu \cdot C_{ox} \cdot \frac{W}{L} \cdot \left(v_{GS} - V_{th} - \frac{v_{DS}}{2}\right) \cdot v_{DS} \cdot (1 + \lambda \cdot v_{DS}), & 0 \le v_{DS} \le v_{GS} - V_{th} \\ \frac{1}{2} \cdot \mu \cdot C_{ox} \cdot \frac{W}{L} \cdot (v_{GS} - V_{th})^2 \cdot (1 + \lambda \cdot v_{DS}), & v_{GS} - V_{th} \le v_{DS} \end{cases} \tag{370}$$

$$\begin{array}{ll} W, L\,[m] & : \text{transistor width and length} \\ \mu_n \left[\frac{m^2}{V \cdot s}\right] & : \text{electron mobility in silicon} \\ C_{ox} \left[\frac{F}{cm^2}\right] & : \text{capacitance per area due to oxide} \\ V_{th}\,[V] & : \text{threshold voltage} \\ \lambda[\frac{1}{V}] & : \text{channel length modulation factor} \end{array} \tag{371}$$

saturation region, no channel length modulation:

$$i_{DS} = K \cdot \frac{W}{L} \cdot (v_{GS} - V_{th})^2 \quad \text{with} \quad K = \frac{1}{2} \cdot \mu_n \cdot C_{ox} \tag{372}$$

derivatives:

$$\begin{array}{rcl} \nabla i_{DS}\,(K) & = & i_{DS}/K \\ \nabla i_{DS}\,(W) & = & i_{DS}/W \\ \nabla i_{DS}\,(L) & = & -i_{DS}/L \\ \nabla i_{DS}\,(V_{th}) & = & -\frac{2}{v_{GS} - V_{th}} \cdot i_{DS} \end{array} \tag{373}$$

variances:

$$\left(\frac{\sigma_K}{K}\right)^2 = \frac{A_K}{W \cdot L} \tag{374}$$

$$\left(\sigma_{V_{th}}\right)^2 = \frac{A_{V_{th}}}{W \cdot L} \tag{375}$$

area law: Lakshmikumar et al. *IEEE Journal of solid-state circuits* SC-21, Dec.1986

i_{DS} variance, **assumptions**: K, W, L, V_{th} variations statistically independent, saturation region, no channel length modulation, linear transformation of variances

$$\sigma_{i_{DS}}^2 \approx \sum_{x \in \{K,W,L,V_{th}\}} \left[\nabla i_{DS}\left(x\right)\right]^2 \cdot \sigma_x^2 \tag{376}$$

$$\boxed{\frac{\sigma_{i_{DS}}^2}{i_{DS}^2} \approx \frac{A_k}{W \cdot L} + \frac{\sigma_W^2}{W^2} + \frac{\sigma_L^2}{L^2} + \frac{4}{\left(v_{GS} - V_{th}\right)^2} \cdot \frac{A_{V_{th}}}{W \cdot L}} \tag{377}$$

$W \uparrow,\ L \uparrow,\ W \cdot L \uparrow \rightarrow\ \sigma_{i_{DS}} \downarrow$

larger transistor geometries reduce i_{DS} variance due to manufacturing variations in K, W, L, V_{th}

11.2.1 Sizing rules for a single transistor that acts as a voltage-controlled current source (vccs)

	type	origin
(1) $v_{DS} \geq 0$	electrical (DC)	transistor
(2) $v_{GS} - V_{th} \geq 0$	function	in saturation
(3) $v_{DS} - (v_{GS} - V_{th}) \geq V_{sat,min}^{\star}$		
(4) $W \geq W_{min}^{\star}$	geometry	manufacturing
(5) $L \geq L_{min}^{\star}$	robustness	variations affect i_{DS}
(6) $W \cdot L \geq A_{min}^{\star}$		

$\star$ technology–specific value

11.2.2 Sizing rules for a single transistor that acts as a voltage-controlled resistor (vcres)

	type	origin
(1) $v_{DS} \geq 0$	electrical (DC)	transistor
(2) $v_{GS} - V_{th} \geq 0$	function	in linear region
(3) $(v_{GS} - V_{th}) - v_{DS} \geq V_{lin,min}^{\star}$		

$\star$ technology–specific value

11.3 Transistor pairs (NMOS)

11.3.1 Simple current mirror

function: $I_2 = x \cdot I_1$

assumptions: $K_1 = K_2;\ \lambda_1 = \lambda_2 = 0;\ V_{th1} = V_{th2};\ L_1 = L_2;\ T_1,\ T_2$ in saturation

$$x = \frac{I_2}{I_1} = \frac{W_2}{W_1} \cdot \frac{(v_{GS} - V_{th2})^2}{(v_{GS} - V_{th1})^2} \cdot \frac{1 + \lambda_2 \cdot v_{DS2}}{1 + \lambda_1 \cdot v_{DS1}} \tag{378}$$

Sizing rules for simple current mirror

in addition to sec. 11.2.1 (1) – (6):

	type	origin
(1) $L_1 = L_2$	geometric	current ratio
(2) $x = \frac{W_2}{W_1}$	function/robustness	layout (matching)
(3) $\lvert v_{DS1} - v_{DS2} \rvert \leq \Delta V^{\star}_{DSmax}$	electrical function	influence of channel length modulation
(4) $v_{GS} - V_{th1,2} \geq V^{\star}_{GSmin}$	electrical robustness	influence of local variations in threshold voltages

$\star$ technology-specific value

11.3.2 Level shifter

	type	origin
(1) $L_1 = L_2$	geometric/function	systematic mismatch
(2) $\lvert v_{GS} - V_{th_{1,2}} \rvert \leq V^{\star}_{GS,min}$	electrical	influence of local variations in threshold voltages

$\star$ technology-specific value

11.3.3 Differential pair

	type	origin
(1) $L_1 = L_2$ (2) $W_1 = W_2$	geometric/function	symmetry
(3) $\lvert v_{DS_2} - v_{DS_1} \rvert \leq \Delta V^{\star}_{DS,max(dp)}$	electrical/function	linear behaviour
(4) $\lvert v_{GS_2} - v_{GS_1} \rvert \leq \Delta V^{\star}_{GS,max}$	electrical/robustness	influence of local variations

$\star$ technology-specific value

11.3.4 Cross-coupled pair

	type	origin
(1) $L_1 = L_2$ (2) $W_1 = W_2$	geometric/function	symmetry

11.4 Transistor pair groups

11.4.1 Cascode current mirror

	type	origin
(1) $L_{ls(1)} = L_{ls(2)}$ (2) $W_{ls(1)} = W_{ls(2)}$	geometric/function	same voltage at source pins of ls

11.4.2 4-Transistor current mirror

	type	origin
(1) $W_{vr1(1)} = W_{vr1(2)}$ (2) $W_{cml(1)} = W_{cml(2)}$	geometric/function	same voltage at source pins of upper pair
(3) $\lvert v_{DS_{vr1(2)}} - v_{DS_{cml(2)}} \rvert \leq \Delta V^{\star}_{DS,max(4TCM)}$	electrical/function	linear behaviour

$\star$ technology-specific value

11.4.3 Wide swing cascode current mirror

upper pair: level shifter constraints
lower pair: simple current mirror constraints

11.4.4 Wilson current mirror

upper transistor: vcres constraints
lower pair: simple current mirror constraints

11.4.5 Improved Wilson current mirror

upper pair: level shifter constraints
lower pair: simple current mirror constraints

Appendix A Matrix and vector notations

A.1 Vector

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} \vdots \\ b_i \\ \vdots \end{bmatrix}, \quad b_i \in \mathbb{R}, \quad \mathbf{b} \in \mathbb{R}^m, \quad \mathbf{b}_{<m>} \tag{379}$$

$\mathbf{b}$: column vector

$$\mathbf{b}^T = [\cdots \quad b_i \quad \cdots] \quad \text{transposed vector} \tag{380}$$

$\mathbf{b}^T$: row vector

A.2 Matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = \begin{bmatrix} & \cdots & \\ \vdots & a_{ij} & \vdots \\ & \cdots & \end{bmatrix}, \quad a_{ij} \in \mathbb{R}, \quad \mathbf{A} \in \mathbb{R}^{m \times n}, \quad \mathbf{A}_{<m \times n>} \tag{381}$$

i: row index, j: column index

$$\mathbf{A} = [\mathbf{a}_{\diamond 1} \quad \mathbf{a}_{\diamond 2} \quad \cdots \quad \mathbf{a}_{\diamond n}] \tag{382}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_{1\diamond}^T \\ \mathbf{a}_{2\diamond}^T \\ \vdots \\ \mathbf{a}_{m\diamond}^T \end{bmatrix} \tag{383}$$

transposed matrix: $\mathbf{A}^T_{<n \times m>}$

$$\mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_{\diamond 1}^T \\ \mathbf{a}_{\diamond 2}^T \\ \vdots \\ \mathbf{a}_{\diamond n}^T \end{bmatrix} = [\mathbf{a}_{1\diamond} \mathbf{a}_{2\diamond} \cdots \mathbf{a}_{m\diamond}] \tag{384}$$

A.3 Addition

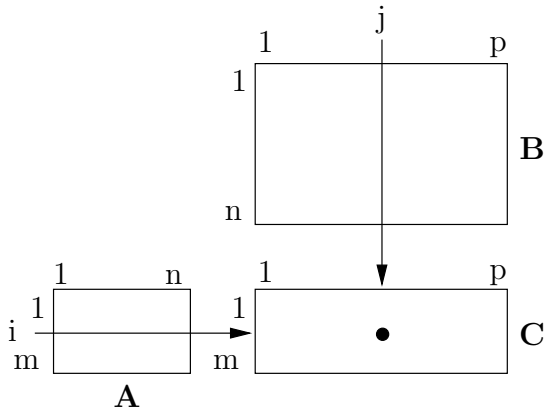
$$\mathbf{A}_{<m\times n>} + \mathbf{B}_{<m\times n>} = \mathbf{C}_{<m\times n>} \tag{385}$$

$$c_{ij} = a_{ij} + b_{ij} \tag{386}$$

A.4 Multiplication

$$\mathbf{A}_{<m\times n>} \cdot \mathbf{B}_{<n\times p>} = \mathbf{C}_{<m\times p>} \tag{387}$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \tag{388}$$



$$c_{ij} = \underbrace{\mathbf{a}_{i\diamond}^{T}}_{\substack{i\text{-th row} \\ \text{in } \mathbf{A} \text{ (387)}}} \cdot \underbrace{\mathbf{b}_{\diamond j}}_{\substack{j\text{-th column} \\ \text{in } \mathbf{B} \text{ (387)}}} \tag{389}$$

A.5 Special cases

$$\mathbf{a} = \mathbf{A}_{<m\times 1>} \tag{390}$$

$$\mathbf{a}^T = \mathbf{A}_{<1\times m>} \tag{391}$$

$$a = \mathbf{A}_{<1\times 1>} = \mathbf{a}_{<1>} = \mathbf{a}^T_{<1>} \tag{392}$$

$$\mathbf{a}^T_{<m>} \cdot \mathbf{b}_{<m>} = c \text{ scalar product} \tag{393}$$

$$\mathbf{a}_{<m>} \cdot \mathbf{b}^T_{<n>} = \mathbf{C}_{<m\times n>} \text{ dyadic product} \tag{394}$$

$$\mathbf{A}_{<m\times n>} \cdot \mathbf{b}_{<n>} = \mathbf{c}_{<m>} \tag{395}$$

$$\mathbf{b}^T_{<m>} \cdot \mathbf{A}_{<m\times n>} = \mathbf{c}^T_{<n>} \tag{396}$$

$$\mathbf{b}^T_{<m>} \cdot \mathbf{A}_{<m\times n>} \cdot \mathbf{c}_{<n>} = d \tag{397}$$

$$\text{identity matrix } \mathbf{I}_{<m\times m>} = \begin{bmatrix} 1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & 1 \end{bmatrix} \tag{398}$$

$$\text{diagonal matrix } \operatorname{diag}(\mathbf{a}_{<m>}) = \begin{bmatrix} a_1 & & & \mathbf{0} \\ & a_2 & & \\ & & \ddots & \\ \mathbf{0} & & & a_m \end{bmatrix}_{<m\times m>} \tag{399}$$

A.6 Determinant of a quadratic matrix

$$\begin{aligned}\det\left(\mathbf{A}_{<m\times m>}\right) = |\mathbf{A}| &= \sum_{j=1}^{m} a_{ij}\cdot\alpha_{ij},\ \ i\in 1,\ldots,m \\ &= \sum_{i=1}^{m} a_{ij}\cdot\alpha_{ij},\ \ j\in 1,\ldots,m \end{aligned} \tag{400}$$

adjugate matrix $\mathrm{adj}\left(\mathbf{A}\right)$

$$\mathrm{adj}\left(\mathbf{A}\right) = \begin{bmatrix} & \cdots & \\ \vdots & \alpha_{ij} & \vdots \\ & \cdots & \end{bmatrix}^{T} \tag{401}$$

minor determinant (row i, column j deleted):

$$\alpha_{ij} = (-1)^{i+j}\ \cdot \left| \begin{bmatrix} \cdots & & \\ & a_{ij} & \\ \cdots & & \end{bmatrix} \right|$$

A.7 Inverse of a quadratic non-singular matrix

$$\mathbf{A}^{-1}_{<m\times m>}\cdot\mathbf{A}_{<m\times m>} = \mathbf{A}\cdot\mathbf{A}^{-1} = \mathbf{I} \tag{402}$$

$$\mathbf{A}^{-1} = \frac{1}{\det\left(\mathbf{A}\right)}\cdot\mathrm{adj}\left(\mathbf{A}\right) \tag{403}$$

A.8 Some properties

$$(\mathbf{A}\cdot\mathbf{B})\cdot\mathbf{C}=\mathbf{A}\cdot(\mathbf{B}\cdot\mathbf{C}) \tag{404}$$

$$\mathbf{A}\cdot(\mathbf{B}+\mathbf{C})=\mathbf{A}\cdot\mathbf{B}+\mathbf{A}\cdot\mathbf{C} \tag{405}$$

$$(\mathbf{A}\cdot\mathbf{B})^T=\mathbf{B}^T\cdot\mathbf{A}^T \tag{406}$$

$$\left(\mathbf{A}^T\right)^T=\mathbf{A} \tag{407}$$

$$(\mathbf{A}+\mathbf{B})^T=\mathbf{A}^T+\mathbf{B}^T \tag{408}$$

$$\mathbf{A}\text{ symmetric}\iff\mathbf{A}=\mathbf{A}^T \tag{409}$$

$$\mathbf{A}\text{ positive (semi)definite}\iff\underset{\mathbf{x}\neq\mathbf{0}}{\forall}\mathbf{x}^T\cdot\mathbf{A}\cdot\mathbf{x}>(\geq)\ 0 \tag{410}$$

$$\mathbf{A}\text{ negative (semi)definite}\iff\underset{\mathbf{x}\neq\mathbf{0}}{\forall}\mathbf{x}^T\cdot\mathbf{A}\cdot\mathbf{x}<(\leq)\ 0 \tag{411}$$

$$\mathbf{I}^T=\mathbf{I} \tag{412}$$

$$\mathbf{A}\cdot\mathbf{I}=\mathbf{I}\cdot\mathbf{A}=\mathbf{A} \tag{413}$$

$$\mathbf{A}\cdot\mathrm{adj}\,(\mathbf{A})=\det(\mathbf{A})\cdot\mathbf{I}=\mathrm{adj}\,(\mathbf{A})\cdot\mathbf{A} \tag{414}$$

$$\det\left(\mathbf{A}^T\right)=\det(\mathbf{A}) \tag{415}$$

$$|\mathbf{a}_{\diamond 1}\ \cdots\ b\cdot\mathbf{a}_{\diamond j}\ \cdots\ \mathbf{a}_{\diamond m}|=b\cdot\det(\mathbf{A}) \tag{416}$$

$$\det\left(b\cdot\mathbf{A}^T\right)=b^m\cdot\det(\mathbf{A}) \tag{417}$$

$$|\mathbf{a}_{\diamond 1}\ \cdots\ \mathbf{a}_{\diamond j}^{(1)}\ \cdots\ \mathbf{a}_{\diamond m}|+|\mathbf{a}_{\diamond 1}\ \cdots\ \mathbf{a}_{\diamond j}^{(2)}\ \cdots\ \mathbf{a}_{\diamond m}|=|\mathbf{a}_{\diamond 1}\ \cdots\ \mathbf{a}_{\diamond j}^{(1)}+\mathbf{a}_{\diamond j}^{(2)}\ \cdots\ \mathbf{a}_{\diamond m}| \tag{418}$$

$$|\mathbf{a}_{\diamond 2}\ \mathbf{a}_{\diamond 1}\ \mathbf{a}_{\diamond 3}\ \cdots\ \mathbf{a}_{\diamond m}|=-\det(\mathbf{A}) \tag{419}$$

$$\mathbf{a}_{\diamond j}=\mathbf{a}_{\diamond k}\ (\mathbf{A}\text{ rank deficient}):\det(\mathbf{A})=0 \tag{420}$$

$$|\mathbf{a}_{\diamond 1}\ \cdots\ \mathbf{a}_{\diamond j}+b\cdot\mathbf{a}_{\diamond k}\ \cdots\ \mathbf{a}_{\diamond m}|=\det(\mathbf{A}) \tag{421}$$

$$\det(\mathbf{A}\cdot\mathbf{B})=\det(\mathbf{A})\cdot\det(\mathbf{B}) \tag{422}$$

$$(\mathbf{A}\cdot\mathbf{B})^{-1}=\mathbf{B}^{-1}\cdot\mathbf{A}^{-1} \tag{423}$$

$$\left(\mathbf{A}^T\right)^{-1}=\left(\mathbf{A}^{-1}\right)^T \tag{424}$$

$$\left(\mathbf{A}^{-1}\right)^{-1}=\mathbf{A} \tag{425}$$

$$\det\left(\mathbf{A}^{-1}\right)=\frac{1}{\det(\mathbf{A})} \tag{426}$$

$$\mathbf{A}^{-1}=\mathbf{A}^T\iff\mathbf{A}\text{ orthogonal} \tag{427}$$

Appendix B Notations and abbreviations for first- and second-order derivatives

Gradient vector: vector of first partial derivatives

$$\nabla_{\mathbf{x}_s} f(\mathbf{x})_{<n_{xs}>} = \left.\frac{\partial f}{\partial \mathbf{x}_s}\right|_{\mathbf{x}} = \left.\begin{bmatrix} \frac{\partial f}{\partial x_{s,1}} \\ \frac{\partial f}{\partial x_{s,2}} \\ \vdots \\ \frac{\partial f}{\partial x_{s,n_{xs}}} \end{bmatrix}\right|_{\mathbf{x}=\begin{bmatrix} \mathbf{x}_d \\ \mathbf{x}_s \\ \mathbf{x}_r \end{bmatrix}} \tag{428}$$

Abbreviation:

$$\nabla f(\mathbf{x}'_s) = \left.\frac{\partial f}{\partial \mathbf{x}_s}\right|_{\mathbf{x}=\begin{bmatrix} \mathbf{x}_d \\ \mathbf{x}'_s \\ \mathbf{x}_r \end{bmatrix}} \tag{429}$$

Hessian matrix: matrix of second partial derivatives

Abbreviation:

$$\nabla^2 f(\mathbf{x}'_s)_{<n_{xs}\times n_{xs}>} = \left.\frac{\partial^2 f}{\partial \mathbf{x}_s \cdot \partial \mathbf{x}_s^T}\right|_{\mathbf{x}=\begin{bmatrix} \mathbf{x}_d \\ \mathbf{x}'_s \\ \mathbf{x}_r \end{bmatrix}} \tag{430}$$

$$= \left.\begin{bmatrix} \frac{\partial^2 f}{\partial x_{s,1}^2} & \frac{\partial^2 f}{\partial x_{s,1}\cdot\partial x_{s,2}} & \cdots & \frac{\partial^2 f}{\partial x_{s,1}\cdot\partial x_{s,n_{xs}}} \\ \frac{\partial^2 f}{\partial x_{s,2}\cdot\partial x_{s,1}} & \frac{\partial^2 f}{\partial x_{s,2}^2} & \cdots & \frac{\partial^2 f}{\partial x_{s,2}\cdot\partial x_{s,n_{xs}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_{s,n_{xs}}\cdot\partial x_{s,1}} & \frac{\partial^2 f}{\partial x_{s,n_{xs}}\cdot\partial x_{s,2}} & \cdots & \frac{\partial^2 f}{\partial x_{s,n_{xs}}^2} \end{bmatrix}\right|_{\mathbf{x}=\begin{bmatrix} \mathbf{x}_d \\ \mathbf{x}'_s \\ \mathbf{x}_r \end{bmatrix}} \tag{431}$$

Abbreviation:

$$\nabla f\left(\mathbf{x}'_d, \mathbf{x}'_s\right)_{<(n_{xd}+n_{xs})\times(n_{xd}+n_{xs})>} = \left[\begin{array}{cc} \frac{\partial^2 f}{\partial \mathbf{x}_d \cdot \partial \mathbf{x}_d^T} & \frac{\partial^2 f}{\partial \mathbf{x}_d \cdot \partial \mathbf{x}_s^T} \\ \frac{\partial^2 f}{\partial \mathbf{x}_s \cdot \partial \mathbf{x}_d^T} & \frac{\partial^2 f}{\partial \mathbf{x}_s \cdot \partial \mathbf{x}_s^T} \end{array}\right]\Bigg|_{\mathbf{x}=\left[\begin{array}{c} \mathbf{x}'_d \\ \mathbf{x}'_s \\ \mathbf{x}_r \end{array}\right]} \tag{432}$$

$$= \left[\begin{array}{cc} \nabla^2 f\left(\mathbf{x}_d\right) & \frac{\partial^2 f}{\partial \mathbf{x}_d \cdot \partial \mathbf{x}_s^T} \\ \frac{\partial^2 f}{\partial \mathbf{x}_s \cdot \partial \mathbf{x}_d^T} & \nabla^2 f\left(\mathbf{x}_s\right) \end{array}\right]\Bigg|_{\mathbf{x}=\left[\begin{array}{c} \mathbf{x}'_d \\ \mathbf{x}'_s \\ \mathbf{x}_r \end{array}\right]} \tag{433}$$

If no index of ∇ or ∇^2 is given, then the corresponding derivative $\nabla f(\cdot)$ refers to the (sub)set of parameters in the argument $(\cdot)$.

The parameter vector may be for a subset, e.g., for statistical parameters $\mathbf{x}_s$. Then, the remaining parameters are at a value that is not considered.

Jacobian matrix, sensitivity matrix

Abbreviation:

$$\nabla \mathbf{f}\left(\mathbf{x}_s'^T\right)_{<n_f\times n_{xs}>} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}_s^T}\Bigg|_{\mathbf{x}=\left[\begin{array}{c} \mathbf{x}_d \\ \mathbf{x}'_s \\ \mathbf{x}_r \end{array}\right]} \tag{434}$$

$$= \left[\begin{array}{cccc} \frac{\partial f_1}{\partial x_{s,1}} & \frac{\partial f_1}{\partial x_{s,2}} & \cdots & \frac{\partial f_1}{\partial x_{s,n_{xs}}} \\ \frac{\partial f_2}{\partial x_{s,1}} & \frac{\partial f_2}{\partial x_{s,2}} & \cdots & \frac{\partial f_2}{\partial x_{s,n_{xs}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{n_f}}{\partial x_{s,1}} & \frac{\partial f_{n_f}}{\partial x_{s,2}} & \cdots & \frac{\partial f_{n_f}}{\partial x_{s,n_{xs}}} \end{array}\right]\Bigg|_{\left[\begin{array}{c} \mathbf{x}_d \\ \mathbf{x}'_s \\ \mathbf{x}_r \end{array}\right]} \tag{435}$$

The Jacobian (or sensitivity) matrix is the matrix of the partial derivatives of all performance features with regard to all parameters or a subset of parameters.

In row i is the partial derivative of performance feature f_i with regard to all considered parameters.

In column j is the partial derivative of all performance features with regard to parameter x_j.

Appendix C Partial derivatives of linear, quadratic terms in matrix/vector notation

	$\frac{\partial}{\partial \mathbf{x}}$	$\frac{\partial}{\partial \mathbf{x}^T}$
$\mathbf{a}^T \cdot \mathbf{x}$	$\mathbf{a}$	$\mathbf{a}^T$
$\mathbf{x}^T \cdot \mathbf{a}$	$\mathbf{a}$	$\mathbf{a}^T$
$\mathbf{x}^T_{<n>} \cdot \mathbf{A}_{<n \times n>} \cdot \mathbf{x}_{<n>}$	$\mathbf{A} \cdot \mathbf{x} + \mathbf{A}^T \cdot \mathbf{x}$ $\mathbf{A} = \mathbf{A}^T$ (symmetric): $2 \cdot \mathbf{A} \cdot \mathbf{x}$	$\mathbf{x}^T \cdot \mathbf{A}^T + \mathbf{x}^T \cdot \mathbf{A}$ $\mathbf{A} = \mathbf{A}^T$: $2 \cdot \mathbf{x}^T \cdot \mathbf{A}^T$
$\mathbf{x}^T \cdot \mathbf{x}$	$2 \cdot \mathbf{x}$	$2 \cdot \mathbf{x}^T$
$\mathbf{A}_{<m \times n>} \cdot \mathbf{x}_{<n>}$ $= \mathbf{a}_{\diamond 1} \cdot x_1 + \mathbf{a}_{\diamond 2} \cdot x_2 + \ldots + \mathbf{a}_{\diamond n} \cdot x_n$ $= \begin{bmatrix} \mathbf{a}^T_{1\diamond} \cdot \mathbf{x} \\ \mathbf{a}^T_{2\diamond} \cdot \mathbf{x} \\ \vdots \\ \mathbf{a}^T_{m\diamond} \cdot \mathbf{x} \end{bmatrix}$	$\begin{bmatrix} \mathbf{a}_{1\diamond} \\ \mathbf{a}_{2\diamond} \\ \vdots \\ \mathbf{a}_{m\diamond} \end{bmatrix}_{<mn>}$	$\mathbf{A} = \begin{bmatrix} \mathbf{a}^T_{1\diamond} \\ \mathbf{a}^T_{2\diamond} \\ \vdots \\ \mathbf{a}^T_{m\diamond} \end{bmatrix}$
$\mathbf{x}^T_{<n>} \cdot \mathbf{A}^T_{<n \times m>}$ $= \mathbf{a}^T_{\diamond 1} \cdot x_1 + \mathbf{a}^T_{\diamond 2} \cdot x_2 + \ldots + \mathbf{a}^T_{\diamond n} \cdot x_n$ $= \begin{bmatrix} \mathbf{a}^T_{1\diamond} \cdot \mathbf{x} & \mathbf{a}^T_{2\diamond} \cdot \mathbf{x} & \cdots & \mathbf{a}^T_{m\diamond} \cdot \mathbf{x} \end{bmatrix}$	$\mathbf{A}^T = \begin{bmatrix} \mathbf{a}_{1\diamond} & \mathbf{a}_{2\diamond} & \cdots & \mathbf{a}_{m\diamond} \end{bmatrix}$	$\begin{bmatrix} \mathbf{a}^T_{1\diamond} & \mathbf{a}^T_{2\diamond} & \cdots & \mathbf{a}^T_{m\diamond} \end{bmatrix}_{<mn>}$

$$\frac{\partial \begin{bmatrix} \vdots \\ \otimes \\ \vdots \end{bmatrix}_{<m>}}{\partial \begin{bmatrix} \quad \end{bmatrix}_{<n>}} = \begin{bmatrix} \vdots \\ \frac{\partial \otimes}{\partial \begin{bmatrix} \quad \end{bmatrix}_{<n>}} \\ \vdots \end{bmatrix}_{<mn>} \tag{436}$$

$$\frac{\partial \begin{bmatrix} \vdots \\ \otimes \\ \vdots \end{bmatrix}_{<m>}}{\partial [\qquad]_{<n>}} = \begin{bmatrix} \vdots \\ \frac{\partial \otimes}{\partial [\qquad]_{<n>}} \\ \vdots \end{bmatrix}_{<m \times n>} \tag{437}$$

$$\frac{\partial [\cdots \otimes \cdots]_{<m>}}{\partial \begin{bmatrix} \quad \end{bmatrix}_{<n>}} = \begin{bmatrix} \cdots & \frac{\partial \otimes}{\partial \begin{bmatrix} \quad \end{bmatrix}_{<n>}} & \cdots \end{bmatrix}_{<n \times m>} \tag{438}$$

$$\frac{\partial [\cdots \otimes \cdots]_{<m>}}{\partial [\qquad]_{<n>}} = \begin{bmatrix} \cdots & \frac{\partial \otimes}{\partial [\qquad]_{<n>}} & \cdots \end{bmatrix}_{<mn>} \tag{439}$$

Appendix D Norms

Vector norm

$$l_1\text{-norm} \quad \|\mathbf{x}\|_1 = \sum_{i=1}^{n_x} |x_i| \tag{440}$$

$$l_2\text{-norm} \quad \|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n_x} x_i^2} = \sqrt{\mathbf{x}^T \cdot \mathbf{x}} \tag{441}$$

$$l_\infty\text{-norm} \quad \|\mathbf{x}\|_\infty = \max_i |x_i| \tag{442}$$

$$l_p\text{-norm} \quad \|\mathbf{x}\|_p = \left(\sum_{i=1}^{n_x} |x_i|^p \right)^{\frac{1}{p}} \tag{443}$$

$$\tag{444}$$

Matrix norm

$$\text{max norm} \quad \|\mathbf{A}\|_M = n_x \cdot \max_{i,j} |a_{ij}| \tag{445}$$

$$\text{row norm} \quad \|\mathbf{A}\|_Z = \max_i \sum_j |a_{ij}| \tag{446}$$

$$\text{column norm} \quad \|\mathbf{A}\|_S = \max_j \sum_i |a_{ij}| \tag{447}$$

$$\text{Euclidean norm} \quad \|\mathbf{A}\|_E = \sqrt{\sum_i \sum_j |a_{ij}|^2} \tag{448}$$

$$\text{spectral norm} \quad \|\mathbf{A}\|_\lambda = \sqrt{\lambda_{\max} \left(\mathbf{A}^T \cdot \mathbf{A} \right)} \tag{449}$$

Appendix E Pseudo-inverse, singular value decomposition (SVD)

E.1 Moore-Penrose conditions

For every matrix $\mathbf{A}_{<m\times n>}$, there always exists a *unique* matrix $\mathbf{A}^{+}_{<n\times m>}$ that satisfies the following conditions.

$$\mathbf{A}\cdot\mathbf{A}^{+}\cdot\mathbf{A} = \mathbf{A}, \quad \mathbf{A}\cdot\mathbf{A}^{+} \text{ maps each column of } \mathbf{A} \text{ to itself} \tag{450}$$

$$\mathbf{A}^{+}\cdot\mathbf{A}\cdot\mathbf{A}^{+} = \mathbf{A}^{+}, \quad \mathbf{A}^{+}\cdot\mathbf{A} \text{ maps each column of } \mathbf{A}^{+} \text{ to itself} \tag{451}$$

$$\left(\mathbf{A}\cdot\mathbf{A}^{+}\right)^{T} = \mathbf{A}\cdot\mathbf{A}^{+}, \quad \mathbf{A}\cdot\mathbf{A}^{+} \text{ is symmetric} \tag{452}$$

$$\left(\mathbf{A}^{+}\cdot\mathbf{A}\right)^{T} = \mathbf{A}^{+}\cdot\mathbf{A}, \quad \mathbf{A}^{+}\cdot\mathbf{A} \text{ is symmetric} \tag{453}$$

E.2 Singular value decomposition

Every matrix $\mathbf{A}_{<m\times n>}$ with $\text{rank}(\mathbf{A}) = r \leq \min(m,n)$ has a singular value decomposition

$$\mathbf{A}_{<m\times n>} = \mathbf{V}_{<m\times m>} \cdot \hat{\mathbf{A}}_{<m\times n>} \cdot \mathbf{U}^T_{<n\times n>} \tag{454}$$

$\mathbf{V}$: matrix of the m left singular vectors (eigenvectors of $\mathbf{A}\cdot\mathbf{A}^T$)

$\mathbf{U}$: matrix of the n right singular vectors (eigenvectors of $\mathbf{A}^T\cdot\mathbf{A}$)

$\mathbf{U},\mathbf{V}$ orthogonal: $\mathbf{U}^{-1}=\mathbf{U}^T$, $\mathbf{V}^{-1}=\mathbf{V}^T$

$$\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{D}_{<r\times r>} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{<m\times n>}, \quad r = \text{rank}\,(\mathbf{A}) \tag{455}$$

$$\mathbf{D} = \text{diag}(\,d_1,\cdots,d_r) \tag{456}$$

$$\uparrow \text{ singular values}$$

$$\mathbf{A}^+_{<n\times m>} = \mathbf{U}_{<n\times n>} \cdot \hat{\mathbf{A}}^+_{<n\times m>} \cdot \mathbf{V}^T_{<m\times m>} \tag{457}$$

$$\hat{\mathbf{A}}^+ = \begin{bmatrix} \mathbf{D}^{-1}_{<r\times r>} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{<n\times m>} \tag{458}$$

$$\mathbf{D}^{-1} = \text{diag}\left(\frac{1}{d_1},\ldots,\frac{1}{d_r}\right) \tag{459}$$

singular values: positive roots of eigenvalues of $\mathbf{A}^T\cdot\mathbf{A}$

columns of $\mathbf{V}$:	basis for $\mathbb{R}^m$
columns $1,\ldots,r$ of $\mathbf{V}$:	basis for column space of $\mathbf{A}$
columns $(r+1),\ldots,m$ of $\mathbf{V}$:	basis for kernel/null-space of $\mathbf{A^T}$
columns of $\mathbf{U}$:	basis for $\mathbb{R}^n$
columns $1,\ldots,r$ of $\mathbf{U}$:	basis for row space of $\mathbf{A}$
columns $(r+1),\ldots,n$ of $\mathbf{U}$:	basis for kernel/null-space of $\mathbf{A}$

$$\hat{\mathbf{A}}\cdot\hat{\mathbf{A}}^+ = \begin{bmatrix} \mathbf{I}_{<r\times r>} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{<m\times m>}$$

$$\hat{\mathbf{A}}^+\cdot\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{I}_{<r\times r>} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{<n\times n>}$$

$$\mathbf{A}\cdot\mathbf{A}^+ = \begin{bmatrix} \mathbf{I}_{<r\times r>} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{<m\times m>}$$

$$\mathbf{A}^+\cdot\mathbf{A} = \begin{bmatrix} \mathbf{I}_{<r\times r>} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{<n\times n>}$$

Appendix F Linear equation system, rectangular system matrix with full rank

$$\begin{aligned}\mathbf{A}_{<m\times n>}\cdot\mathbf{x}_{<n>}&=\mathbf{c}_{<m>}\\ \operatorname{rank}(\mathbf{A})&=\min(m,n)\end{aligned}\tag{460}$$

F.1 $m<n$, $\operatorname{rank}(\mathbf{A})=m$, underdetermined system of equations

$$\mathbf{A}\cdot\mathbf{A}^T \quad \text{is invertible}$$

$$\mathbf{A}\cdot\mathbf{x}=\mathbf{c} \quad\rightarrow\quad \mathbf{A}\cdot\underbrace{\mathbf{A}^T\cdot\mathbf{w}'}_{\mathbf{x}}=\mathbf{c} \quad\rightarrow\quad \mathbf{w}'=(\mathbf{A}\cdot\mathbf{A}^T)^{-1}\cdot\mathbf{c}$$

$$\mathbf{x}=\underbrace{\mathbf{A}^T\cdot(\mathbf{A}\cdot\mathbf{A}^T)^{-1}}_{\mathbf{A}^+}\cdot\mathbf{c} \quad \text{minimum length solution}$$

Solving (460) **by SVD**

$$\mathbf{A}=\mathbf{V}\cdot[\mathbf{D}_{<m\times m>}\ \mathbf{0}]_{<m\times n>}\cdot\mathbf{U}^T$$

$$\mathbf{V}\cdot[\mathbf{D}\ \mathbf{0}]\cdot\underbrace{\mathbf{U}^T\cdot\mathbf{x}}_{\mathbf{w}''}=\mathbf{c}$$

$$[\mathbf{D}|\mathbf{0}]\cdot\mathbf{w}''=\mathbf{V}^T\cdot\mathbf{c} \qquad \mathbf{w}''=\begin{bmatrix}\mathbf{w}''_{a<m>}\\ \mathbf{w}''_b\end{bmatrix}_{<n>}$$

$$\mathbf{D}\cdot\mathbf{w}''_a=\mathbf{V}^T\cdot\mathbf{c} \quad\overset{\text{element-wise division}}{\rightarrow}\quad \mathbf{w}''_a$$

$$\mathbf{x}=\mathbf{U}\cdot\begin{bmatrix}\mathbf{w}''_a\\ \mathbf{0}\end{bmatrix}$$

Solving (460) **by QR-decomposition**

$$\mathbf{A}^T_{<n\times m>}=\underbrace{\left[\mathbf{Q}_{<n\times m>}\ \mathbf{Q}^{\perp}\right]_{<n\times n-m>}}_{\substack{\text{orthogonal, i.e.,}\\ \mathbf{Q}^T\cdot\mathbf{Q}^{\perp}=\mathbf{0}}}\cdot\begin{bmatrix}\mathbf{R}_{<m\times m>}\\ \mathbf{0}\end{bmatrix}_{<n-m\times m>}\tag{461}$$

$$\mathbf{A}^+=\mathbf{A}^T\cdot(\mathbf{A}\cdot\mathbf{A}^T)^{-1}=\mathbf{Q}\cdot\mathbf{R}^{-T}$$

$$\mathbf{R}^T\cdot\underbrace{\mathbf{Q}^T\cdot\mathbf{x}}_{\mathbf{w}'''}=\mathbf{c} \quad\overset{\text{forward substitution}}{\rightarrow}\quad \mathbf{w}''' \quad\rightarrow\quad \mathbf{x}=\mathbf{Q}\cdot\mathbf{w}'''$$

F.2 $m > n$, $\mathrm{rank}(\mathbf{A}) = n$, overdetermined system of equations

$$\begin{aligned} & \mathbf{A}^T \cdot \mathbf{A} \quad \text{is invertible} \\ \mathbf{A} \cdot \mathbf{x} = \mathbf{c} \quad \rightarrow \quad & \mathbf{A}^T \cdot \mathbf{A} \cdot \mathbf{x} = \mathbf{A}^T \cdot \mathbf{c} \\ \mathbf{x} = \underbrace{(\mathbf{A}^T \cdot \mathbf{A})^{-1} \cdot \mathbf{A}^T}_{\mathbf{A}^+} \mathbf{c} \quad & \text{least squares solution} \end{aligned}$$

Solving (460) **by SVD**

$$\begin{aligned} \mathbf{A} &= \mathbf{V} \cdot \begin{bmatrix} \mathbf{D}_{<n \times n>} \\ \mathbf{0} \end{bmatrix}_{<m \times n>} \cdot \mathbf{U}^T \\ \mathbf{V} \cdot \begin{bmatrix} \mathbf{D} \\ \mathbf{0} \end{bmatrix} \cdot \underbrace{\mathbf{U}^T \cdot \mathbf{x}}_{\mathbf{w}''} &= \mathbf{c} \\ \begin{bmatrix} \mathbf{D} \\ \mathbf{0} \end{bmatrix} \cdot \mathbf{w}'' = \mathbf{V}^T \cdot \mathbf{c} &= \begin{bmatrix} \mathbf{v}_{a\ <n>} \\ \mathbf{v}_b \end{bmatrix}_{<m>} \\ \mathbf{D} \cdot \mathbf{w}'' = \mathbf{v}_a \quad & \overset{\text{element-wise division}}{\rightarrow} \quad \mathbf{w}'' \\ \mathbf{x} &= \mathbf{U} \cdot \mathbf{w}'' \end{aligned}$$

Solving (460) **by QR-decomposition**

$$\begin{aligned} \mathbf{A}_{<m \times n>} = \underbrace{\left[\mathbf{Q}_{<m \times n>}\ \mathbf{Q}^{\perp}\right]_{<m \times m>}}_{\text{orthogonal}} \cdot \begin{bmatrix} \mathbf{R}_{<n \times n>} \\ \mathbf{0} \end{bmatrix}_{<m \times n>} &= \mathbf{Q} \cdot \mathbf{R} \\ \mathbf{A}^+ = (\mathbf{A}^T \cdot \mathbf{A})^{-1} \cdot \mathbf{A}^T &= \mathbf{R}^{-1} \mathbf{Q}^T \\ \mathbf{Q} \cdot \mathbf{R} \cdot \mathbf{x} = \mathbf{c} \Leftrightarrow \mathbf{R} \cdot \mathbf{x} = \mathbf{Q}^T \cdot \mathbf{c} \quad & \overset{\text{backward substitution}}{\rightarrow} \quad \mathbf{x} \end{aligned}$$

F.3 $m = n$, $\text{rank}(\mathbf{A}) = m = n$, determined system of equations

$$\mathbf{A} \text{ is invertible}$$
$$\mathbf{A}^+ = \mathbf{A}^{-1}$$

Solving (460) **by SVD**

$$\mathbf{A} = \mathbf{V} \cdot \mathbf{D} \cdot \mathbf{U}^T$$
$$\mathbf{V} \cdot \mathbf{D} \cdot \underbrace{\mathbf{U}^T \cdot \mathbf{x}}_{\mathbf{w}''} = \mathbf{c}$$
$$\mathbf{D} \cdot \mathbf{w}'' = \mathbf{V}^T \cdot \mathbf{c} \quad \overset{\text{element-wise division}}{\rightarrow} \quad \mathbf{w}''$$
$$\mathbf{x} = \mathbf{U} \cdot \mathbf{w}''$$

Solving (460) **by QR-decomposition**

$$\mathbf{A} = \mathbf{Q} \cdot \mathbf{R}$$
$$\mathbf{A}^+ = \mathbf{R}^{-1} \cdot \mathbf{Q}^T$$
$$\mathbf{Q} \cdot \mathbf{R} \cdot \mathbf{x} = \mathbf{c} \;\Leftrightarrow\; \mathbf{R} \cdot \mathbf{x} = \mathbf{Q}^T \cdot \mathbf{c} \quad \overset{\text{backward substitution}}{\rightarrow} \quad \mathbf{x}$$

Solving (460) **by LU-decomposition**

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U} \qquad \begin{aligned} \mathbf{L} &: \text{lower left triangular matrix} \\ \mathbf{U} &: \text{upper right triangular matrix} \end{aligned}$$
$$\mathbf{L} \cdot \underbrace{\mathbf{U} \cdot \mathbf{x}}_{\mathbf{w}''''} = \mathbf{c}$$
$$\mathbf{L} \cdot \mathbf{w}'''' = \mathbf{c} \quad \overset{\text{forward substitution}}{\rightarrow} \quad \mathbf{w}''''$$
$$\mathbf{U} \cdot \mathbf{x} = \mathbf{w}'''' \quad \overset{\text{backward substitution}}{\rightarrow} \quad \mathbf{x}$$

$\mathbf{A}$ orthogonal: $\mathbf{A}^T \cdot \mathbf{A} = \mathbf{A} \cdot \mathbf{A}^T = \mathbf{I}$
$\mathbf{A}^+ = \mathbf{A}^{-1} = \mathbf{A}^T$

$\mathbf{A}$ diagonal: $A_{ii}^+ = \frac{1}{A_{ii}}$
(rank deficient, i.e. $\exists_i A_{ii} = 0$, then $A_{ii}^+ = 0$)

Appendix G Probability space

event B: e.g., performance value φ is less than or equal φ_U, i.e., $B = \{\varphi | \varphi \leq \varphi_U\}$

probability P: e.g., $P(B) = P(\{\varphi \,|\, \varphi \leq \varphi_U\}) = cdf_\varphi(\varphi_U) = \int_{-\infty}^{\varphi_U} pdf_\varphi(\varphi) \cdot d\varphi$

event $B \subset \Omega$: subset of sample set of possible outcomes of probabilistic experiment Ω

elementary event: event with one element (if event sets are countable)

event set $\mathcal{B}$: system of subsets of sample set, $B \in \mathcal{B}$

definitions:

$$\Omega \in \mathcal{B} \tag{462}$$

$$B \in \mathcal{B} \;\Rightarrow\; \bar{B} \in \mathcal{B} \tag{463}$$

$$B_i \in \mathcal{B}, i \in N \;\Rightarrow\; \cup_i B_i \in \mathcal{B} \tag{464}$$

event set $\mathcal{B}$: "σ-algebra"

Kolmogorov axioms:

$$P(B) \geq 0 \text{ for all } B \in \mathcal{B} \tag{465}$$

$$P(\Omega) = 1 \text{ "}\Omega\text{: certain event"} \tag{466}$$

$$B_i \cap B_j = \{\}, B_{i,j} \in \mathcal{B} \Rightarrow P(B_i \cup B_j) = P(B_i) + P(B_j) \tag{467}$$

corollaries:

$$P(\{\}) = 0 \text{ "}\{\}\text{: impossible event"} \tag{468}$$

$$B_i \subseteq B_j \Rightarrow P(B_i) \leq P(B_j) \tag{469}$$

$$P\left(\bar{B}\right) = 1 - P(B) \tag{470}$$

$$0 \leq P(B) \leq 1 \tag{471}$$

$$\int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} pdf(\mathbf{t}) \cdot d\mathbf{t} = 1 \tag{472}$$

Appendix H Convexity

H.1 Convex set $K \in \mathbb{R}^n$

$$\underset{\mathbf{p}_0,\mathbf{p}_1 \in K, a \in [0,1]}{\forall} \mathbf{p}_a \triangleq (1-a)\cdot \mathbf{p}_0 + a \cdot \mathbf{p}_1 \in K \tag{473}$$

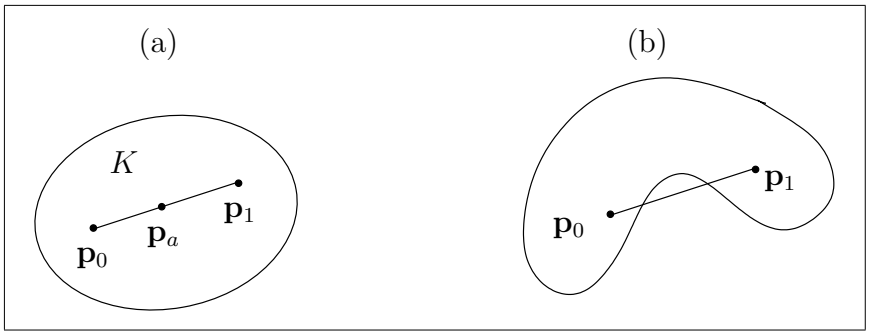


Figure 53. (a) convex set, (b) non-convex set

H.2 Convex function

$\nabla^2 f$ is positive definite

$$\underset{\substack{\mathbf{p}_0, \mathbf{p}_1 \in K, a \in [0,1] \\ \mathbf{p}_a = (1-a)\cdot \mathbf{p}_0 + a \cdot \mathbf{p}_1}}{\forall} f(\mathbf{p}_a) \leq (1-a)\cdot f(\mathbf{p}_0) + a \cdot f(\mathbf{p}_1) \tag{474}$$

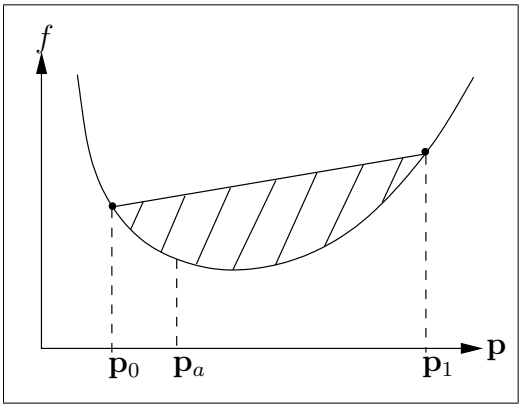


Figure 54. convex function

$c(\mathbf{p})$ is a concave function $\iff \{\mathbf{p} \mid c(\mathbf{p}) \geq \mathbf{c}_0\}$ is a convex function
convex optimization problem: local minima are global minima
strict convexity: unique global solution

Appendix I Derivatives of the statistically estimated yield

$\nabla Y,\ \nabla^2 Y$

$$Y = \underset{pdf(\underline{x}_s)}{E}\{\sigma(\underline{x}_s)\} = \int_{-\infty}^{+\infty}\cdots\int \sigma(\underline{x}_s)\cdot pdf(\underline{x}_s)\cdot d\underline{x}_s$$

$$\underset{\underline{x}_{s,0}}{\nabla} Y(\underline{x}_{s,0}) = \int_{-\infty}^{+\infty}\cdots\int \sigma(\underline{x}_s)\cdot \underset{\underline{x}_{s,0}}{\nabla} pdf(\underline{x}_{s,0})\cdot d\underline{x}_s$$

assumption $\nabla\sigma(\underline{x}_{s,0}) = 0$

$$pdf(\underline{x}_s) = k\cdot e^{-\frac{1}{2}(\underline{x}_s-\underline{x}_{s,0})^T\cdot \underset{\sim}{C}^{-1}\cdot(\underline{x}_s-\underline{x}_{s,0})}$$

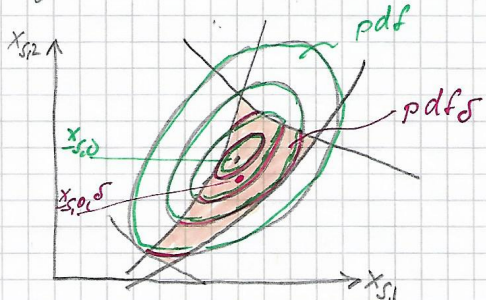
$$\underset{\underline{x}_{s,0}}{\nabla} pdf(\underline{x}_{s,0}) = pdf(\underline{x}_s)\cdot\left(-\frac{1}{2}\right)\cdot 2\,\underset{\sim}{C}^{-1}\cdot(\underline{x}_s-\underline{x}_{s,0})\cdot(-1)$$

$$= pdf(\underline{x}_s)\cdot \underset{\sim}{C}^{-1}\cdot(\underline{x}_s-\underline{x}_{s,0})$$

$$\underset{\underline{x}_{s,0}}{\nabla} Y(\underline{x}_{s,0}) = \int_{-\infty}^{+\infty}\cdots\int \underset{\sim}{C}^{-1}\cdot(\underline{x}_s-\underline{x}_{s,0})\cdot\sigma(\underline{x}_s)\cdot pdf(\underline{x}_s)\cdot d\underline{x}_s$$

$$= Y\cdot\int_{-\infty}^{+\infty}\cdots\int \underset{\sim}{C}^{-1}\cdot(\underline{x}_s-\underline{x}_{s,0})\cdot\underbrace{\frac{1}{Y}\cdot\sigma(\underline{x}_s)\cdot pdf(\underline{x}_s)}_{\text{truncated pdf } pdf_\sigma(\underline{x}_s)}\cdot d\underline{x}_s$$

$$\int_{-\infty}^{+\infty}\cdots\int pdf_\sigma(\underline{x}_s)\cdot d\underline{x}_s = \frac{1}{Y}\cdot\int_{-\infty}^{+\infty}\cdots\int \sigma(\underline{x}_s)\cdot pdf(\underline{x}_s)\cdot d\underline{x}_s = \frac{1}{Y}\cdot Y = 1$$

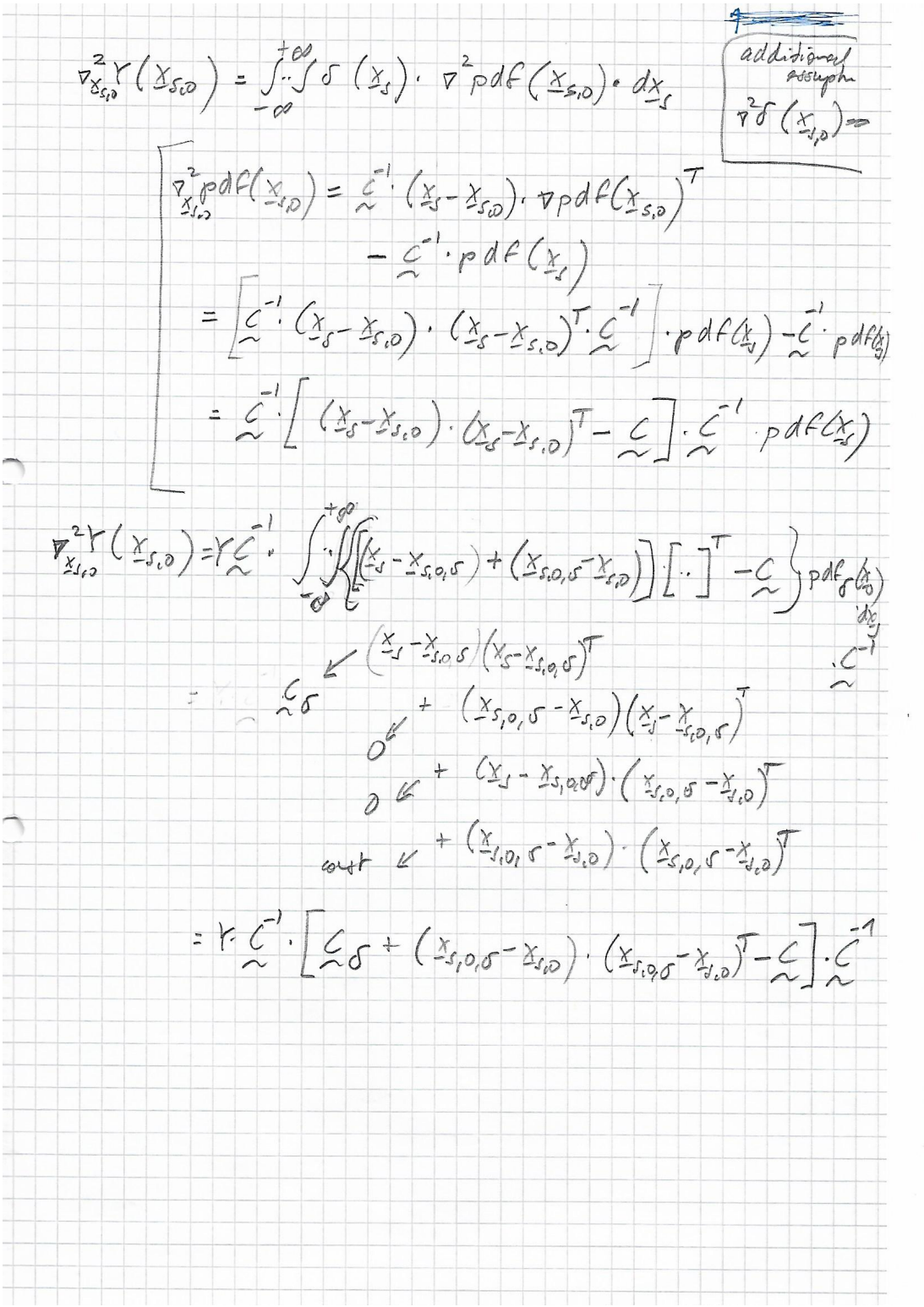


$$\underline{x}_{s,0,\sigma} = \underset{pdf_\sigma}{E}\{\underline{x}_s\} = \int_{-\infty}^{+\infty}\cdots\int \underline{x}_s\cdot pdf_\sigma(\underline{x}_s)\cdot d\underline{x}_s$$

mean value of truncated pdf

$$\underset{\underline{x}_{s,0}}{\nabla} Y(\underline{x}_{s,0}) = Y\cdot\underset{\sim}{C}^{-1}\cdot(\underline{x}_{s,0,\sigma}-\underline{x}_{s,0})$$

center of gravity — design center — stationary point $\underline{x}^*_{s,0,\sigma} = \underline{x}^*_{s,0}$



$$\nabla^2_{\underline{x}_{s,0}} \gamma(\underline{x}_{s,0}) = \int_{-\infty}^{+\infty} \cdots \int \delta(\underline{x}_s) \cdot \nabla^2 pdf(\underline{x}_{s,0}) \cdot d\underline{x}_s$$

$$\nabla^2_{\underline{x}_{s,0}} pdf(\underline{x}_{s,0}) = \underline{\underline{C}}^{-1} \cdot (\underline{x}_s - \underline{x}_{s,0}) \cdot \nabla pdf(\underline{x}_{s,0})^T - \underline{\underline{C}}^{-1} \cdot pdf(\underline{x}_s)$$

$$= \left[\underline{\underline{C}}^{-1} \cdot (\underline{x}_s - \underline{x}_{s,0}) \cdot (\underline{x}_s - \underline{x}_{s,0})^T \cdot \underline{\underline{C}}^{-1}\right] \cdot pdf(\underline{x}_s) - \underline{\underline{C}}^{-1} \cdot pdf(\underline{x}_s)$$

$$= \underline{\underline{C}}^{-1} \cdot \left[(\underline{x}_s - \underline{x}_{s,0}) \cdot (\underline{x}_s - \underline{x}_{s,0})^T - \underline{\underline{C}}\right] \cdot \underline{\underline{C}}^{-1} \cdot pdf(\underline{x}_s)$$

$$\nabla^2_{\underline{x}_{s,0}} \gamma(\underline{x}_{s,0}) = \gamma \underline{\underline{C}}^{-1} \cdot \int_{-\infty}^{+\infty} \cdots \int \left\{ \left[(\underline{x}_s - \underline{x}_{s,0,\delta}) + (\underline{x}_{s,0,\delta} - \underline{x}_{s,0}) \right] \cdot [\cdot]^T - \underline{\underline{C}} \right\} pdf_\delta(\underline{x}_s) \cdot d\underline{x}_s \cdot \underline{\underline{C}}^{-1}$$

$$= \gamma \cdot \underline{\underline{C}}^{-1} \cdot \left[\underline{\underline{C}}_\delta + (\underline{x}_{s,0,\delta} - \underline{x}_{s,0}) \cdot (\underline{x}_{s,0,\delta} - \underline{x}_{s,0})^T - \underline{\underline{C}}\right] \cdot \underline{\underline{C}}^{-1}$$