Sequential Convex Programming Methods

K. Schittkowski[1], C. Zillober[2]

Abstract. Sequential convex programming methods became very popular in the past for special domains of application, e.g. the optimal structural design in mechanical engineering. The algorithm uses an inverse approximation of certain variables so that a convex, separable nonlinear programming problem must be solved in each iteration. In this paper the method is outlined and it is shown, how the iteration process can be stabilized by a line search. The convergence results are presented for a special variant called *method of moving asymptotes*. The algorithm was implemented in FORTRAN and the numerical performance is evaluated by a comparative study, where the test problems are formulated through a finite element analysis.

Keywords. Nonlinear programming, structural optimization, convex approximation

1 Introduction

In this paper we proceed from the general formulation of a nonlinear programming problem

$$\begin{array}{rll} & \min h_0(\boldsymbol{x}) & \\ \boldsymbol{x} \in \mathbb{R}^n : & h_j(\boldsymbol{x}) \leq 0 & , j = 1, ..., m \\ & \boldsymbol{x}_l \leq \boldsymbol{x} \leq \boldsymbol{x}_u & \end{array} \tag{1}$$

where all problem functions are continuously differentiable. We may imagine, for example, that the objective function describes the weight of a structure that is to be minimized subject to sizing variables, and that the constraints impose limitations on structural response quantities, e.g. upper bounds for stresses or displacements under static loads. Many other objectives or constraints can be modelled in a way so that they fit into the above general frame.

Especially in design optimization for mechanical engineering, sequential convex programming (SCP) algorithms as developed by Fleury (1989), Svanberg (1987) and others became very popular for several reasons:

[1]Mathematisches Institut, Universität Bayreuth,D–95440 Bayreuth, Germany
[2]IWR, Universität Heidelberg, D–69120 Heidelberg, Germany

1. The mathematical formulation of certain types of constraints, in particular of stress constraints, contains inverse optimization variables e.g. in case of cross-sections of bars. Thus an inverse approximation of these variables *linearizes* the problem functions defining the restrictions.

2. SCP methods are first order methods, which are attractive in situations, where round-off errors prevent the precise evaluation of gradients, and are therefore unable to update any second order information in a sufficiently accurate manner.

3. SCP methods are able to solve large optimization problems with hundreds or even thousands of variables, since the convex and separable subproblem to be solved in each iteration, can be adapted to large scale optimization and is solved easily also in these situations.

As a result of these observations, the finite element analysis of very many structural design software systems was extended by optimization modules based on convex approximation methods, see e.g. Hörnlein and Schittkowski (1992) for a review. Practical experience shows that SCP methods are often much more efficient than other methods, e.g. sequential quadratic programming, feasible direction or generalized reduced gradient methods.

However the *typical* implemetation of an SCP methods has a severe drawback: The methods are not stabilized in the sense that convergence towards a solution from an arbitray initial design, is not guaranteed. In Zillober (1993a), a stabilization by a line search procedure was proposed which allows to prove global convergence theorems. The main results are repeated in this paper.

Moreover we summarize the results of an extensive comparative study of structural optimization codes, where the analysis is based on a finite element formulation and performed by the software system MBB-LAGRANGE, see Kneppe, Krammer, and Winkler (1987) or Zotemantel (1993). Besides of the nine optimization algorithms included in the official version, two additional methods are added to the system, i.e. certain variants of convex approximation methods. The codes represent all major classes of algorithms that are in practical use at present.

To conduct the numerical tests, 79 design problems have been collected. Most of them are *academic* ones, i.e. are more or less simple design problems found in the literature. The remaining ones possess some practical *real life* background from project work or are suitable modifications to act as benchmark test problems for the development of the software system. In all situations, we minimize the weight of a structure subject to displacement, stress, strain, buckling, dynamic and other constraints. Design variables are sizing variables, e.g. cross sectional areas and thicknesses of skeletal as well as membran or shell structures.

In the following section we will describe some convex approximations used for SCP methods. The optimization algorithm is outlined in Section 3 together with some convergence results. Section 4 contains a summary of computational results which allow a direct comparison with other methods.

2 Convex Approximations

By using reciprocal variables, Fleury and Braibant (1986) developed an optimization method called CONLIN (convex linearization). An approximation of a function is defined by separate linearization for each variable depending on the sign of the partial derivative at the expansion point. If the sign is positive then the linearization is performed with respect to the original variable. Otherwise the approximation is obtained with respect to the inverse variable, leading to a convex approximation of the original function.

In a more formal notation, we replace the original problem functions $h_j(\boldsymbol{x})$, $j = 0, \ldots, m$, with respect to a given iterate $\boldsymbol{x}_k \in \mathbb{R}^n$ by

$$\overline{h}_j(\boldsymbol{x}) := h_j(\boldsymbol{x}_k) + \sum_{i \in I^+_{j,k}} \frac{\partial}{\partial x_i} h_j(\boldsymbol{x}_k)(x_i - x_i^k) - \sum_{i \in I^-_{j,k}} \frac{\partial}{\partial x_i} h_j(\boldsymbol{x}_k)(x_i^k / x_i - x_i^k) \quad (2)$$

where $\boldsymbol{x} = (x_1, ..., x_n)^T$ and $\boldsymbol{x}_k = (x_1^k, ..., x_n^k)^T$ and where

$$\begin{aligned} I^-_{j,k} &:= \{i : 1 \le i \le n, \frac{\partial}{\partial x_i} h_j(\boldsymbol{x}_k) \le 0\} \\ I^+_{j,k} &:= \{i : 1 \le i \le n, \frac{\partial}{\partial x_i} h_j(\boldsymbol{x}_k) > 0\} \end{aligned}$$

for $j = 0, \ldots, m$, and where $\overline{h}_j(\boldsymbol{x})$ is defined for all $\boldsymbol{x} > 0$.

The reason for inverting design variables in the above way is, that e.g. in design optimization stresses and displacements are exact linear functions of the reciprocal linear homogeneous sizing variables in case of a statically determinated structure. Moreover the numerical experience shows that also in other cases, convex linearization is applied quite successfully in practice, in particular in shape optimization, although a mathematical motivation cannot be given in this case.

After some reorganization of constant data, we get a convex and separable subproblem of the following form:

$$\begin{gathered} \min \textstyle\sum_{i \in I^+_{0,k}} c^i_{0,k} x_i - \sum_{i \in I^-_{0,k}} c^i_{0,k} / x_i \\ \boldsymbol{x} \in \mathbb{R}^n : \quad \textstyle\sum_{i \in I^+_{j,k}} c^i_{j,k} x_i - \sum_{i \in I^-_{j,k}} c^i_{j,k} / x_i + \overline{c}^i_{j,k} \le 0, \quad j = 1, ..., m \\ \boldsymbol{x}_l \le \boldsymbol{x} \le \boldsymbol{x}_u \end{gathered} \quad (3)$$

where $c^i_{j,k}$ and $\overline{c}^i_{j,k}$ are the constant parameters of the convex approximation with respect to objective function and constraints, i.e. are defined by

$$\overline{h}_j(\boldsymbol{x}) := \overline{c}^i_{j,k} + \sum_{i \in I^+_{j,k}} c^i_{j,k} x_i - \sum_{i \in I^-_{j,k}} c^i_{j,k} / x_i$$

for $j = 0, \ldots, m$. Without loss of generality we assume that $\boldsymbol{x}_l > 0$.

The solution of the above problem determines then the next iterate $\boldsymbol{x}_{k+1}$. We do not investigate here the question how the mathematical structure of the subproblem can be exploited to get an efficient solution algorithm. As long as the problem is not too big, we may assume without loss of generality, that is solved by any standard nonlinear programming technique.

To control the degree of convexification and to adjust it with respect to the problem to be solved, Svanberg (1987) introduced so-called moving asymptotes U_i and L_i to replace x_i and $1/x_i$ by

$$\frac{1}{x_i - L_i} \quad , \quad \frac{1}{U_i - x_i}$$

where L_i and U_i are given parameters, which can also be adjusted from one iteration to the next. The algorithm is called *method of moving asymptotes*. The larger flexibility allows a better convex approximation of the problem and thus a more efficient and robust solution.

In this case, the approximation is of the form

$$\begin{aligned} \overline{h}_j(\boldsymbol{x}) \quad := \quad & h_j(\boldsymbol{x}_k) + \sum_{i \in I^+_{j,k}} \frac{\partial}{\partial x_i} h_j(\boldsymbol{x}_k) \left(\frac{(U_i - x_i^k)^2}{U_i - x_i} - (U_i - x_i^k) \right) \\ & - \sum_{i \in I^-_{j,k}} \frac{\partial}{\partial x_i} h_j(\boldsymbol{x}_k) \left(\frac{(x_i^k - L_i)^2}{x_i - L_i} - (x_i^k - L_i) \right) \end{aligned} \tag{4}$$

which is defined for all $x \in \mathbb{R}^n$ with $L_i < x_i < U_i$, $i = 1, \ldots, n$, and for all j with $0 \leq j \leq m$.

It is easy to verify that $\overline{h}_j(\boldsymbol{x})$ is a first order approximation of $h_j(\boldsymbol{x})$ at $\boldsymbol{x}_k$, i.e. that

$$\overline{h}_j(\boldsymbol{x}_k) = h_j(\boldsymbol{x}_k) \quad \text{and} \quad \nabla \overline{h}_j(\boldsymbol{x}_k) = \nabla h_j(\boldsymbol{x}_k) \, ,$$

and that $\overline{h}_j(\boldsymbol{x})$ is a convex and separable function, $j = 0, \ldots, m$. The first convex approximation (2) can be considered as a limit case of the approximation by moving asymptotes, since we get it back through $L_i = 0$ and $U_i \to \infty$.

Practical experience shows, that in some cases the approximation of the objective function is almost linear because of small constants $c^i_{0,k}$ e.g. in the neighbourhood of a solution. To avoid instabilities when solving the subproblem, Svanberg (1993) suggested to append a quadratic term to the approximation of the objective function which guarantees strict convexity of the objective function. Proceeding from the notation

$$\overline{h}_j(\boldsymbol{x}) := \overline{c}^i_{j,k} + \sum_{i \in I^+_{j,k}} \frac{c^i_{j,k}}{U_i - x_i} - \sum_{i \in I^-_{j,k}} \frac{c^i_{j,k}}{x_i - L_i}$$

with suitable constants $c^i_{j,k}$ and $\overline{c}^i_{j,k}$, we get the modified objective function of the subproblem by

$$\overline{h}_0(\boldsymbol{x}) := \overline{c}^i_{0,k} + \sum_{i \in I^+_{0,k}} \frac{c^i_{0,k} + \epsilon (x_i - x_i^k)^2}{U_i - x_i} - \sum_{i \in I^-_{0,k}} \frac{c^i_{0,k} + \epsilon (x_i - x_i^k)^2}{x_i - L_i} \tag{5}$$

with a suitable constant $\epsilon > 0$.

Moreover we have to avoid that a solution of the subproblem approaches one of the asymptotes. Thus we introduce a sufficiently small constant $\omega > 0$ and define new bounds by

$$\begin{aligned} \overline{x}_i^l &:= \max\{x_i^l, L_i + \omega(x_i^k - L_i)\} \\ \overline{x}_i^u &:= \min\{x_i^u, U_i - \omega(U_i - x_i^k)\} \end{aligned}$$

where $\boldsymbol{x}_l = (x_1^l, \ldots, x_n^l)^T$, $\boldsymbol{x}_u = (x_1^u, \ldots, x_n^u)^T$. With $\overline{\boldsymbol{x}}_l = (\overline{x}_1^l, \ldots, \overline{x}_n^l)^T$ and $\overline{\boldsymbol{x}}_u = (\overline{x}_1^u, \ldots, \overline{x}_n^u)^T$ we finally get the subproblem

$$\boldsymbol{x} \in \mathbb{R}^n : \quad \begin{array}{l} \min \overline{h}_0(\boldsymbol{x}) \\ \overline{h}_j(\boldsymbol{x}) \leq 0 \qquad , j = 1, ..., m \\ \overline{\boldsymbol{x}}_l \leq \boldsymbol{x} \leq \overline{\boldsymbol{x}}_u \end{array} \tag{6}$$

This approximation remains separable and is strictly convex with two major advantages:

- The subproblem has a unique solution, if it is at all solvable.
- Very efficient dual methods for solving the subproblem are applicable, cf. Fleury (1989), Svanberg (1987) or Zillober (1992).

Next we state a result which is important to identify a solution.

Lemma 2.1: *$\boldsymbol{x}^\star$ is a stationary point of (1) if and only if $\boldsymbol{x}^\star$ is a stationary point of (6).*

The proof for this lemma as well as the proofs for all following statements can be found in Zillober (1993a) and Zillober (1992).

The asymptotes L_i and U_i, respectively, can be adapted during the iteration process to keep them as tight as possible on the one hand, and to extend them on the other hand whenever it turns out that the initial choice was too narrow. To be able to prove the desired convergence results, we consider only special asymptotes.

Definition 2.2: *A strategy for the choice of asymptotes is called continuous, if for any sequence $\boldsymbol{x}_k \to \boldsymbol{x}$, $\boldsymbol{x} \in \mathbb{R}^n$, we have $L_i(\boldsymbol{x}_k) \to L_i(\boldsymbol{x})$ and $U_i(\boldsymbol{x}_k) \to U_i(\boldsymbol{x})$ for $i = 1 \ldots n$.*

By the notation $L_i(\boldsymbol{x}_k), U_i(\boldsymbol{x}_k)$ we identify the asymptotes resulting from the evaluation of the chosen strategy at the point $\boldsymbol{x}_k \in \mathbb{R}^n$. These asymptotes may depend on the current iteration point, additionally on previous iteration points, or may be independent of the iteration point. In the theorems of the subsequent section we will always assume that the strategy for the choice of the asymptotes is continuous.

3 The Sequential Convex Programming Method

A sequential convex programming method with moving asymptotes consists basically of the following steps:

1. Starting from an initial iterate $\boldsymbol{x}_0$ and initial asymptotes L_i and U_i, $i = 1, \ldots, n$, create a convex, separable subproblem of the form (6) and solve it by any optimization technique.

2. Update the asymptotes and repeat.

For the original SCP method based on subproblem (3), Nguyen et al. (1987) presented a convergence proof for the case that (1) consists of convex problem functions. They showed furthermore by some examples that a generalization of the result to non-convex constraints is not possible.

It is well known that a line search with respect to a suitable merit function stabilizes an optimization algorithm in particular by preventing too large steps outside of the feasible region. In general a line search requires additional function evaluations, i.e. extra costs.

First we reformulate (1) to get a simplified notation for the theoretical analysis of this section, by assuming that upper and lower bounds for the variables are treated as part of the general inequality constraints. Without loss of generality, we proceed now from the problem

$$\begin{array}{rll} & \min h_0(\boldsymbol{x}) & \\ \boldsymbol{x} \in \mathbb{R}^n : & h_j(\boldsymbol{x}) \leq 0 & , j = 1, ..., m \end{array} \tag{7}$$

In this case the lower and upper bounds in subproblem (6) are defined by

$$\begin{aligned} \overline{x}_i^l &:= L_i + \omega(x_i^k - L_i) \\ \overline{x}_i^u &:= U_i - \omega(U_i - x_i^k) \end{aligned}$$

Next we introduce an augmented Lagrange function by

$$\Phi_r(\boldsymbol{x}, \boldsymbol{u}) = h_0(\boldsymbol{x}) + \sum_{j=1}^{m} \begin{cases} u_j h_j(\boldsymbol{x}) + \dfrac{r}{2} h_j^2(\boldsymbol{x}) \ , & \text{if } h_j(\boldsymbol{x}) \geq -\dfrac{u_j}{r} \\ \dfrac{u_j^2}{2r} \ , & \text{otherwise} \end{cases} \tag{8}$$

which is defined for all $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{u} \in \mathbb{R}^m$. The so-called penalty parameter r must be sufficiently large and controls the degree of *penalization* when leaving the feasible region. This function is also used in a sequential quadratic programming method for solving general nonlinear optimization problems, see Schittkowski (1981).

Now we formulate the algorithm, which we call SCP or sequential convex programming method to indicate the similarity to the SQP-method.

Algorithm 3.1:

Step 0 : *Choose* $\boldsymbol{x}_0 \in \mathbb{R}^n,\ \boldsymbol{u}_0 \geq 0,\ \boldsymbol{u}_0 \in \mathbb{R}^m,\ 0 < c < 1,\ 0 < \psi < 1,\ r > 0,$ $\epsilon > 0$, *and let* $k := 0$.

Step 1 : *Compute* $h_j(\boldsymbol{x}_k),\ \nabla h_j(\boldsymbol{x}_k),\ j = 0, \ldots, m$.

Step 2 : *Compute* L_i^k *and* U_i^k , $i = 1, \ldots, n$, *by a suitable strategy, and define* $\overline{h}_j(\boldsymbol{x}_k)$ *for* $j = 0, \ldots, m$, *by (4) and (5), respectively.*

Step 3 : *Solve (6) by any internal method and let* $\boldsymbol{y}_k$, $\boldsymbol{v}_k$ *be the solution, where* $\boldsymbol{v}_k$ *denotes the corresponding vector of Lagrange multipliers.*

Step 4 : *If* $\boldsymbol{y}_k = \boldsymbol{x}_k$, *then stop,* $\boldsymbol{x}_k$ *is a stationary point.*

Step 5 : *Let* $\delta_k := \|\boldsymbol{y}_k - \boldsymbol{x}_k\|$,

$$\eta_k := \tfrac{1}{2} \min \left\{ \min_{i=1..n} \left\{ 2\epsilon \frac{(U_i^k - x_i^k)^2}{(U_i^k - L_i^k)^3} \right\} \ , \ \min_{i=1..n} \left\{ 2\epsilon \frac{(x_i^k - L_i^k)^2}{(U_i^k - L_i^k)^3} \right\} \right\}$$

Step 6 : *Compute* $\Phi_r(\boldsymbol{x}_k, \boldsymbol{u}_k)$, $\nabla \Phi_r(\boldsymbol{x}_k, \boldsymbol{u}_k)$, *and* $\gamma_k := \nabla \Phi_r(\boldsymbol{x}_k, \boldsymbol{u}_k)^T \binom{\boldsymbol{x}_k - \boldsymbol{y}_k}{\boldsymbol{u}_k - \boldsymbol{v}_k}$.

Step 7 : *If* $\gamma_k < \frac{1}{4}\eta_k \delta_k^2$, *let* $r := 10r$ *and goto step 6. Otherwise compute the smallest* $j = 0, 1, 2 \ldots$, *such that*

$$\Phi_r(\boldsymbol{x}_k - \psi^j \boldsymbol{y}_k, \boldsymbol{u}_k - \psi^j \boldsymbol{v}_k) \leq \Phi_r(\boldsymbol{x}_k, \boldsymbol{u}_k) - c\psi^j \gamma_k .$$

Set $\alpha_k := \psi^j$.

Step 8 : *Let* $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k - \alpha_k \boldsymbol{y}_k$, $\boldsymbol{u}_{k+1} := \boldsymbol{u}_k - \alpha_k \boldsymbol{v}_k$, *and* $k := k + 1$. *Then repeat with step 1.*

Suitable constants for initializing the algorithm, are $c = 0.001$, $\psi = 0.5$, $\epsilon = 0.001$ and $r = 1$.

The difficulty in proving any global convergence result for the sequential convex programming method is to show, that the search direction defined by Step 3 of the algorithm, is a descent direction for Φ_r and that the resulting sequence of penalty parameters is bounded.

The next lemma shows that the sequence $(\boldsymbol{x}_k, \boldsymbol{u}_k)_{k=0,1,2,\ldots}$ is bounded under some reasonable assumptions.

Lemma 3.2: *Let the sequence* $\{(\boldsymbol{x}_k, \boldsymbol{u}_k)\}_{k=0,1,2,\ldots}$ *be produced by Algorithm (3.1), all subproblems be solvable and gradients of active constraints at the intermediate iterates* $\boldsymbol{x}_k$ *be linear independent as well as those at any possible accumulation point of* $\{\boldsymbol{x}_k\}_{k=1,2,3,\ldots}$. *If the sequence* $\{x_k\}$ *is bounded, then also the sequence* $\{\boldsymbol{u}_k\}$ *for* $k = 1, 2, 3, \ldots$.

Note that the boundedness of $\{\boldsymbol{x}_k\}$ is guaranteed as soon as we include again our original bounds $\boldsymbol{x}_l$ and $\boldsymbol{x}_u$ for the variables. Now we are able to state the first convergence result.

Theorem 3.3: *Let the assumptions of Lemma (3.2) be valid for some iterates* $\boldsymbol{x}_k$ *and* $\boldsymbol{u}_k \geq 0$ *of Algorithm (3.1),* $k = 1, 2, 3, \ldots$, *where none of the* $\boldsymbol{x}_k$ *is a stationary point for (6). Moreover let* η_k, δ_k *be defined as in Algorithm (3.1) and let the choice of asymptotes be continuous. Then*

1) there is a penalty parameter $r_k > 0$ such that $(\boldsymbol{y}_k, \boldsymbol{v}_k)$ is a direction of descent for all $r \geq r_k$ with respect to the augmented Lagrange function Φ_r, in other words

$$\nabla \Phi_r(\boldsymbol{x}_k, \boldsymbol{u}_k)^T \begin{pmatrix} \boldsymbol{x}_k - \boldsymbol{y}_k \\ \boldsymbol{u}_k - \boldsymbol{v}_k \end{pmatrix} \geq \frac{\eta_k \delta_k^2}{4}$$

for all $r \geq r_k$,

2) for each $\delta > 0$ there is a finite r_δ such that for all $\boldsymbol{x}_k, \boldsymbol{u}_k$ and $\delta_k \geq \delta$ we have

$$\nabla \Phi_r(\boldsymbol{x}_k, \boldsymbol{u}_k)^T \begin{pmatrix} \boldsymbol{x}_k - \boldsymbol{y}_k \\ \boldsymbol{u}_k - \boldsymbol{v}_k \end{pmatrix} \geq \frac{\eta_k \delta_k^2}{4} \geq \frac{\eta \delta^2}{4}$$

for all $r \geq r_\delta$.

The next theorem shows some conditions which guarantee the boundedness of the penalty parameter also in case of $\delta_k \to 0$.

Theorem 3.4: *Let the asssumptions of Lemma (3.2) be valid and assume a continuous choice of the asymptotes in Step 2 of Algorithm (3.1). For $\delta_k \neq 0$ we define $\gamma_k := (\|\boldsymbol{u}_k - \boldsymbol{v}\|/\delta_k)^2$ with respect to a series of iterates $\boldsymbol{x}_k$, $\boldsymbol{y}_k$, $\boldsymbol{u}_k$, and $\boldsymbol{v}_k$ of Algorithm (3.1), $k = 1, 2, 3, \ldots$, where none of the iterates $\boldsymbol{x}_k$ is a stationary point. Moreover assume that*

a) there is a $\gamma \in \mathbb{R}$, such that $\gamma_k \leq \gamma < \infty$.

b) $h_j(\boldsymbol{x}_k) \geq -u_j^k/r$ if and only if $\overline{h}_j(\boldsymbol{y}_k) = 0$ for $j = 1, \ldots, m$, where $\boldsymbol{u}_k = (u_1^k, \ldots, u_m^k)^T$,

Then there is a $\delta_r > 0$, such that

$$\nabla \Phi_r(\boldsymbol{x}_k, \boldsymbol{u}_k)^T \begin{pmatrix} \boldsymbol{x}_k - \boldsymbol{y}_k \\ \boldsymbol{u}_k - \boldsymbol{v}_k \end{pmatrix} \geq \frac{\eta_k \delta_k^2}{4}$$

with $\delta_k \leq \delta_r$, where $r := \min\{10^j : j = 0, 1, 2, \ldots \text{ with } 10^j \geq 2\gamma/\eta\}$.

As a special consequence of the above theorem, it can be shown that the penalty parameters of the augmented Lagrangian remain bounded, see Zillober (1993a). Assumption a) of Theorem (3.4) guarantees a uniform convergence of both sequences $\{\boldsymbol{x}_k\}$ and $\{\boldsymbol{u}_k\}$, where b) requires that the same active constraints are identified by the convex subproblem (6) and the augmented Lagrangian (8).

From the above descent properties of the search directions generated by Algorithm (3.1), the following convergence theorem can be proved.

Theorem 3.5: *Let $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ be computed by Algorithm (3.1) satisfying the assumptions of Lemma (3.2) and Theorem (3.4). Then the algorithm either terminates at a stationary point, or every accumulation point of the iteration sequence is a stationary point for (6).*

It is possible to omit the additional assumptions of Theorem (3.5). In this case it is only possible to prove that at least one accumulation point is stationary, see Zillober (1993a).

An important assumption for the convergence analysis outlined above, is the solvability of the convex, separable subproblems, i.e. that these problems possess non-empty feasible regions. This cannot be ensured in advance and is sometimes not fulfilled, especially in the first iterations, when we are still far away from a solution. However there are various techniques to overcome this situation, see e.g. Fleury, Braibant (1986), Svanberg (1987) or Schittkowski (1983).

4 Numerical Results

The FE-analysis of the comparative study is performed by the software system MBB-LAGRANGE, see Kneppe, Krammer, and Winkler (1987) or Zotemantel (1993). MBB-LAGRANGE is a computer aided structural design system based on the finite element technique and mathematical programming. The optimization model is characterized by the design variables and different types of restrictions.

Design variables are element thicknesses, cross sections, concentrated masses, and fiber angles. Besides of isotropic, orthotropic, and anisotropic applications, the analysis and optimization of composite structures is among the most important features. The design can be restricted with respect to a static, dynamic or aeroelastic analysis. The general aim is to minimize the structural weight subject to some of the following constraints:

- displacements
- stresses
- strains
- buckling
- local compressive stresses
- aeroelastic efficiencies
- flutter speed
- natural frequencies
- dynamic responses
- eigenmodes
- weight
- bounds for the design variables (gages)

Gradients are evaluated either analytically or semi-analytically by a special sensitivity analysis.

Besides of the nine optimization algorithms included in the official version, two additional methods are added to the system, i.e. certain variants of convex approximation methods. The codes represent all major classes of algorithms that are in practical use at present. Most of the methods have been developed, implemented, and tested outside of the MBB-LAGRANGE environment, and are taken over from external authors.

By the subsequent comments, some additional features of the algorithms and special implementation details are to be outlined. To identify the optimization codes we use the notation of the MBB-LAGRANGE documentation.

SRM: The stress ratio code belongs to the class of optimality criteria methods and is motivated by statically determinated structures. The algorithm is applicable to problems with stress constraints only, consists of a simple update formula for the design variables, and does not need any gradient information.

IBF: The inverse barrier function method is an implementation of a penalty method which needs an feasible design to start the algorithm. The unconstrained minimization is performed with respect to a quasi-Newton update (BFGS) and an Hermite interpolation procedure for the line search. It is recommended to perform only a relatively small number of iterations, e.g. 5 or 10, and to start another cycle by increasing the penalty parameter through a constant factor.

MOM: Proceeding from the same unconstrained optimization routine as IBF, a sequential unconstrained minimization technique is applied. The method of multipliers uses an augmented Lagrangian function similar to (8) for the subproblem and the usual update rules for the multipliers. Both methods, i.e. IBF and MOM, have a special advantage when evaluating gradients of the objective function in the subproblem. The inverse of the stiffness matrix obtained by a decomposition technique, is multiplied only once with the remaining part of the gradient, not for each restriction as required for most of the subsequent methods.

SLP: The sequential linear programming method was implemented by Kneppe (1985). The linear subproblem is solved by a simplex method. So-called move limits are introduced to prevent cycling and iterates too far away from the feasible area. They are reduced in each iteration by the formula $\delta_{k+1} = \delta_k/(1 + \delta_k)$ and an additional cubic line search is performed as soon as cycling is observed.

RQP1: The first recursive or sequential quadratic programming code is subroutine NLPQL of Schittkowski (1985/86). Subproblems are solved by a dual algorithm based on a routine written by Powell (1983). The augmented Lagrangian function (8) serves as a merit function and BFGS-updates are used for the quasi-Newton formula. The special implementation of NLPQL is capable to solve also problems with very many constraints, see Schittkowski (1992), and is implemented in MBB-LAGRANGE in reverse communication. The idea is to save as much working memory as possible by writing optimization data on a file during the analysis, and by saving analysis data during an optimization cycle.

RQP2: This is the original sequential quadratic programming code VMCWD of Powell (1978) with an L_1-merit function. Also in this case, the BFGS-update is used internally together with a suitable modification of the penalty parameter.

GRG: The generalized reduced gradient code was implemented by Bremicker (1986). During the line search an extrapolation is performed to follow the boundary of active constraints closer. The Newton-algorithm for projecting non-feasible iterates during the line search onto the feasible domain, uses the derivative matrix for the very first step. Subsequently a rank-1-quasi-Newton formula of Broyden is updated.

QPRLT: To exploit the advantages of SQP and GRG methods, a hybrid method was implemented by Sömer (1987). Starting from a feasible design, a search direction is evaluated by the SQP-approach, i.e. by solving a quadratic programming subproblem. This direction is then divided in basic and non-basic variables, and a line search is performed very similar to the generalized reduced gradient method GRG.

CONLIN: This is the original implementation of Fleury (1989), where a convex and separable subproblem is generated as outlined in Section 2. In particular only variables belonging to negative partial derivatives, are inverted. The nonlinear subproblem is solved by a special dual method.

SCP: The sequential convex programming method was implemented by Zillober (1993b) and added to the MBB-LAGRANGE-system for the purpose of this comparative study. The algorithm uses moving asymptotes and a line search procedure for stabilization with respect to the merit function (8).

MMA: The code is a reimplementation of the original convex approximation method of Svanberg (1987) with moving asymptotes. As for CONLIN and SCP, the subproblems are solved by a special dual approach. The adaption of moving asymptotes is described in Zillober (1993b).

To conduct the numerical tests, 79 design problems have been collected. Most of them are *academic* ones, i.e. are more or less simple design problems found in the literature. The remaining ones possess some practical *real life* background from project work or are suitable modifications to act as benchmark test problems for the development of the software system. In all situations, we minimize the weight of a structure subject to displacement, stress, strain, buckling, dynamic and other constraints. Design variables are sizing variables, e.g. cross sectional areas and thicknesses of skeletal as well as membran or shell structures.

Since moreover all test examples are to be solvable by all available optimization algorithms, the dimension of the structures, i.e. number of elements and degrees of freedom, is relatively small compared to real life applications. More details about the test cases are found in Schittkowski, Zillober and Zotemantel (1993).

We believe that the present set of test cases is representative at least for small or medium size structural designs. It is also important to note that we do not want to test the analysis part of an FE-system. Instead the response of optimization routines when applied to solve structural optimization problems, is to be investigated.

All tests have been performed on a VAX 6000-510 running under VMS, at the Computing Center of the University of Bayreuth. The numerical codes are implemented in double precision FORTRAN. The intention behind our tests is to apply all optimization routines to all test examples that are available. To evaluate the results achieved, we need some information about the optimal solution, since the difference from the minimal weight of a test structure and the corresponding constraint violation serves as a measure for the accuracy of an actual iterate.

Thus we have to compute an optimal solution for each test case as accurate as possible. The most reliable codes were executed with a very small termination tolerance and a large number of iterations, until we got a stable and reliable solution. Test examples that did not lead to a clear solution point e.g. because of too many different local minimizers, have not been included in our set of test problems.

Having now an accepted reference value, it is possible to define whether an actual iterate $\boldsymbol{x}_k$ is sufficiently close to the optimal solution $\boldsymbol{x}^\star$ subject to a given tolerance $\epsilon > 0$ or not. For each function or gradient evaluation during a test run, we store the corresponding objective function value $h_0(\boldsymbol{x}_k)$ and the maximum constraint violation

$$r(\boldsymbol{x}_k) := \max\{\max(0, h_j(\boldsymbol{x}_k)) : j = 1, \ldots, m\}$$

together with some further data for analysis number and calculation time.

Now we are able to evaluate the performance of an algorithm subject to a given accuracy level ϵ. We sum up the performance criterion, e.g. calculation time or number of function and gradient evaluations, until for the first time the conditions

$$h_0(\boldsymbol{x}_k) \leq h_0(\boldsymbol{x}^\star)(1+\epsilon) \ , \quad r(\boldsymbol{x}_k) \leq \epsilon \tag{9}$$

are satisfied. We should note here that the constraint functions are scaled internally by the analysis procedure of MBB-LAGRANGE.

Moreover there are some reasonable upper bounds for the number of iterations, and we must be aware of the fact that there are situations where a code is unable to find at least one solution in a test problem class within the given accuracy level and the maximum number of iterations.

For the purpose of our comparative study, we evaluate the performance criteria

- calculation time in seconds
- number of function evaluations, where an evaluation of objective and all constraints is counted as one function call
- number of gradient evaluations, i.e. evaluation of gradient of objective function and of all active constraints, where active constraints are determined by the internal active set strategy of MBB-LAGRANGE

A complete description of the test procedure and the numerical results is found in Schittkowski, Zillober and Zotemantel (1993). For the purpose of this review paper, we present some numerical comparative results in form of quotients of mean values of two optimization algorithms, where the mean value is taken over the common subset of successfully solved problems. Since the numerical figures for the performance criterion calculation time differ drastically, we use the geometric mean in this case. Thus we get a matrix with the corresponding quotients, and their normalized row sums can be considered as weights or performance items.

Since we present only results for the complete test set without further restrictions, the optimization codes IBF and SRM are omitted in the subsequent tables. The first one requires a feasible initial design, and the second one is applicable only to problems with stress constraints.

For the accuracy levels $\epsilon = 0.01$ and $\epsilon = 0.0001$, the subsequent tables present the corresponding results. Table 1 and Table 5 show the number of test problems that could be solved successfully by two optimization algorithms, and the other ones show the corresponding mean values for the performance criteria calculation time, number of function evaluations and number of gradient evaluations, respectively.

	MOM	SLP	RQP1	RQP2	GRG	QPRLT	CONLIN	SCP	MMA
MOM	49	44	47	45	44	43	35	44	41
SLP	44	59	58	56	46	53	41	54	54
RQP1	47	58	66	60	50	60	44	57	57
RQP2	45	56	60	61	49	57	43	57	56
GRG	44	46	50	49	53	49	39	48	47
QPRLT	43	53	60	57	49	64	43	56	55
CONLIN	35	41	44	43	39	43	45	42	43
SCP	44	54	57	57	48	56	42	60	56
MMA	41	54	57	56	47	55	43	56	58

Table 1: Number of problems solved successfully by two optimization codes w.r.t. $\epsilon = 0.01$

	MOM	SLP	RQP1	RQP2	GRG	QPRLT	CONLIN	SCP	MMA	Weight
MOM	21.96 21.96	19.17 5.08	20.56 10.81	16.43 6.04	18.55 7.34	17.87 5.30	15.84 3.23	16.77 6.85	16.05 4.03	27.16
SLP	5.08 19.17	6.62 6.62	6.52 12.87	6.20 7.70	4.66 7.07	6.51 7.68	4.14 3.44	6.31 8.26	6.04 5.59	7.28
RQP1	10.81 20.56	12.87 6.52	17.38 17.38	13.25 8.68	9.92 7.57	16.08 9.51	10.63 4.12	12.72 9.07	12.16 6.53	14.15
RQP2	6.04 16.43	7.70 6.20	8.68 13.25	8.69 8.69	6.37 6.99	9.22 8.59	6.87 3.88	8.29 8.99	8.07 6.36	9.39
GRG	7.34 18.55	7.07 4.66	7.57 9.92	6.99 6.37	8.51 8.51	8.64 6.16	7.09 3.69	8.38 6.97	7.24 4.32	11.20
QPRLT	5.30 17.87	7.68 6.51	9.51 16.08	8.59 9.22	6.16 8.64	10.46 10.46	6.79 4.42	9.74 10.75	8.93 6.93	8.61
CONLIN	3.23 15.84	3.44 4.14	4.12 10.63	3.88 6.87	3.69 7.09	4.42 6.79	4.29 4.29	4.11 6.52	4.11 4.47	5.82
SCP	6.85 16.77	8.26 6.31	9.07 12.72	8.99 8.29	6.97 8.38	10.75 9.74	6.52 4.11	10.22 10.22	8.93 6.80	9.54
MMA	4.03 16.05	5.59 6.04	6.53 12.16	6.36 8.07	4.32 7.24	6.93 8.93	4.47 4.11	6.80 8.93	6.63 6.63	6.86

Table 2: Geometric mean values for calculation time over common sets of successfully solved problems w.r.t. $\epsilon = 0.01$

	MOM	SLP	RQP1	RQP2	GRG	QPRLT	CONLIN	SCP	MMA	Weight
MOM	98.06 98.06	103.02 9.59	101.04 17.64	100.98 12.56	99.82 29.41	103.30 16.74	108.46 5.09	95.05 22.55	100.78 6.54	43.72
SLP	9.59 103.02	10.29 10.29	10.16 15.95	10.41 12.29	9.46 46.43	10.53 37.70	7.66 6.24	9.96 15.91	9.74 6.70	3.66
RQP1	17.64 101.04	15.95 10.16	18.65 18.65	15.72 13.13	16.18 45.02	17.88 48.53	15.09 6.45	14.68 16.95	13.35 7.25	5.59
RQP2	12.56 100.98	12.29 10.41	13.13 15.72	13.08 13.08	12.78 45.00	13.46 39.39	11.81 6.37	11.54 17.18	10.91 7.16	4.49
GRG	29.41 99.82	46.43 9.46	45.02 16.18	45.00 12.78	45.21 45.21	47.45 24.94	41.85 6.13	48.29 21.40	46.15 6.94	17.34
QPRLT	16.74 103.30	37.70 10.53	48.53 17.88	39.39 13.46	24.94 47.45	46.94 46.94	32.47 6.81	41.18 20.07	41.27 7.29	13.45
CONLIN	5.09 108.46	6.24 7.66	6.45 15.09	6.37 11.81	6.13 41.85	6.81 32.47	6.69 6.69	6.74 17.50	6.60 6.58	2.63
SCP	22.55 95.05	15.91 9.96	16.95 14.68	17.18 11.54	21.40 48.29	20.07 41.18	17.50 6.74	19.60 19.60	15.55 7.39	6.39
MMA	6.54 100.78	6.70 9.74	7.25 13.35	7.16 10.91	6.94 46.15	7.29 41.27	6.58 6.60	7.39 15.55	7.29 7.29	2.73

Table 3: Arithmetic mean values for number of function evaluations over common sets of successfully solved problems w.r.t. $\epsilon = 0.01$

	MOM	SLP	RQP1	RQP2	GRG	QPRLT	CONLIN	SCP	MMA	Weight
MOM	121.71 121.71	120.59 17.23	124.26 22.64	118.00 18.40	112.27 6.73	117.63 5.30	125.49 8.23	116.05 20.59	122.15 11.12	55.67
SLP	17.23 120.59	18.61 18.61	18.34 22.10	18.86 18.96	16.96 9.04	19.09 8.98	13.37 10.54	17.96 17.70	17.52 11.44	6.62
RQP1	22.64 124.26	22.10 18.34	24.45 24.45	20.90 19.53	20.88 8.88	23.33 10.13	20.32 10.95	19.79 18.91	18.84 12.53	7.69
RQP2	18.40 118.00	18.96 18.86	19.53 20.90	19.51 19.51	18.98 8.86	19.96 9.05	18.14 10.79	17.75 18.95	17.43 12.36	7.05
GRG	6.73 112.27	9.04 16.96	8.88 20.88	8.86 18.98	8.77 8.77	9.12 6.47	8.26 10.31	9.25 20.50	8.81 11.91	3.62
QPRLT	5.30 117.63	8.98 19.09	10.13 23.33	9.05 19.96	6.47 9.12	9.80 9.80	7.26 11.67	9.32 20.25	9.20 12.62	3.02
CONLIN	8.23 125.49	10.54 13.37	10.95 20.32	10.79 18.14	10.31 8.26	11.67 7.26	11.42 11.42	11.52 18.67	11.26 11.21	4.59
SCP	20.59 116.05	17.70 17.96	18.91 19.79	18.95 17.75	20.50 9.25	20.25 9.32	18.67 11.52	20.00 20.00	17.71 12.82	7.11
MMA	11.12 122.15	11.44 17.52	12.53 18.84	12.36 17.43	11.91 8.81	12.62 9.20	11.21 11.26	12.82 17.71	12.62 12.62	4.64

Table 4: Arithmetic mean values for number of gradient evaluations over common sets of successfully solved problems w.r.t. $\epsilon = 0.01$

	MOM	SLP	RQP1	RQP2	GRG	QPRLT	CONLIN	SCP	MMA
MOM	19	16	17	15	15	10	13	15	14
SLP	16	48	46	46	36	37	32	43	44
RQP1	17	46	60	54	43	48	34	51	52
RQP2	15	46	54	57	43	46	35	52	53
GRG	15	36	43	43	48	42	33	40	41
QPRLT	10	37	48	46	42	55	31	43	44
CONLIN	13	32	34	35	33	31	37	35	35
SCP	15	43	51	52	40	43	35	54	51
MMA	14	44	52	53	41	44	35	51	54

Table 5: Number of problems solved successfully by two optimization codes w.r.t. $\epsilon = 0.0001$

	MOM	SLP	RQP1	RQP2	GRG	QPRLT	CONLIN	SCP	MMA	Weight
MOM	42.20 42.20	29.70 5.61	28.32 12.64	19.00 6.88	24.93 8.58	16.32 5.23	24.33 4.24	24.42 5.73	25.51 4.76	31.58
SLP	5.61 29.70	5.48 5.48	5.25 9.80	5.08 5.67	4.82 6.49	5.69 5.74	4.64 3.54	4.99 6.32	4.84 4.40	7.30
RQP1	12.64 28.32	9.80 5.25	15.35 15.35	12.17 7.43	8.99 6.89	13.83 8.76	8.04 3.37	11.77 8.19	11.67 6.23	13.07
RQP2	6.88 19.00	5.67 5.08	7.43 12.17	8.42 8.42	5.94 7.35	7.50 7.74	5.69 3.44	7.46 8.33	7.41 6.25	8.31
GRG	8.58 24.93	6.49 4.82	6.89 8.99	7.35 5.94	9.50 9.50	9.98 7.01	8.23 3.89	7.93 6.37	6.90 4.36	10.68
QPRLT	5.23 16.32	5.74 5.69	8.76 13.83	7.74 7.50	7.01 9.98	11.33 11.33	6.23 3.97	9.09 9.14	7.76 6.48	8.17
CONLIN	4.24 24.33	3.54 4.64	3.37 8.04	3.44 5.69	3.89 8.23	3.97 6.23	4.19 4.19	4.24 7.76	3.84 4.28	5.33
SCP	5.73 24.42	6.32 4.99	8.19 11.77	8.33 7.46	6.37 7.93	9.14 9.09	7.76 4.24	9.79 9.79	8.97 6.40	9.03
MMA	4.76 25.51	4.40 4.84	6.23 11.67	6.25 7.41	4.36 6.90	6.48 7.76	4.28 3.84	6.40 8.97	6.69 6.69	6.54

Table 6: Geometric mean values for calculation time over common sets of successfully solved problems w.r.t. $\epsilon = 0.0001$

	MOM	SLP	RQP1	RQP2	GRG	QPRLT	CONLIN	SCP	MMA	Weight
MOM	131.63	139.00	137.35	145.00	133.27	99.00	157.77	146.07	148.57	47.76
	131.63	9.69	20.29	17.40	39.73	12.00	8.54	15.13	8.36	
SLP	9.69	13.42	13.33	13.74	13.11	14.62	11.78	14.05	13.70	4.22
	139.00	13.42	16.00	12.22	41.33	29.86	7.41	18.88	8.52	
RQP1	20.29	16.00	19.52	15.98	17.56	18.90	14.85	14.73	14.56	4.59
	137.35	13.33	19.52	13.57	49.00	56.65	7.35	19.80	9.88	
RQP2	17.40	12.22	13.57	13.88	12.65	12.54	12.74	13.52	12.94	3.88
	145.00	13.74	15.98	13.88	50.49	42.80	7.29	19.92	9.81	
GRG	39.73	41.33	49.00	50.49	55.44	56.76	55.91	59.65	47.93	15.71
	133.27	13.11	17.56	12.65	55.44	37.07	7.06	20.70	8.93	
QPRLT	12.00	29.86	56.65	42.80	37.07	60.31	35.65	55.37	43.84	12.72
	99.00	14.62	18.90	12.54	56.76	60.31	6.19	20.09	9.34	
CONLIN	8.54	7.41	7.35	7.29	7.06	6.19	7.35	7.49	7.17	2.35
	157.77	11.78	14.85	12.74	55.91	35.65	7.35	22.97	7.57	
SCP	15.13	18.88	19.80	19.92	20.70	20.09	22.97	20.76	19.27	6.01
	146.07	14.05	14.73	13.52	59.65	55.37	7.49	20.76	9.49	
MMA	8.36	8.52	9.88	9.81	8.93	9.34	7.57	9.49	9.81	2.75
	148.57	13.70	14.56	12.94	47.93	43.84	7.17	19.27	9.81	

Table 7: Arithmetic mean values for number of function evaluations over common sets of successfully solved problems w.r.t. $\epsilon = 0.0001$

	MOM	SLP	RQP1	RQP2	GRG	QPRLT	CONLIN	SCP	MMA	Weight
MOM	237.21	240.94	242.29	246.60	219.40	91.70	277.62	249.00	247.93	63.20
	237.21	17.44	29.12	24.87	8.60	4.50	15.15	20.47	14.79	
SLP	17.44	24.85	24.67	25.50	24.25	27.27	21.59	26.12	25.43	7.25
	240.94	24.85	23.15	19.76	8.14	7.81	12.84	21.74	15.07	
RQP1	29.12	23.15	27.42	22.91	23.51	25.73	20.50	21.78	21.67	5.75
	242.29	24.67	27.42	21.50	9.37	12.27	12.74	23.39	17.79	
RQP2	24.87	19.76	21.50	22.30	19.84	20.07	19.91	21.52	21.26	5.23
	246.60	25.50	22.91	22.30	9.74	10.63	12.60	23.44	17.64	
GRG	8.60	8.14	9.37	9.74	10.25	10.33	9.91	10.68	9.24	2.71
	219.40	24.25	23.51	19.84	10.25	8.10	12.15	22.77	15.88	
QPRLT	4.50	7.81	12.27	10.63	8.10	12.29	7.65	12.26	10.80	2.53
	91.70	27.27	25.73	20.07	10.33	12.29	10.42	22.81	16.70	
CONLIN	15.15	12.84	12.74	12.60	12.15	10.42	12.73	13.00	12.37	3.49
	277.62	21.59	20.50	19.91	9.91	7.65	12.73	24.14	13.17	
SCP	20.47	21.74	23.39	23.44	22.77	22.81	24.14	24.39	22.88	5.65
	249.00	26.12	21.78	21.52	10.68	12.26	13.00	24.39	17.00	
MMA	14.79	15.07	17.79	17.64	15.88	16.70	13.17	17.00	17.65	4.19
	247.93	25.43	21.67	21.26	9.24	10.80	12.37	22.88	17.65	

Table 8: Arithmetic mean values for number of gradient evaluations over common sets of successfully solved problems w.r.t. $\epsilon = 0.0001$

Conclusions

The paper describes a class of methods for nonlinear programming, which is applicable only in certain situations, where inverse variables in constraints dominate. The algorithm generates a sequence of convex, separable nonlinear subproblems, which must be solved in each iteration. It is shown how the algorithm can be stabilized by a line search with an augmented Lagrangian merit function. Some theoretical convergence results are mentioned.

Also the numerical results of a comparative study of optimization routines are included. The 79 test problems are taken from structural design optimization, where the FE-analysis is performed by the software system MBB-LAGRANGE. The results indicate, that the sequential convex programming method belongs not only to the most reliable approach, but is also more efficient than *classical* nonlinear programming algorithms, e.g. sequential quadratic programming.

References

Bremicker, M. (1986): *Entwicklung eines Optimierungsalgorithmus der generalisierten reduzierten Gradienten.* Report, Forschungslaboratorium für angewandte Strukturoptimierung, Universität-GH Siegen

Fleury, C. (1989): *An efficient dual optimizer based on convex approximation concepts.* Structural Optimization, Vol. 1, 81 - 89

Fleury, C.; Braibant, V. (1986): *Structural Optimization – a new dual method using mixed variables.* International Journal for Numerical Methods in Engineering, Vol. 23, 409 - 428

Hörnlein, H.; Schittkowski, K. (eds.) (1992): *Software systems for structural optimization.* Birkhäuser Verlag, Basel, ISNM 110

Kneppe, G. (1985): *Direkte Lösungsstrategien zur Gestaltsoptimierung von Flächentragwerken.* Reihe 1, No. 135, VDI-Verlag, Düsseldorf

Kneppe, G.; Krammer, J.; Winkler, F. (1987): *Structural Optimization of large scale problems using MBB LAGRANGE.* 5th World Congress and Exhibition on FEM, Salzburg, Austria

Nguyen, V.H.; Strodiot, J.J.; Fleury, C. (1987): *A mathematical convergence analysis for the convex linearization method for engineering design optimization.* Engineering Optimization, Vol. 11, 195 - 216

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

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

Schittkowski, K. (1981): *The Nonlinear Programming method of Wilson, Han*

and Powell with an augmented Lagrangian type line search function. Numerische Mathematik, Vol. 38, 83 - 114

Schittkowski, K. (1983): *On the convergence of a Sequential Quadratic Programming method with an augmented Lagrangian line search function.* Optimization, Vol. 14, 197 - 216

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

Schittkowski, K. (1992): *Solving nonlinear programming problems with very many constraints.* Optimization, Vol. 25, 179-196

Schittkowski, K.; Zillober, Ch.; Zotemantel, R. (1993): *Numerical Comparison of Nonlinear Programming Algorithms for Structural Optimization.* To appear: Structural Optimization

Sömer, M. (1987): *Aufstellen und Testen eines hybriden SQP-GRG Optimierungsalgorithmus.* Studienarbeit, Institut für Mechanik und Regelungstechnik, Universität-GHS Siegen, Siegen

Svanberg, K. (1987): *The Method of Moving Asymptotes – a new method for Structural Optimization.* International Journal for Numerical Methods in Engineering, Vol. 24, 359 - 373

Svanberg, K. (1993): *The Method of Moving Asymptotes (MMA) with some extensions.* In: Rozvany, G. (ed.) Lecture Notes for the NATO/DFG ASI "Optimization of Large Structural Systems", Vol. 1, 55 - 66, Berchtesgaden, Germany

Zillober, Ch. (1992): *Eine global konvergente Methode zur Lösung von Problemen aus der Strukturoptimierung.* Dissertation, Technische Universität München

Zillober, Ch. (1993a): *A globally convergent version of the Method of Moving Asymptotes.* To appear: Structural Optimization

Zillober, Ch. (1993b): *SCP - An implementation of a sequential convex programming algorithm for nonlinear programming.* DFG–Report No. , Schwerpunkt Anwendungsbezogene Optimierung und Steuerung

Zotemantel, R. (1993): *MBB-LAGRANGE: A computer aided structural design system.* in: Software Systems for Structural Optimization, H. Hörnlein, K. Schittkowski eds., Birkhäuser, Basel, Boston, Berlin