Optimal Design of Electronic Components by Mixed-Integer Nonlinear Programming

G. van de Braak[1], M. J. Bünner[2], K. Schittkowski[3]

Abstract

Computer-aided design optimization of electronic components is a powerful tool to reduce development costs on one hand and to improve the performance of the components on the other. In this paper, a mathematical model of an electronic filter is outlined. It depends on certain parameters, some of them of being continuous, others of integer type. The purpose of the paper is to introduce an extension of the well-known sequential quadratic programming (SQP) method to solve the mixed-integer programming problem (MINLP). It is assumed that the integer variables cannot be relaxed to real ones, that the integer range is sufficiently large, and that they possess some physical meaning so that they basically behave like continuous ones. The general idea is to combine an SQP step with a direct search cycle in the integer space. Hessian information is updated based on difference formulae at neighbored grid points. Numerical results are included to show the feasibility of the mixed-integer nonlinear programming code for academic test examples and in addition for the optimal design of an electronic filter.

Keywords: electronic components; mixed-integer nonlinear programming; discrete optimization; categorial variables; sequential quadratic programming; direct search method

[1]Department of Mathematics, University of Münster, 48149 Münster, Germany
[2]EPCOS AG, 81617 München, Germany
[3]Department of Mathematics, University of Bayreuth, 95440 Bayreuth, Germany

1 Introduction

A bandpass filter selects a band of frequencies out of the electro-magnetic spectrum. In this paper, we consider surface-acoustic-wave (SAW) filters consisting of a piezo-electric substrate, where the surface is covered by metal structures [19]. The incoming electrical signal is converted to a mechanical signal by this setup. The SAW filter acts as a transducer of electrical energy to mechanical energy and vice versa. The efficiency of the conversion depends strongly on the frequencies of the incoming signals and the geometry parameters of the metal structures, for example length, height, etc. On this basis, the characteristic properties of a filter are achieved.

Surface acoustic waves (SAWs) have been described for the first time by Lord Rayleigh in 1885 in combination with the scientific description of earthquakes. SAWs were technically used for the first time in the 1960s for military purposes. Later, SAW-bandpass filters entered the market for TV applications. Due to small physical sizes and unique electrical properties, SAW-bandpass filters raised tremendous interest in mobile phone applications. The large demand of the mobile phone industry is covered by large-scale, industrial mass-production of SAW-filters.

For industrial applications, bandpass filters are designed in order to satisfy predefined electrical specifications. The *art of filter design* consists of defining the internal structure, or the geometry parameters, respectively, of a filter such that the specifications are satisfied. Bandpass filters are designed with the help of computer-aided-engineering (CAE) techniques. The electrical properties of the filters are simulated based on physical models. The simulation of a bandpass filter consists of

1. the acoustic tracks, i.e., the areas on the piezo-electrical substrate on which the electrical energy is converted to mechanical vibrations and vice versa,

2. the electrical combinations of the different acoustic tracks.

Typically, only the properties of the acoustic tracks are varied during the design process, and are defined by several physical parameters. Some of them are given in form of real numbers, some others in form of integer numbers. As soon as the filters properties fit to the demands, the mass production of the filter is started.

To describe an acoustic track mathematically, Tobolka [39] introduced the *P-matrix model* in 1979. Different sections of the acoustic track are described by different P-matrices, which reflect the physics of that section. The P-matrices depend on the geometry parameters of the acoustic tracks. Based on these P-Matrices, the scattering matrix, that describes the electrical properties of the band-pass filter, can be derived.

Once the scattering matrix is computed, the transmission of a filter is obtained as a function of frequency. Geometry parameters of the filter are considered as independent variables to be optimized. Objective function to be maximized is the

minimum of the transmission within a given range, and constraints are lower transmission bounds in some outer frequency ranges. The continuous max-min optimization problem is transformed into a smooth, nonlinear optimization problem, and can be solved by an available nonlinear programming algorithm, for example by the code NLPQL of Schittkowski [35].

Because of the highly complex computations needed for establishing the scattering matrix, the number of iterations must be as low as possible. Since the applied numerical methods require first derivatives of nonlinear model functions, it is necessary to approximate them numerically by forward differences, an additional burden for the numerical complexity of the described approach. Sequential quadratic programming (SQP) methods, for example NLPQL, are well established algorithms for problems of the type considered, see for example Hartwanger et al. [15] for a quite similar application, where the geometric parameters of corrugated antenna horns are optimized. Comparative performance evaluations for more general classes of optimization problems, are found in Schittkowski [32] and Hock, Schittkowski [16].

However, the situation becomes much more complex if additional integer variables must be taken into account, for example the number of metallized layers of a filter. Instead of a trial-and-error approach, we propose a more generally applicable procedure for the simultaneous adoption of all parameters, the discrete and the continuous ones. There are numerous proposals on how to solve mixed-integer nonlinear programs, see for example Floudas [9] or Grossmann and Kravanja [12] for reviews. Typically, all these approaches require convex model functions, a limited range of the integer variables, and/or continuous relaxations of integer variables. By a continuous relaxation, we understand that integer variables can be treated as continuous variables, i.e., function values can also be computed in a small neighborhood of an integer value, for example to approximate gradients numerically subject to an integer term.

But all these conditions are violated in the present situation, in which the model functions are highly nonlinear and non-convex. We have relatively large ranges for integer values, and it is impossible to get sufficiently accurate approximations of derivatives of integer variables. Our proposal is based on the idea to compute successively optimal solutions with respect to the continuous variables under the assumption that all integer variables are fixed. For simplicity, we suppose that a global solution is always obtained. Starting from a given set of integer variables, objective function values at neighboring grid points are evaluated, again based on a complete optimization cycle with respect to the continuous variables. Then we formulate a quadratic approximation in the integer space by a second order difference formula, compute a minimal solution by a continuous quadratic programming code, round the values obtained, and restart the procedure.

It is possible that the quadratic approximation in the integer space does not yield a positive definite matrix or that the solution of the quadratic program is not different from the actual iterate in integer space, or too close leading to an infinite loop. In these cases, the quadratic approximation does not lead to further

improvement, and a direct search method is locally applied along the axes of the integer variables with the goal to find an improved approximation.

We assume that the integer variables possess some physical meaning and are *continuous* in some sense. A slight alteration of an integer value, say by one, changes the model functions only slightly, at least much less that a more drastic change. A typical practical situation is considered in the paper, the number of fingers or layers of an electrical filter. By increasing or decreasing the number of fingers by one, we expect only a small alteration in the total outcome, the transmission energy. The more drastically the variation of the integer variable is, the more drastically model function values are supposed to change. We call these variables *physical* ones to distinguish them from the so-called *categorial* variables, which are introduced to enumerate certain situations and where any change leads to a completely different category and thus to a completely different response.

The procedure is motivated by the observation that in case of optimizing electronic filters, the objective function seems to possess a convex structure with respect to the integer variables, if optimal values for the continuous variables are inserted. A particular advantage is that an available optimization program for the continuous formulation can be used in form of a black box.

Numerous difficulties are observed in practice, such as a nonlinear programming problem with respect to the continuous variables becomes infeasible, the continuous solution is only a local one, there is no suitable criterion for a local solution of the mixed-integer problem, the direct search stops at a non-optimal point, there is no progress along the boundaries of the box constraints, and many others. Moreover, the whole process is extremely time-consuming and sometimes unstable. Nevertheless, the resulting computer code is able to find acceptable designs of electronic filters.

Section 2 contains a brief outline of the underlying physical model equations, i.e., the wave equation, the cascading of P-matrices, and the scattering matrix from which the transmission is computed. The optimization problem is formulated in Section 3, where we describe the design problem in more detail. In Section 4, the most important features of sequential quadratic programming method are summarized to understand the computational procedure. A general procedure for solving constrained nonlinear mixed-integer programs is presented in Section 5, where the above side conditions are the main motivation for the approach proposed. To illustrate the computational performance of different variants, some computational test results are summarized in Section 6. They are obtained for an academic test problem. Finally numerical results of a real-life case study are shown in Section 7, where we consider the optimal design of an existing electronic filter.

2 Physical Model Equations

The design of SAW filters is supported by computer simulation based on a physical model. The input data of a simulation run are the parameters that define the geometry of the acoustic tracks. The output of the simulation characterizes the electrical properties of the filter computed for a series of pre-defined frequency points. In Figure 1, we show the transmission of a filter for a typical design.

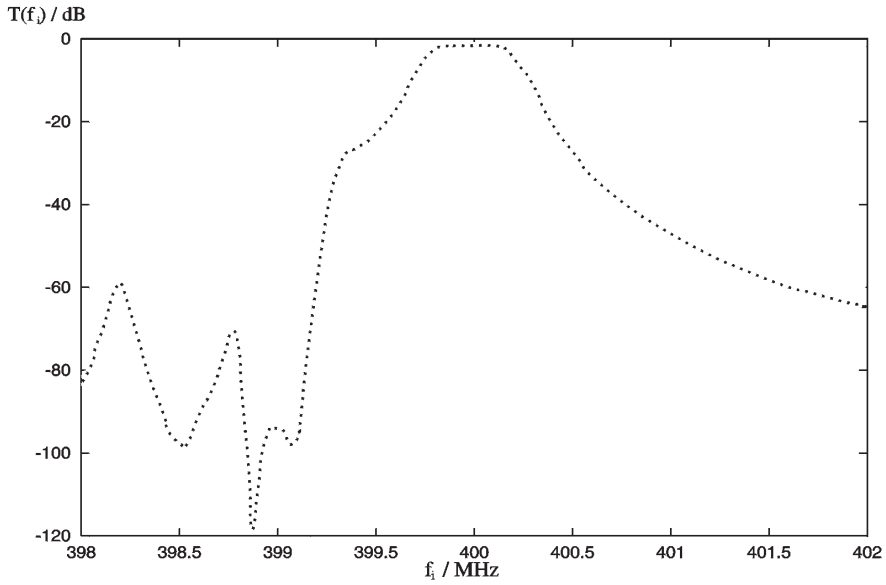

Figure 1: Transmission coefficient $\mathrm{T}(f_i)$ over frequency points f_i

As a basis of the simulation, several physical models differing in complexity and accuracy are available. Typically, the increase in accuracy has to be paid by a more complex model and, consequently, by higher computing times. Thus, the required accuracy is proportional to computing time.

In the following, we consider a SAW filter in more detail. When observing the surface of a single-crystal, we see that any deviation of an ion from its equilibrium position provokes a restoring force and an electrical field due to the piezo-electric effect. Describing the deviations of ions at the surface in terms of a scalar potential, we conclude that the SAW is described by a scalar wave equation

$$\phi_{tt} = c^2 \Delta \phi \quad . \tag{1}$$

We use the notation

- Δ - two-dimensional Laplace operator,
- c - velocity,
- t - time.

The boundary conditions are given by the physical conditions at the surface. In this case, the boundary conditions are non-trivial, since the surface is partly covered by a metal layer. In addition, the piezo-electric crystal is non-isotropic, and the velocity of the wave depends on its direction. For the numerical simulation, additional effects such as polarization charges in the metal layers have to be taken into account. Consequently, the fundamental wave equation is not solvable in a closed form. For this reason Tobolka [39] introduced the P-matrix model as an equivalent mathematical description of the SAW.

To show the complexity of the P-matrix model, we describe an example in detail. The basic element of the model is a simple base cell, which consists of two acoustic ports, and an additional electric port. The acoustic ports describe the incoming and outgoing acoustic signals, the electrical ports the electric voltage at this cell, see Figure 2. The quantities a_1, a_2, b_1 and b_2 denote the intensities of the acoustic waves. In terms of a description based on the wave equation, we have $a_1 \propto \phi$, u is the electrical voltage at the base cell, and i is the electrical current.

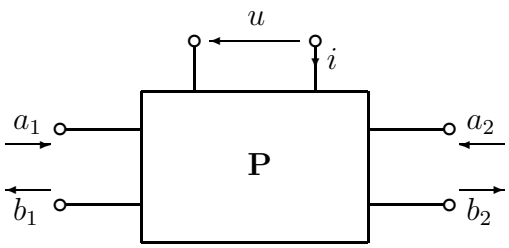

Figure 2: Base cell of the P-matrix model with two acoustic and one electric port

The P-matrix model describes the interaction of the acoustic waves at the acoustic ports, with the electric port in a linear form,

$$\begin{pmatrix} b_1 \\ b_2 \\ i \end{pmatrix} = \mathbf{P} \begin{pmatrix} a_1 \\ a_2 \\ u \end{pmatrix} = \begin{pmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ u \end{pmatrix} \quad . \tag{2}$$

The interaction between the incoming and outgoing acoustic waves a_1, and a_2, or b_1, and b_2 with i, and u at the electrical port, are given by the elements P_{13}, and P_{23}, P_{31}, and P_{32}. The elements P_{11}, P_{22}, and P_{12}, P_{21} are the dimensionless acoustic reflection and transmission coefficients in the case of a short-circuited electrical port. The submatrix

$$\begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} \tag{3}$$

is therefore the scattering matrix of the acoustic waves and describes the interaction of the incoming and outgoing waves. The submatrices $(P_{31} \;\; P_{32})$, and $(P_{13} \;\; P_{23})^T$ of $\mathbf{P}$ characterize the relation of the acoustic waves with the electric voltage, and therefore the piezo-electric effect of the substrate. P_{33} is the admittance of the base cell. The admittance is the quotient of current to voltage and the reciprocal value of the impedance.

Proceeding from the P-matrix model, we calculate the scattering matrix S. This matrix is the basic physical unit that describes the electro-acoustic properties of the acoustic tracks, and finally the filter itself. The transmission coefficient T is one element of the scattering matrix, i.e., $T = S_{21}$.

3 Optimal Design of Electronic SAW Filters

Mobile phone manufacturers provide strict specifications towards the design of a bandpass filter. Typically, the transmission has to be above certain bounds in the pass band and below certain bounds in the stop band depending on the actual frequency. These specifications have to be achieved by designing the filter in a proper way. Depending on the exact requirements upon the filter to be designed, different optimization problems can be derived.

To give an example, consider the design goals of a SAW filter as shown in Figure 3. In this case, we want to maximize the transmission of the filter in the pass band R_0. There is one lower transmission bound left from the bandpass interval in the frequency interval R_1, and another one in the frequency interval R_2. The corresponding bounds are denoted by T_0, T_1, and T_2.

To formulate the optimization problem, let us assume that $x \in \mathbb{R}^n$ denotes the vector of continuous real design variables and $y \in \mathbb{Z}^m$ the vector of the integer design variables as discussed above. $\mathbb{Z}$ is the set of all integer values. By $T(f, x, y)$ we denote the transmission subject to frequency f and the geometry parameters x and y. Some disjoint intervals R_0, ..., R_s define the design space within the frequency interval $f_l \leq f \leq f_u$. Our goal is to maximize the minimal distance of transmission $T(f, x, y)$ over the interval R_0, under lower bounds T_1, ..., T_s for the transmission in the remaining intervals R_1, ..., R_s. Moreover, it is required that the transmission is always above a certain bound in R_0, i.e., that $T(f, x, y) \geq T_0$ for all $f \in R_0$. The optimization problem is formulated as

$$\begin{aligned} & \max \; \min \{T(f, x, y) : f \in R_0\} \\ x \in \mathbb{R}^n, y \in \mathbb{Z}^m : \;\; & T(f, x, y) \leq T_i \;\; \text{for } f \in R_i \;,\; i = 1, \ldots, s \;\;, \\ & \underline{x} \leq x \leq \overline{x} \;,\;\; \underline{y} \leq y \leq \overline{y} \;\; . \end{aligned} \tag{4}$$

Here $\underline{x}, \overline{x} \in \mathbb{R}^n$ and $\underline{y}, \overline{y} \in \mathbb{Z}^m$ are lower and upper bounds for the design variables.

To transform the infinite dimensional optimization problem into a finite dimensional one, we proceed from a given discretization of the frequency variable f by

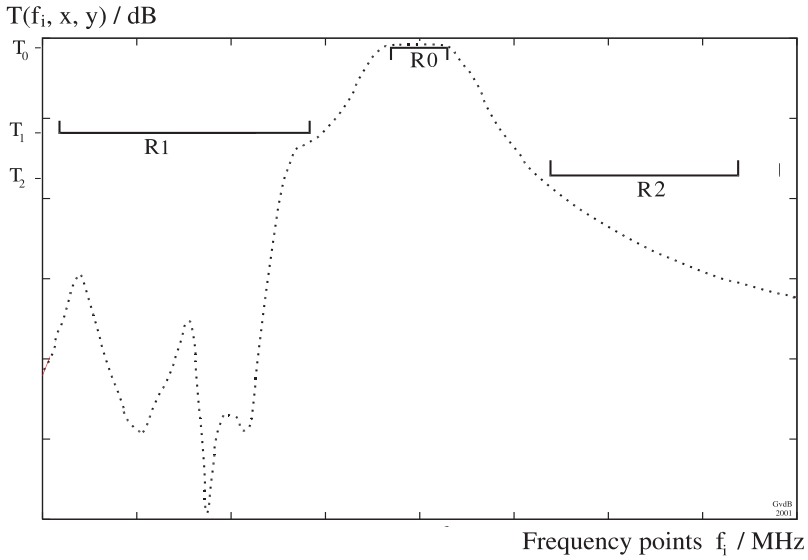

Figure 3: Design goals of a SAW filter

an equidistant grid in each interval. The corresponding index sets are called J_0, J_1, ..., J_s. Let l be the total number of all grid points. First we introduce the notation $T_j(x,y) = T(f_j,x,y)$, f_j suitable grid point, $j = 1, \ldots, l$. All indices are ordered sequentially so that $\{1,\ldots,l\} = J_0 \cup J_1 \cup \ldots \cup J_s$, i.e., $J_0 = \{1,\ldots,l_0\}$, $J_1 = \{l_0+1,\ldots,l_1\}$, ..., $J_s = \{l_{s-1}+1,\ldots,l\}$. Then the discretized optimization problem is

$$\begin{array}{ll} & \max\ \min\ \{T_j(x,y) : j \in J_0\} \\ x \in \mathbb{R}^n, y \in \mathbb{Z}^m : & T_j(x,y) \le T_i \ \text{ for } j \in J_i\ ,\ i = 1,\ldots,s\ \ , \\ & \underline{x} \le x \le \overline{x}\ ,\ \ \underline{y} \le y \le \overline{y}\ \ . \end{array} \tag{5}$$

The existence of a feasible design is easily checked by performing the test $T_j(x,y) \ge T_0$ for all $j \in J_0$. Problem (5) is equivalent to a smooth nonlinear program after a simple standard transformation.

4 Sequential Quadratic Programming Methods

We consider the continuous optimization problem with nonlinear inequality constraints

$$x \in \mathbb{R}^n : \begin{array}{l} \min\ f(x) \\ g(x) \le 0\ \ , \end{array} \tag{6}$$

where x is an n-dimensional parameter vector, and the vector-valued function $g(x)$ defines c inequality constraints, $g(x) = (g_1(x), \ldots, g_c(x))^T$. The model functions $f(x)$, $g_1(x)$, ..., $g_c(x)$ are continuously differentiable.

Sequential quadratic programming (SQP) methods are well known and are considered as the standard general purpose algorithms for solving smooth nonlinear optimization problems. They have their roots in unconstrained optimization, and can be considered as extensions of quasi-Newton methods. The basic idea is to establish a quadratic approximation of the Lagrangian function based on second order information, with the goal to achieve a fast local convergence speed. They belong to the most powerful nonlinear programming algorithms we know today for solving differentiable nonlinear programming problems of the form (6). The theoretical background is described for example in Stoer [38] in form of a review, or in Spellucci [37] in form of an extensive text book, see also Gill et. al [10]. From the more practical point of view, SQP methods are also introduced in the books of Papalambros and Wilde [28] or Edgar and Himmelblau [7]. Their excellent numerical performances are shown in comparison with other methods in Schittkowski [32], [33], and Hock and Schittkowski [16]. For many years, they belong to the most frequently used algorithms to solve practical optimization problems.

SQP solves a quadratic programming subproblem in each iteration step, which is obtained by linearizing the constraints and approximating the Lagrangian function

$$L(x,u) = f(x) + u^T g(x) \tag{7}$$

quadratically. To formulate the quadratic programming subproblem, we proceed from the given iterate $x_k \in \mathbb{R}^n$, an approximation of the solution, $v_k \in \mathbb{R}^c$, an approximation of the multipliers, and $B_k \in \mathbb{R}^{n\times n}$, an approximation of the Hessian of the Lagrangian function in a certain sense. Then one has to solve the quadratic programming subproblem

$$d \in \mathbb{R}^n : \begin{array}{l} \min \ \frac{1}{2} d^T B_k d + \nabla f(x_k)^T d \\ \nabla g(x_k)^T d + g(x_k) \le 0 \ . \end{array} \tag{8}$$

Here $\nabla g(x_k)$ denotes the Jacobian matrix of $g(x)$ at $x = x_k$. Let d_k be the optimal solution and u_k the corresponding multiplier. A new iterate is obtained by

$$\begin{pmatrix} x_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ v_k \end{pmatrix} + \alpha_k \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix} \ , \tag{9}$$

where $\alpha_k \in (0, 1]$ is a suitable steplength parameter.

The steplength parameter α_k is required in (9) to enforce global convergence of the SQP method, i.e., the convergence to a point satisfying the necessary Karush-Kuhn-Tucker optimality conditions when starting from arbitrary initial values, e.g.,

a user-provided $x_0 \in \mathbb{R}^n$ and $v_0 = 0$, $B_0 = I$. α_k should satisfy at least a sufficient decrease of a merit function $\phi_r(\alpha)$ given by

$$\phi_r(\alpha) = \psi_r(x + \alpha d, v + \alpha(u - v)) \tag{10}$$

with a suitable penalty function $\psi_r(x, v)$, for example the augmented Lagrangian function

$$\psi_r(x, v) = f(x) + \sum_{j \in J} \left(v_j g_j(x) + \frac{1}{2} r_j g_j(x)^2 \right) - \frac{1}{2} \sum_{j \in K} v_j^2 / r_j \tag{11}$$

with $J = \{j \leq c : g_j(x) \geq -v_j/r_j\}$ and $K = \{1, \ldots, c\} \setminus J$, see Schittkowski [34]. The objective function is *penalized* as soon as an iterate leaves the feasible domain. The corresponding penalty parameters r_j, $j = 1, \ldots, c$, that control the degree of constraint violation, must be chosen in a suitable way to guarantee a descent direction of the merit function. Then we get the following result that is essential to prove convergence,

$$\phi'_{r_k}(0) = \nabla \psi_{r_k}(x_k, v_k)^T \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix} < 0 \quad . \tag{12}$$

For the proof see Han [14] or Schittkowski [34]. Additional modifications of (8) are available to guarantee, that in case of a non-feasible domain a descent direction can be obtained, see Schittkowski [34].

The update of matrix B_k can also be performed by standard quasi-Newton techniques known from unconstrained optimization. In most cases, the BFGS method is applied, see Powell [29], [30], or Stoer [38]. Starting from the identity or any other positive definite matrix B_0, the difference vectors

$$\begin{aligned} q_k &= \nabla_x L(x_{k+1}, u_k) - \nabla_x L(x_k, u_k) \quad , \\ p_k &= x_{k+1} - x_k \end{aligned} \tag{13}$$

are used to update B_k in the form

$$B_{k+1} = \Pi(B_k, q_k, p_k) \quad , \tag{14}$$

where

$$\Pi(B, q, p) = B + \frac{qq^T}{q^T p} - \frac{Bpp^T B}{p^T B p} \quad . \tag{15}$$

The above formula yields a positive definite matrix B_{k+1} provided that B_k is positive definite and $q_k^T p_k > 0$. A simple modification of Powell [29] guarantees that B_{k+1} is positive definite even if the latter condition is violated.

Among the most attractive features of sequential quadratic programming methods is the superlinear convergence rate in the neighborhood of a solution, see Powell [30]. Under a couple of additional assumptions, we get

$$\|x_{k+1} - x^\star\| < \gamma_k \|x_k - x^\star\| \tag{16}$$

with $\gamma_k \to 0$. The motivation for the fast convergence rate of SQP methods is based on the following observation: An SQP method is identical to Newton's method to solve the necessary optimality conditions, if B_k is the Hessian of the Lagrangian function at x_k and u_k and if we start sufficiently close to a solution. The statement is easily derived in case of equality constraints only, but holds also for inequality restrictions by applying for example the Fischer-Burmeister function, see Fischer [8].

5 A Combined SQP and Direct Search Method for Solving Mixed-Integer Nonlinear Programs

Our goal is to extend the methodology behind SQP methods to solve mixed-integer nonlinear programming (MINLP) problems of the kind

$$\begin{array}{rl} & \min\ f(x,y) \\ x \in \mathbb{R}^n, y \in \mathbb{Z}^m: & g(x,y) \le 0\ , \\ & \underline{x} \le x \le \overline{x}\ ,\ \ \underline{y} \le y \le \overline{y}\ \ , \end{array} \tag{17}$$

where upper and lower bounds of the variables are separately handled. The electrical filter design problem (5) is of the same form after some modifications. A typical integer range consists of 50 to 100 different values for a scalar integer variable, or, in other words, $50 \le \overline{y}^i - \underline{y}^i \le 100$, where $\overline{y}^i$ and $\underline{y}^i$ denote the i-th upper and lower bound for the i-th integer variable, $i = 1, \ldots, m$. As in this practical case, we assume that there are no inequality constraints depending only on the integer variables.

There are very many different possibilities to take integer variables into account. Rigorous mathematical approaches are available in situations, when the integer variables can be relaxed. In this case, function values $f(x,y)$ and $g(x,y)$ can be obtained also for any $y \in \mathbb{R}^m$ and it is possible to solve the relaxed problem

$$\begin{array}{rl} & \min\ f(x,y) \\ x \in \mathbb{R}^n, y \in \mathbb{R}^m: & g(x,y) \le 0\ , \\ & \underline{x} \le x \le \overline{x}\ ,\ \ \underline{y} \le y \le \overline{y}\ \ . \end{array} \tag{18}$$

After truncating the relaxed variables, it is sometimes possible to get an approximate solution of the MINLP. Derivatives of $f(x,y)$ and $g(x,y)$ subject to y at integer

values are available and allow the development of more advanced mathematical techniques developed in the last years, for example the so-called outer approximation method of Duran and Grossmann [6], see also Floudas [9]. Successive linear cuts at previous iterates are constructed and added to a linear mixed-integer program (MILP), that can be solved by any standard algorithm. These methods are particularly used in the design of chemical reactors, see Kocis and Grossmann [20] or again Floudas [9].

The approach differs from the successive linear programming technique which is frequently applied in mechanical structural optimization, see for example Olsen and Vanderplaats [27] or Loh and Papalambros [25], where all problem functions are linearized subject to the continuous and integer variables at the current iterate. The resulting MILP is again solved by any available standard code. Successful extensions to mixed-integer SQP methods are also available, see Cha and Mayne [3, 4], where a mixed-integer quadratic program (MIQP) must be solved in each step, again requiring relaxable variables. In addition, there are numerous other methods for example based on classical branch-and-bound strategy, see Dakin [5], Beale [2], or Gupta and Ravindran [13]. Also any combinations are available, for example for SQP and branch-and-bound, see Leyffer [23]. Reviews of these algorithms and more references are found in Floudas [9], Grossmann and Kravanja [12], and Leyffer [22]. But in most of these situations, convergence analysis and reliable, efficient implementations require convex nonlinear programs.

In some other applications, it is tried to relax the integer variables and to add constraints of the form

$$(y^i - d_1^i)(y^i - d_2^i) \dots (y_i - d_{m_i}^i) = 0$$

to the resulting nonlinear program, where $i = 1, \dots, m$ and where $d_1^i, \dots, d_{m_i}^i$ are all discrete values the variable y_i may accept, see for example Li and Chou [24] or Wang et al. [42]. However, the highly non-convex artificial constraints, even if relaxed, increase the probability that only a local solution is obtained, and require global optimization techniques. Having the complex design optimization problem in mind, we are forced to reduce the number of simulations as much as possible making this strategy unacceptable.

If the integer variables cannot be relaxed, then in two situations. First, there are so-called *categorial* variables, see for example Audet and Dennis [1], which define certain alternatives like different types of material. Typically, these variables enumerate situations which must be handled separately in a simulation code, and there is no way to distinguish between a *close* and a *far* value. Since model function values may differ completely when moving from one integer value to the next neighbor, it is not possible to approximate the variable in an adequate form. Under this condition, search methods are applicable, for example pattern search, evolutionary algorithms, multidirectional search, or stochastic search, see for example the books of Kelley [18], Schwefel [36], or Miettinen et al. [26].

In our case, we assume that the integer variables possess some physical meaning and are *continuous* in some sense. A slight alteration of an integer value, say by one, changes the model functions only slightly, at least much less than a more drastic change. A typical practical situation is considered in the paper, the number of fingers or layers of an electronic filter. By increasing or decreasing the number of fingers by one, we expect only a small alteration in the total outcome, the transmission energy. The more drastically the variation of the integer variable is, the more drastically model function values are supposed to change. We call these variables *physical* ones to distinguish them from the categorial variables mentioned above.

To summarize, we have the following situation:

1. The model functions are highly nonlinear and non-convex.
2. There are relatively large ranges for integer values.
3. The feasible domain can become empty.
4. There is no continuous relaxation.
5. The integer variables have a physical meaning.

Our proposal is based on the idea to be able to compute an optimal solution with respect to the continuous variables under the assumption that all integer variables are fixed. If $x(y)$ denotes the solution of

$$x \in \mathbb{R}^n : \begin{array}{l} \min\ f(x,y) \\ g(x,y) \leq 0 \\ \underline{x} \leq x \leq \overline{x} \end{array} , \tag{19}$$

for a fixed set of integer variables $y \in \mathbb{Z}^m$ with $\underline{y} \leq y \leq \overline{y}$, we get the nonlinear bound-constrained integer problem

$$y \in \mathbb{Z}^m : \begin{array}{l} \min\ F(y) \\ \underline{y} \leq y \leq \overline{y} \end{array} , \tag{20}$$

where

$$F(y) = f(x(y), y) \ . \tag{21}$$

There are no additional inequality constraints, since we assume that the nonlinear constraints in (17) do not depend on the integer variables only. When solving (19) numerically, for example by an SQP method as outlined in the previous section, we hope that a global solution of (19) is computed, in particular that (19) possesses a feasible solution. Otherwise, some additional safeguards are performed to proceed to another iterate in the integer space.

The proposed algorithm consists of a successive quadratic approximation combined with a direct search method. Starting from a given set of integer variables y_0, objective function values of $F(y)$ are evaluated at neighboring grid points, where a complete optimization cycle with respect to the continuous variables must be performed. Then we formulate a continuous quadratic approximation in the integer space by a second order difference formula with unit stepsizes along the axes, to get a matrix B_k and a vector h_k. In case of relaxable integer variables, we would interpret B_k as an approximation of $\nabla^2 F(y_k)$ and h_k as an approximation of $\nabla F(y_k)$ at the k-th iterate y_k. Then the solution $d_k \in \mathbb{R}^m$ of the continuous quadratic programming (QP) problem

$$d \in \mathbb{R}^m : \quad \begin{array}{l} \min \ \frac{1}{2} d^T B_k d + h_k^T d \\ \underline{y} - y_k \leq d \leq \overline{y} - y_k \end{array} \tag{22}$$

is computed. The guess $y_k + d_k$ is rounded to get the subsequent iterate $y_{k+1} = y_k + [d_k]$, and the procedure is restarted.

The algorithm is motivated by the observation that the objective function $F(y)$ seems to possess a convex structure with respect to the integer variables y in case of the optimal design of electronic filters. A particular advantage of this approach is that an available optimization program for the continuous formulation (19) can be used in form of a black box. The continuous programs under consideration are highly non-convex, badly scaled, and ill conditioned.

It is possible that the quadratic approximation does not yield a positive definite matrix, or that the solution of the quadratic program is not different from the actual iterate or too close. If a strict descent property is not observed, we have to expect an infinite loop. In these cases, a direct search method is applied along the axes of the integer variables with the goal to find a better solution value. For a historical, see for example Lewis, Torczon, and Trosset [21]. Convergence results are found in Torczon [40] or Hough, Kolda, and Torczon [17].

Algorithm 5.1 *Let $y_1 = (y_1^1, \ldots, y_1^m)^T \in \mathbb{Z}^m$ be given where we want to start the direct search (DS) algorithm, and $d_0 = (1, \ldots, 1)^T$ a first guess for the search direction. Moreover, we define by $e_i \in \mathbb{Z}^m$ the i-th unit vector, $i = 1, \ldots, m$, and the maximum number of iterations is j_{max}. Then we proceed as follows for $j = 1, 2, 3, \ldots$:*

1. *Try to find an intermediate descent direction proceeding from $d_{j-1} = (d_{j-1}^1, \ldots, d_{j-1}^m)^T$:*
   For $i = 1, \ldots, m$ do
   If $d_{j-1}^i = 1$ or $d_{j-1}^i = 0$ then
   If $F(y_j + e_i) < F(y_j)$ then $s_j^i = 1$ else $s_j^i = 0$
   Else
   If $F(y_j - e_i) < F(y_j)$ then $s_j^i = -1$ else $s_j^i = 0$.

2. *Move to a better intermediate point:*
   $z_j = y_j + s_j,\ s_j = (s_j^1, \ldots, s_j^m)^T$.

3. *Now try the opposite direction for coordinates without success:*
   For $i = 1, \ldots, m$ *do*
   If $s_j^i = 0$ *then*
   If $d_{j-1}^i = 1$ *or* $d_{j-1}^i = 0$ *then*
   If $F(z_j - e_i) < F(z_j)$ *then* $d_j^i = -1$ *else* $d_j^i = 0$
   Else
   If $F(z_j + e_i) < F(z_j)$ *then* $d_j^i = 1$ *else* $d_j^i = 0$
   Else
   $d_j^i = d_{j-1}^i$.

4. *Project* d_j *onto the feasible domain:*
   For $i = 1, \ldots, m$ *do*
   If $z_j^i + d_j^i < \underline{y}^i$ *then* $d_j^i = 0$,
   If $z_j^i + d_j^i > \overline{y}^i$ *then* $d_j^i = 0$.

5. *Check stopping condition:*
   If $d_j = 0$ *or* $j = j_{max}$ *then stop.*

6. *Perform search along direction* d_k*:*
   Let $\alpha = 1$.
   While $\underline{y} \leq y_j + \alpha d_j \leq \overline{y}$ *and* $F(y_j + \alpha d_j) < F(y_j + (\alpha - 1)d_j)$ *do* $\alpha = \alpha + 1$.
   Let α_j *be the final* α *obtained.*

7. *New iterate* y_{j+1}*:*
   Let $y_{j+1} = y_j + \alpha_j d_j$.

The basic idea of this algorithm is to find a new descent direction based on a given one. Afterwards, a successive line search will be performed. First, the given search direction is analyzed by varying integer coefficients coordinate by coordinate, i.e., by investigating surrounding grid points. If the previous search direction is profitable, this old direction is retained. The auxiliary variable s_j is used for defining an intermediate trial point z_j based on the coordinates of the old search direction that are still profitable at y_k. At z_j, the remaining coordinates are analyzed and reset, if necessary. Successive moves along the search direction follow. The procedure is stopped if there is no descent with respect to all neighboring integer points is found or when the maximum number of search steps is reached.

To summarize, the MINLP algorithm consists of the following two steps:

Algorithm 5.2 *Let* $y_0 \in \mathbb{Z}^m$ *be suitable starting points for solving the MINLP (17). For* $k = 1, 2, 3,, \ldots$ *compute* y_k *from* y_{k-1} *as follows:*

1. *Compute $F(y_{k-1})$ by solving (19), compute a matrix B_{k-1} and a vector h_k by finite differences to approximate $F(y)$ at y_{k-1}, and solve the quadratic programming problem (22) to get a new trial point $\hat{y}_k = y_{k-1} + [d_k]$, where d_k is the continuous solution of (22) and $[d_k]$ its truncated integer value.*

2. *If B_{k-1} is not positive definite, if there is no sufficient progress in objective function values, or if the $\hat{y}_k$ is too close to y_{k-1}, then start the direct search algorithm 5.1 at $\hat{y}_k$ to get y_k.*

3. *Otherwise, check termination conditions and restart the procedure, if necessary.*

There remain a couple of additional algorithmic difficulties to be taken into account. We mention only the most important ones, for more details see van de Braak [41].

1. For the very first solution of the continuous problem (19), we need a starting point provided by the user. Depending on the question how drastically model functions change if an integer variable is in- or decreased, we have the possibility either to apply the given starting point subsequently or to start always from the last computed continuous solution. This decision is made by the user.

2. The full approximation of the curvature of $F(y)$ at an iterate $y_k \in \mathbb{Z}^m$ is very time-consuming for the same reason, at least in case of expensive real-life simulations. Instead of full approximation, it is also possible to compute only the diagonal elements of B_k by a two-sided difference formula.

3. For mixed-integer programming, there is no adequate criterion to check local convergence. The stopping rule is chosen heuristically and there is no guarantee for local or even global convergence in strict mathematical sense.

6 Implementation and Numerical Tests

The proposed mixed-integer nonlinear programming algorithm is implemented in form of a Fortran subroutine, see van de Braak [41], and denoted by MINLPQL. Relaxed nonlinear programs of the form (19) are solved by the SQP code NLPQL of Schittkowski [35]. The corresponding quadratic programming subproblems of NLPQL are passed to QL, an implementation of the primal-dual method of Goldfarb and Idnani [11], see Powell [31]. The subroutine is also used for finding optimal solutions of quadratic programming subproblems of the nonlinear integer program (22). In case of a continuous SQP method, all matrices B_k of subproblem (8) are guaranteed to be positive definite. When approximating the integer function $F(y)$ quadratically, see (22), positive definite matrices cannot be ensured. However, if it turns out that an initial Cholesky decomposition cannot be obtained, the diagonal

of matrix B_k is increased until a successful decomposition is obtained. A solution of (22) is rejected and the direct search cycle is started, if QL reports an error or if the solution value is not smaller than the value at the actual iterate y_k.

The first order derivative of the integer function $F(y)$ at y_k is approximated by two-sided central differences at neighboring grid points. Second order approximations needed for the matrix B_k in (22) are obtained by two different procedures:

Case 1: Only the diagonal of B_k is computed by a central 2nd order difference formula, where function values also needed for the gradient approximation are used.

Case 2: The diagonal values of B_k are calculated as before, all other coefficients are computed by 2nd order differences.

All integer iterates y_k and corresponding function values $F(y_k)$ are stored in internal arrays to avoid multiple solution of expensive nonlinear programs of the form (19). Starting values for NLPQL are to be provided by the user. In subsequent iterations NLPQL is restarted at an optimal solution from a previous run, where the corresponding set of integer values is closest to the actual iterate y_k.

The numerical tests of this section are performed on a Siemens Celsius 620 with a dual Pentium III processor (700 MHz) running under Windows 2000. The Fortran codes are compiled by Compaq Visual Fortran, Version 6.5.

The subsequent examples are chosen to perform a few benchmark tests and to highlight some numerical features of MINLPQL. A strictly convex quadratic function in the integer variables is combined with highly nonlinear mixed terms in the integer and continuous variables,

$$
\begin{array}{rll}
 & \min & 100(y_1(2y_1+y_2)+y_2(y_1+2y_2)+y_3^2) \\
 & & +\ a(|y_1|+|y_2|+|y_3|+12|y_1y_2|+12|y_2y_3|+12|y_1y_3|) \\
 & & +\ \exp\left(0.01(x_1-y_1)^2\right)+(1.25x_2-y_3)^4+100x_3^2+100x_4^2 \\
x\in\mathbb{R}^4, & & \\
y\in\mathbb{Z}^3 & : & x_1-x_3-y_1+y_3\le 0\ , \\
 & & x_2-x_4-y_2+y_3\le 0\ , \\
 & & -100\le x_i\le 100 \text{ for } i=1,\ldots,4\ , \\
 & & -100\le y_i\le 100 \text{ for } i=1,\ldots,3\ .
\end{array}
\tag{23}
$$

The exact solution is $x^\star = 0$ and $y^\star = 0$, and we have a relatively wide range of integer values. For $a > 0$, the objective function is non-smooth at the optimal solution sunject to the integer variables, a severe handicap for the proposed method. Although possible in this case, we emphasize again that we do not assume that derivatives of the model functions with respect to the integer variables can be obtained by relaxation. Only crude approximations based on neighboring grid points are allowed.

Termination accuracy of NLPQL is set to 10^{-14}, and starting values are

$$y_0 = (-10,-20,-20)^T\ , \quad x_0 = (-10,-20,35,50)^T\ .$$

B_k	n_{NLPQL}	it_{QP}	it_{DS}	n_f	n_g
diagonal	37	4	2	6,944	6,519
full	33	3	0	3,804	3,751

Table 1: Overall performance for $a = 0$

k	y_k^1	y_k^2	y_k^3	x_k^1	x_k^2	x_k^3	x_k^4	$F(y_k)$
0	-10	-20	-20	-10.000	-23.031	0.000	16.969	214,762.04
1	10	3	3	7.000	2.407	0.000	0.000	28,701.09
2	-2	-5	0	-2.000	-3.279	0.000	1.721	8,379.41
3	2	0	0	2.000	-0.006	0.000	0.000	801.00
4	1	0	0	1.000	-0.003	0.000	0.000	201.00
5	0	0	0	0.000	0.000	0.000	0.000	1.00

Table 2: Iteration cycles for diagonal approximation B_k and $a = 0$

We count the number of calls of NLPQL, n_{NLPQL}, the number of QP and DS iterations of MINLPQL, it_{QP} and it_{DS}, and the total number of all function and gradient evaluations, n_f and n_g. Results are reported for diagonal approximation of the curvature of $F(y_k)$, and for full approximation by 2nd order difference formulae.

First we consider the smooth case $a = 0$. When approximating only the diagonal elements of B_k, the QP algorithms terminates after 4 steps because of internal round-off errors, and a direct search cycle is started, see Table 1. Thus, this variant requires more calls of NLPQL and more function and gradient evaluations. The full approximation needs only two steps until convergence of the outer mixed-integer algorithm. In both cases, the overall number of function and gradient calls is relatively high. Reasons are a bad starting point and the necessity to choose relatively small termination accuracy for NLPQL. Otherwise, the code would produce correct integer values, but incorrect continuous values because the influence of the exponential term. When taking a closer look at the iterates of Table 2 and 3, we see that the quadratic approximation approaches the solution quite rapidly. Slower convergence is observed for the diagonal approximation. The continuous variable x_2 is not approximated exactly because of internal scaling of objective function values subject to the first starting point.

k	y_k^1	y_k^2	y_k^3	x_k^1	x_k^2	x_k^3	x_k^4	$F(y_k)$
0	-10	-20	-20	-10.000	-23.031	0.000	16.969	214,762.04
1	1	-2	-1	1.000	-2.677	0.000	0.323	741.73
2	0	0	0	0.000	-0.006	0.000	0.000	1.00

Table 3: Iteration cycles for full approximation B_k and $a = 0$

B_k	n_{NLPQL}	it_{QP}	it_{DS}	n_f	n_g
diagonal	71	3	8	31,308	28,899
full	33	3	0	3,753	3,700

Table 4: Overall performance for $a = 10$

k	y_k^1	y_k^2	y_k^3	x_k^1	x_k^2	x_k^3	x_k^4	$F(y_k)$
0	-10	-20	-20	-10.000	-23.031	0.000	16.969	311,262.04
1	22	10	20	2.104	15.994	0.104	0.000	304,573.46
2	-14	-24	-18	-14.000	-21.846	0.000	20.154	425,084.05
3	21	10	20	1.104	16.003	0.104	0.000	290,363.46
4	17	6	16	1.021	12.794	0.021	0.000	167,802.89
5	13	2	12	1.005	9.594	0.005	0.000	79,194.21
6	9	-2	8	1.002	6.003	0.002	0.003	32,711.95
7	5	2	4	1.000	3.193	0.000	0.000	14,071.17
8	1	-2	0	1.000	-1.742	0.000	0.258	900.13
9	-1	0	0	-1.000	-0.006	0.000	0.000	211.00
10	0	0	0	0.000	0.000	0.000	0.000	1.00

Table 5: Iteration cycles for diagonal approximation B_k and $a = 10$

For the next example, we choose $a = 10$. The solution is not changed, but the structure of the optimization problem is. Since the objective function $F(y)$ is non-smooth at the optimal solution, we have to expect numerical difficulties close to zero, and an earlier switch to the direct search algorithm. Despite of the non-smoothness, MINLPQL performs only three QP steps in case of full approximation of the curvature of $F(y_k)$, see Table 4. On the other hand, the diagonal approximation leads to quadratic approximation with a higher function values after three steps, which requires a switch to the DS algorithm and a much higher number of function and gradient calls, cf. Tables 5 and 6.

The subsequent results are obtained again for example (23), but now with $a = 100$, i.e., the degree of non-smoothness is much higher. In both cases, a few QP iterates are made followed by about the same number of direct search steps, see Table 7. The reason is in both situations, that the quadratic approximation is

k	y_k^1	y_k^2	y_k^3	x_k^1	x_k^2	x_k^3	x_k^4	$F(y_k)$
0	-10	-20	-20	-10.000	-23.031	0.000	16.969	311,262.04
1	2	-3	0	2.000	-2.359	0.000	0.641	2,287.69
2	0	0	0	0.000	-0.006	0.000	0.000	1.00

Table 6: Iteration cycles for full approximation B_k and $a = 10$

B_k	n_{NLPQL}	it_{QP}	it_{DS}	n_f	n_g
diagonal	53	1	7	8,303	8,124
full	77	4	6	18,702	18,616

Table 7: Overall performance for $a = 100$

k	y_k^1	y_k^2	y_k^3	x_k^1	x_k^2	x_k^3	x_k^4	$F(y_k)$
0	-10	-20	-20	-10.000	-23.031	0.000	16.969	1,179,762.04
1	100	66	100	100.000	-22.212	100.000	16.831	300,551,627.69
2	-9	-20	-20	-10.000	-23.031	0.000	16.969	1,123,862.05
3	-5	-16	-16	-10.000	-19.200	0.000	12.800	621,181.28
4	-1	-12	-12	-10.000	-15.240	0.000	8.760	260,046.34
5	3	-8	-8	-10.000	-11.060	0.000	4.940	156,097.06
6	-1	-4	-4	-10.000	-6.400	0.000	1.600	36,014.24
7	3	0	0	-10.000	-0.008	0.000	0.000	2,105.41
8	0	0	0	-10.000	-0.008	0.000	0.000	2.71

Table 8: Iteration cycles for diagonal approximation B_k and $a = 100$

incorrect after a few steps leading to an increase of objective function values, see Table 8 and 9. The total number of NLPQL calls is now much larger for the full approximation, obviously caused by the non-differentiability of $F(y)$. Because of the high nonlinearity of the optimization problem, the diagonal approximation of B_k leads to a situation where the first variable x_1 is kept at the starting value, $x_k^1 = -10$ for all $k > 1$. Thus, the nonlinear programs are solved with a termination tolerance of 10^{-16} in this case to achieve convergence to the optimal solution.

7 Case study: Optimal Design of an Electronic Filter

The electronic filter to be optimized in this case study, is designed for rf-application in cellular phones. The design has to satisfy the specification given for four frequency intervals R_0, R_1, R_2, and R_3 as shown in Figure 4. The optimization goal is to maximize the transmission $T(f_i, x, y)$ over the interval of R_0 by satisfying lower bounds in R_1 and R_2. The lower bounds of area R_3 are considered being weak ones, i.e., may be violated, but the violation should be as low as possible. Thus, a penalty term is added to the objective function, to penalize bound violations in R_3. The results are obtained on a Siemens Celsius 620 with a dual Intel Pentium III processor (700 MHz) running under Linux, Kernel 2.2.10.

Lower and upper bounds for 10 continuous and 3 integer variables are shown in Table 10 together with initial values and final ones obtained by the code MINLPQL.

k	y_k^1	y_k^2	y_k^3	x_k^1	x_k^2	x_k^3	x_k^4	$F(y_k)$
0	-10	-20	-20	-10.000	-23.031	0.000	16.969	1,179,762.04
1	9	-12	-16	9.001	-18.579	0.000	9.421	597,099.54
2	-2	-2	-14	-2.000	-14.400	0.000	1.600	96,313.00
3	6	5	-19	6.000	-15.192	0.000	0.000	344,101.00
4	-1	-1	-14	-1.000	-13.966	0.000	1.034	56,850.82
5	0	0	-13	0.000	-12.529	0.000	0.471	18,273.34
6	0	0	-9	0.000	-8.800	0.000	0.200	9,021.00
7	0	0	-5	0.000	-4.957	0.000	0.043	3,003.23
8	0	0	-1	0.000	-1.000	0.000	0.000	201.00
9	0	0	0	0.000	0.000	0.000	0.000	1.00

Table 9: Iteration cycles for full approximation B_k and $a = 100$

Simulation is performed with respect to 154 frequency points leading to 174 constraints in the continuous model. The initial and final transmissions are shown in Figure 4, also for the continuous case with integer variables fixed at the starting values. We observe that the constraint R_3 remains violated. The example shows that the design goal is only achieved by taking also integer variables into account.

In the QP algorithm, only the diagonal elements of B_k are computed. If two integer iterates possess an Euclidian distance less than 5, MINLPQL switches from the quadratic programming to the direct search algorithm. The number of steps of the direct search algorithm is limited to five. Altogether 36 calls of NLPQL are made within 4 QP cycles and one additional DS iteration. The total number of simulations, i.e., the number of evaluations of the transmission energy $T_j(x, y)$ for all j, is 434 without the function calls needed for the gradient approximations. Table 11 shows the main iterates of the MINLP algorithm.

8 Conclusions

A new algorithm is proposed for constrained mixed-integer nonlinear programming (MINLP), that does not assume availability of a continuous relaxation. In other words, it is supposed that the simulation code is unable to compute function values between given grid points. Especially, we assume that gradient information subject to the integer variables is not available, and that the optimization problem is highly nonlinear and non-convex in the continuous variables. The MINLP problem is transformed into a pure integer problem, where a nonlinear function depending only on integer values is to be minimized subject to lower and upper bounds. One evaluation of this function requires the complete optimization of the corresponding continuous formulation, where integer variables are fixed.

The algorithm proceeds from successive quadratic approximations based on 2-

variable	*lower bound*	*initial value*	*optimal value*	*upper bound*
y_1	7	19	12	19
y_2	12	24	14	25.0
y_3	100	130	124	150
x_1	5.0	11.58	9.589	15.0
x_2	50.0	50.0	92.39	150.0
x_3	10.5	11.39	11.25	11.5
x_4	10.0	10.61	10.62	11.0
x_5	0.3	0.3	0.3	0.5
x_6	0.95	1.033	1.031	1.05
x_7	0.95	1.031	1.023	1.05
x_8	0.95	1.012	1.015	1.02
x_9	0.985	1.001	0.998	1.03
x_{10}	1.0	1.0	1.000	1.03

Table 10: Bounds, initial, and optimal values for design variables

sided second-order difference formulae for getting derivatives from neighboring grid points. If the QP solution is not acceptable, a direct search is started. The feasibility of the method is outlined by a benchmark example. The results show that a good approximation of the integer solution can be obtained in very few QP steps provided that the model function is sufficiently smooth. In some cases, the switch to the direct search part can be prevented by full approximation of the curvature of the integer function. Despite of the large amount of additional function evaluations, the overall performance is better than in case of diagonal approximation. If, on the other hand, the objective function is sufficiently non-smooth in the integer values at the optimal solution, then the approximation of the diagonal leads to better results.

However, the total number of all function and gradient evaluations subject to the continuous variables could become extremely large, in particular if the quadratic approximation of the objective function is misleading. In case of the academic test problem under consideration, the starting point is far away from the optimal solution and causes a larger number of iterations.

In real-life environments, we can expect much better performance, if good initial guesses are available. Thus, we present some results for the optimal design optimization of electronic filters based on the piezo-acoustic effect. The model is outlined and the influence of integer variables is shown. The basic idea is to maximize the transmission in a given frequency range subject to lower bounds for the transmission in some other ranges. A practical design problem shows that the proposed algorithm can be applied also in cases, where function evaluations are extremely expensive because of the highly complex numerical simulation.

k	0	1	2	3	4	5
y_k^1	19	17	12	12	12	12
y_k^2	24	19	13	14	14	14
y_k^3	130	116	116	122	123	124
x_k^1	11.723	10.301	9.191	9.548	9.591	9.589
x_k^2	50.000	64.100	91.921	93.907	92.940	92.391
x_k^3	11.389	11.350	11.255	11.257	11.257	11.256
x_k^4	10.616	10.618	10.621	10.617	10.617	10.617
x_k^5	0.300	0.300	0.300	0.300	0.300	0.300
x_k^6	1.033	1.033	1.030	1.031	1.031	1.031
x_k^7	1.033	1.029	1.021	1.023	1.023	1.023
x_k^8	1.013	1.012	1.015	1.015	1.015	1.015
x_k^9	1.001	0.995	0.998	0.998	0.998	0.998
x_k^{10}	1.000	1.000	1.000	1.000	1.000	1.000
$f(x_k, y_k)$	137.479	40.757	1.312	0.052	0.035	0.000

Table 11: Iteration cycles for the case study

9 Acknowledgements

The authors would like to thank EPCOS AG, Munich, for supporting our research, and the referees for numerous suggestions helping us to improve the presentation.

References

[1] Audet C., Dennis J.E. (2001): *Pattern search algorithms for mixed variable programming*, SIAM Journal on Optimization, Vol. 11, 573-594

[2] Beale E.M.L. (1978): *Integer programming*, in: *The State of the Art in Numerical Analysis*, D.A.H. Jacobs ed., Academic Press, London

[3] Cha J.Z., Mayne R.W. (1989): *Optimization with discrete variables via recursive quadratic programming: part 1 - concepts and definitions*, Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 111, 124-129

[4] Cha J.Z., Mayne R.W. (1989): *Optimization with discrete variables via recursive quadratic programming: part 2 - algorithm and results*, Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 111, 130-136

[5] Dakin R.J. (1965): *A tree search algorithm for mixed integer programming problems*, Computer HJournal, Vol. 8, 250-255

[6] Duran M., Grossmann I.E. (1986): *An outer-approximation algorithm for a class of mixed-integer nonlinear programs*, Mathematical Programming, Vol. 36, 307-339

[7] Edgar T.F., Himmelblau D.M. (1988): *Optimization of Chemical Processes,* McGraw Hill

[8] Fischer A. (1992): *A special Newton-type optimization method*, Optimization, Vol. 24, 269-284

[9] Floudas C.A. (1995): *Nonlinear and Mixed-Integer Optimization, Fundamentals and Applications*, Oxford University Press, New York

[10] Gill P.E., Murray W., Wright M.H. (1981): *Practical Optimization,* Academic Press

[11] Goldfarb D., Idnani A. (1983): *A numerically stable method for solving strictly convex quadratic programs*, Mathematical Programming, Vol. 27, 1-33

[12] Grossmann I.E., Kravanja Z. (1997): *Mixed-integer nonlinear programming: a survey of algorithms and applications*, in: Biegler, Coleman, Conn, Santosa (eds.): The IMA Volumes in Mathematics and its Applications, Vol. 93, *Large-Scale Optimization with Applications. Part II: Optimal Design and Control*, Springer, Berlin, 73-100

[13] Gupta O.K., Ravindran A. (1983): *Nonlinear integer optimization and discrete optimization*, Transactions of the ASME, Vol. 105, 160-164

[14] Han S.-P. (1977): *A globally convergent method for nonlinear programming*, Journal of Optimization Theory and Applications, Vol. 22, 297–309

[15] Hartwanger C., Schittkowski K., Wolf H. (2000): *Computer aided optimal design of horn radiators for satellite communication*, Engineering Optimization, Vol. 33, 221-244

[16] Hock W., Schittkowski K. (1983): *A comparative performance evaluation of 27 nonlinear programming codes*, Computing, Vol. 30, 335-358

[17] Hough P.D., Kolda T.G., Torczon V.J. (2001): *Asynchronous Parallel Pattern Search for Nonlinear Optimization*, SIAM Journal on Scientific Computing, Vol. 23, 134-156

[18] Kelley C.T. (1999): *Iterative Methods for Optimization*, SIAM, Philadelphia

[19] Kleber, W. (1985): *Einführung in die Kristallographie*, VEB Verlag Technik, Berlin

[20] Kocis G.R., Grossmann I.E. (1988): *Global optimization of nonconvex mixed-integer nonlinear programming (MINLP) problems in process synthesis*, Industrial and Engineering Chemical Research, Vol. 27, 1407-1421

[21] Lewis R.M., Torczon V., Trosset M.W. (2000): *Direct search methods: then and now*, Journal of Computational and Applied Mathematics, Vol. 124, 191-207

[22] Leyffer S. (1993):*Deterministic methods for mixed integer nonlinear programming*, PhD Thesis, Department of Mathematics and Conputer Science, University of Dundee

[23] Leyffer S. (2001):*Integrating SQP and branch-and-bound for mixed integer nonlinear programming*, Computational Optimization and Applications, Vol. 18, 295-309

[24] Li H., Chou C. (1994): *A global approach for nonlinear mixed discrete programming in design optimization*, Engineering Optimization, Vol. 22, 109-122

[25] Loh H.T., Papalambros P. (1991): *A sequential linearization approach for solving mixed-discrete nonlinear design optimization problems*, Journal of Mechanical Design, Vol. 113, 325-334

[26] Miettinen K., Mäkelä M.M., Neittaanmäki P., Periaux J (1999): *Evolutionary Algorithms in Engineering and Computer Schience*, John Wiley, Chichester

[27] Olsen G.R., Vanderplaats G.N. (1989): *Method for nonlinear optimization with discrete design variables*, AIAA Journal, Vol. 27, 1584-1589

[28] Papalambros P.Y., Wilde D.J. (2000): *Principles of Optimal Design,* Cambridge University Press

[29] Powell M.J.D. (1978): *A fast algorithm for nonlinearly constraint optimization calculations,* in: *Numerical Analysis*, G.A. Watson ed., Lecture Notes in Mathematics, Vol. 630, Springer

[30] Powell M.J.D. (1978): *The convergence of variable metric methods for nonlinearly constrained optimization calculations*, in: *Nonlinear Programming 3*, O.L. Mangasarian, R.R. Meyer, S.M. Robinson eds., Academic Press

[31] Powell M.J.D. (1983): *On the quadratic programming algorithm of Goldfarb and Idnani*. Report DAMTP 1983/Na 19, University of Cambridge, Cambridge

[32] Schittkowski K. (1980): *Nonlinear Programming Codes,* Lecture Notes in Economics and Mathematical Systems, Vol. 183 Springer

[33] Schittkowski K. (1983): *Theory, implementation and test of a nonlinear programming algorithm*, in: Optimization Methods in Structural Design, H. Eschenauer, N. Olhoff eds., Wissenschaftsverlag

[34] Schittkowski K. (1983): *On the convergence of a sequential quadratic programming method with an augmented Lagrangian search direction,* Optimization, Vol. 14, 197-216

[35] Schittkowski K. (1985/86): *NLPQL: A Fortran subroutine solving constrained nonlinear programming problems,* Annals of Operations Research, Vol. 5, 485-500

[36] Schwefel H.-P. (1994): *Evolution and Optimum Seeking*, John Wiley, New York

[37] Spellucci P. (1993): *Numerische Verfahren der nichtlinearen Optimierung,* Birkhäuser

[38] Stoer J. (1985): *Foundations of recursive quadratic programming methods for solving nonlinear programs,* in: *Computational Mathematical Programming*, K. Schittkowski, ed., NATO ASI Series, Series F: Computer and Systems Sciences, Vol. 15, Springer

[39] Tobolka G. (1979): *Mixed matrix representation of SAW transducers*, Proceedings of the IEEE Ultrasonics Symposium, Vol. 26, 426-428

[40] Torczon V. (1997): *On the Convergence of Pattern Search Algorithms*, SIAM Journal on Optimization, Vol. 7, 1-25

[41] van de Braak G. (2001): *Das Verfahren SQMIP zur gemischt ganzzahligen nichtlinearen Programmierung für den Entwurf elektronischer Bauteile*, Diploma Thesis, Department of Mathematics, University of Münster, Germany

[42] Wang S., Teo K.L., Lee H.W.J. (1998): *A new approach to nonlinear mixed discrete programming problems*, Engineering Optimization, Vol. 30, 249-262

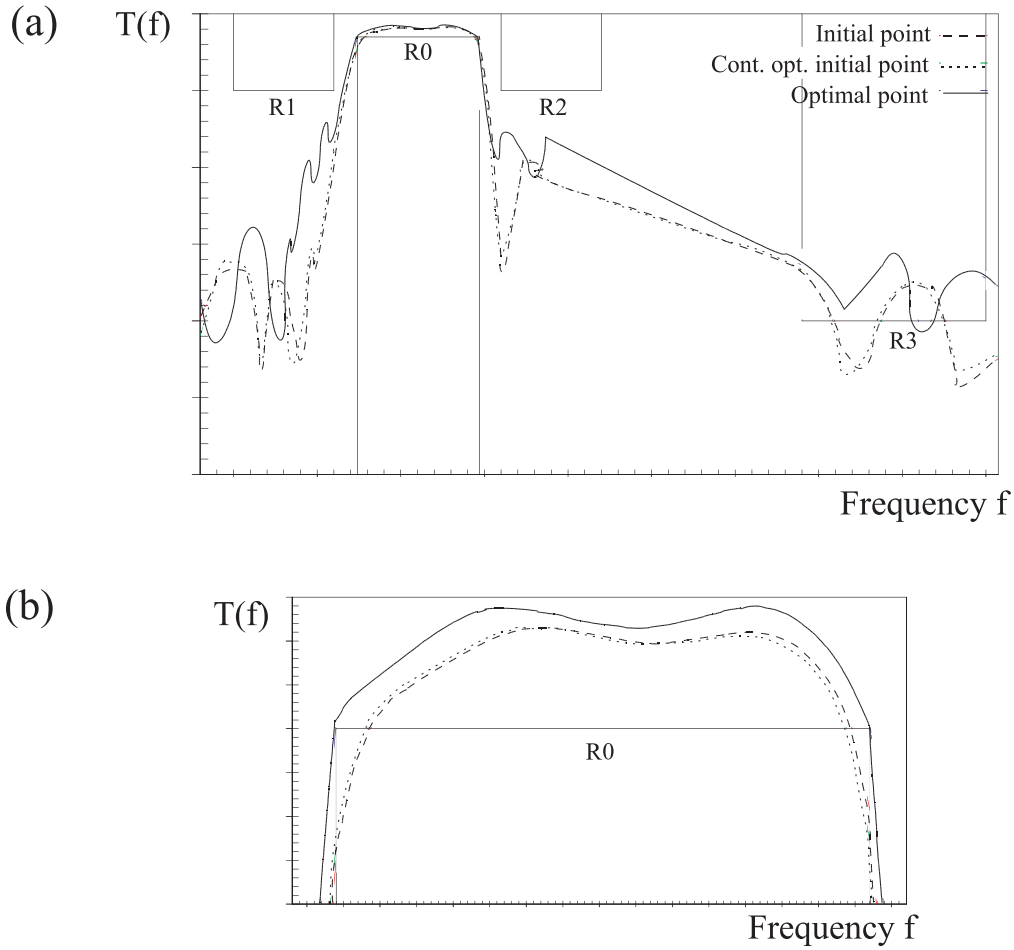

Figure 4: Initial and optimal filter designs

.