Struct Multidisc Optim 24, 362–371 © Springer-Verlag 2002
Digital Object Identifier (DOI) 10.1007/s00158-002-0248-5

SCPIP – an efficient software tool for the solution of structural optimization problems

C. Zillober

Abstract This paper describes SCPIP, a FORTRAN77 subroutine that has been proven to be a reliable implementation of convex programming methods in an industrial environment. Convex approximation methods like the method of moving asymptotes are used nowadays in many software packages for structural optimization. They are known to be efficient tools for the solution of design problems, in particular if displacement dependent constraints like stresses occur. A major advantage over many but not all classical approaches of mathematical programming is that at an iteration point a local model is formulated. For the solution of such a model no further function and gradient evaluations are necessary besides those at the current iteration point. The first versions of convex approximation methods used all a dual approach to solve the subproblems which is still a very efficient algorithm to solve problems with at most a medium number of constraints. But it is not efficient for problems with many constraints. An alternative is the use of an interior point method for the subproblem solution. This leads to more freedom in the definition of the linear systems where most of the computing time to solve the subproblems is spent. In consequence, large-scale problems can be handled more efficiently.

Key words convex approximation methods, method of moving asymptotes, sequential convex programming, structural optimization, large-scale optimization

1 Introduction

SCPIP (**s**equential **c**onvex **p**rogramming with **i**nterior **p**oint method) is an implementation of the convex approximation methods MMA and SCP. MMA (method of moving asymptotes; Svanberg (1987)) and its globally convergent extension SCP (sequential convex programming; Zillober (2001b)) are known to work well for structural optimization problems. One reason is that displacement dependent constraints are approximated very well. Another reason is that at the current iteration point a local model is used that does not require further function and gradient evaluations of the original problem besides that at the iteration point itself. This is very important since evaluations of the original problem are usually time consuming finite element analyses. Classical approaches like augmented Lagrangian methods or penalty approaches need additional evaluations to solve their subproblems. Other methods like sequential quadratic programming use also only local models. For a survey of nonlinear programming methods see e.g. Nocedal and Wright (1999). The local model of MMA and SCP itself is convex such that efficient subproblem solution methods are available.

We consider the following mathematical programming problem:

$$\min \quad f(x)\,, \quad x \in \mathbb{R}^n\,,$$

$$\text{s.t.} \quad g_j(x) = 0\,, \quad j = 1, \dots, m_{eq}\,,$$

$$h_j(x) \le 0\,, \quad j = 1, \dots, m_{ie}\,,$$

$$\underline{x_i} \le x_i \le \overline{x_i}\,, \quad i = 1, \dots, n\,. \tag{OP}$$

The functions f, g_j ($j = 1, \dots, m_{eq}$) and h_j ($j = 1, \dots, m_{ie}$) have to be defined only on $X := \{x \mid \underline{x_i} \le x_i \le \overline{x_i},\, i = 1, \dots, n\}$ and are assumed to be continuous on X and at least twice continuously differentiable in the interior of X. The feasible region is assumed to be nonempty.

(OP) will be approximated by convex subproblems, i.e. the objective function will be replaced by a strictly convex approximation, inequality constraints will be approximated by convex functions, equalities will be linearized and the box-constraints will be allowed to shrink in the local model with respect to the original box-constraints.

Received May 18, 2001
Revised manuscript received April 24, 2002

C. Zillober

Mathematisches Institut, Universität Bayreuth, 95440 Bayreuth, Germany
e-mail: christian.zillober@uni-bayreuth.de

Although equality constraints are not frequently apparent in structural optimization models we keep them in the formulation in order to show that they can also be handled efficiently by the software. This is important since subsequently we will give arguments for at least a trial to incorporate the equilibrium equations as equality constraints in the optimization model. Presently, this cannot be accomplished in the author's software environment due to limited access on the finite element software.

We will present details of the interior point subproblem solution method, a major part of SCPIP. Compared with the classical dual approach, the interior point method provides more flexibility in the formulation of the subproblem. The main computational work to solve the subproblems has to be done in the solution of a large sequence of linear systems. In the dual approach these systems are of dimension $m \times m$, where $m := m_{eq} + m_{ie}$ is the total number of constraints besides the box-constraints. These systems are dense independent of any structure in the problem. The interior point method, however, provides the possibility to proceed by either solving $m \times m$ systems or $n \times n$ systems which is obviously an advantage for problems with many constraints but few variables which is often the case for sizing problems. Moreover, sparsity of the Jacobian of the constraints can be exploited. If the model has this property then larger problems can be solved. This paper does not provide a detailed description of the theory because this is already published in Zillober (2001a,b).

The outline of the paper is as follows. In Sect. 2 we introduce the principles of the methods MMA and SCP. In Sect. 3 the interior point approach for the subproblem solution is presented. The topic of infeasible subproblems is discussed in Sect. 4. Implementational details are presented in Sect. 5, followed by examples in Sect. 6. Finally, conclusions are given in the last section.

2 MMA and SCP

Since problem (OP) is usually too hard to solve it directly a common strategy of all modern nonlinear programming methods is to replace (OP) by a sequence of easier to solve subproblems. The way how these subproblems are defined is the major difference of the methods. The background of convex approximation methods is the observation that in special cases displacement dependent constraints are exact linearizations of the original functions of the inverse variables. Thus, after a variable transformation to inverse variables a problem with linear constraints is obtained which is usually easier to solve. For practical cases the functions do not behave this way. However, a nearly reciprocal behavior could be observed. MMA/SCP perform a linearization with respect to transformed variables $\frac{1}{x_i - L_i^k}$ and $\frac{1}{U_i^k - x_i}$, respectively, leading to convex approximation functions. The parameters L_i^k and U_i^k have to be chosen such that $L_i^k < x_i < U_i^k$. They are asymptotes of the approximation explaining the name of MMA.

In the $(k+1)$-st iteration the objective function f is approximated at the point x^k by

$$f^k(x) := \alpha_{0,0}^k + \sum_{i \in I_{0,+}^k} \frac{\alpha_{0,i}^k(x_i)}{U_i^k - x_i} - \sum_{i \in I_{0,-}^k} \frac{\alpha_{0,i}^k(x_i)}{x_i - L_i^k} \,. \qquad (1)$$

An inequality constraint h_j $(j = 1, \dots, m_{ie})$ is replaced by

$$h_j^k(x) := \alpha_{j,0}^k + \sum_{i \in I_{j,+}^k} \frac{\alpha_{j,i}^k}{U_i^k - x_i} - \sum_{i \in I_{j,-}^k} \frac{\alpha_{j,i}^k}{x_i - L_i^k} \,. \qquad (2)$$

Equality constraints are linearized in the usual sense

$$g_j^k(x) := g_j\left(x^k\right) + \sum_{i=1}^{n} \frac{\partial g_j\left(x^k\right)}{\partial x_i}\left(x_i - x_i^k\right) \,. \qquad (3)$$

Defining $h_0 := f$ the index sets are defined as

$$I_{j,+}^k := \left\{ i \in \{1, \dots, n\} : \frac{\partial h_j\left(x^k\right)}{\partial x_i} \geq 0 \right\} \text{ and}$$

$$I_{j,-}^k := \left\{ i \in \{1, \dots, n\} : \frac{\partial h_j\left(x^k\right)}{\partial x_i} < 0 \right\} \text{ for all}$$

$j = 0, \dots, m_{ie}$.

The constants $\alpha_{j,0}^k$ are defined as $(j = 0, \dots, m_{ie})$:

$$\alpha_{j,0}^k := h_j\left(x^k\right) - \sum_{i \in I_{j,+}^k} \frac{\partial h_j\left(x^k\right)}{\partial x_i}\left(U_i^k - x_i^k\right) + \sum_{i \in I_{j,-}^k} \frac{\partial h_j\left(x^k\right)}{\partial x_i}\left(x_i^k - L_i^k\right) .$$

The constants $\alpha_{j,i}^k$ of the inequalities are defined as $(j = 1, \dots, m_{ie})$:

$$\alpha_{j,i}^k := \begin{cases} \frac{\partial h_j(x^k)}{\partial x_i}\left(U_i^k - x_i^k\right)^2 , & \text{if } i \in I_{j,+}^k \\ \frac{\partial h_j(x^k)}{\partial x_i}\left(x_i^k - L_i^k\right)^2 , & \text{if } i \in I_{j,-}^k \end{cases} \qquad (4)$$

To ensure strict convexity of the approximation of the objective we have to introduce additional positive parameters τ_i with respect to (4):

$$\alpha_{0,i}^k(x_i) := \begin{cases} \frac{\partial f(x^k)}{\partial x_i}\left(U_i^k - x_i^k\right)^2 + \tau_i\left(x_i - x_i^k\right)^2 , & \text{if } i \in I_{0,+}^k \\ \frac{\partial f(x^k)}{\partial x_i}\left(x_i^k - L_i^k\right)^2 - \tau_i\left(x_i - x_i^k\right)^2 , & \text{if } i \in I_{0,-}^k \end{cases} \qquad (5)$$

Strict convexity of the approximation of the objective is necessary to fulfill an assumption concerning the convergence proof. However, it is also helpful in practice to avoid slow convergence. The parameters τ_i will be discussed in Sect. 5.6 in more detail.

The approximations f^k and h_j^k are defined on

$$D^k := \left\{x : L_i^k < x_i < U_i^k,\ i = 1, \ldots, n\right\}.$$

L_i^k and U_i^k are parameters ("moving asymptotes") to be chosen with $L_i^k < x_i^k < U_i^k$, $i = 1, \ldots, n$. In Sect. 5.1 we will discuss a practical scheme to choose the asymptotes.

The approximations have the following properties:

- f^k, g_j^k and h_j^k are first order approximations at x^k.
- g_j^k are linear.
- h_j^k are convex, i.e. they can be strictly convex but also linear.
- f^k is strictly convex.
- f^k, g_j^k and h_j^k are separable.

The $(k+1)$-st subproblem is then

$$\min \quad f^k(x)\,, \quad x \in \mathbb{R}^n\,,$$

$$\text{s.t.} \quad g_j^k(x) = 0\,, \quad j = 1, \ldots, m_{eq}\,,$$

$$h_j^k(x) \le 0\,, \quad j = 1, \ldots, m_{ie}\,,$$

$$\underline{x_i}' \le x_i \le \overline{x_i}'\,, \quad i = 1, \ldots, n\,, \qquad (\mathrm{SP}^k)$$

where $\underline{x_i}' := \max\{\underline{x_i}, x_i^k - \omega(x_i^k - L_i^k)\}$ and $\overline{x_i}' := \min\{\overline{x_i}, x_i^k + \omega(U_i^k - x_i^k)\}$, ω can be between 0 and 1. In SCPIP it is chosen as $\omega = 0.9$. Its role is to keep the variables away from the poles L_i^k and U_i^k and to avoid the computation of too large function and gradient values of the approximations.

Due to the convexity properties a problem (SP^k) has always a unique solution provided its feasible region is nonempty. In general this cannot be guaranteed but there are remedies by enlarging the feasible region with artificial variables. This will be discussed in Sect. 4.

We will now formulate the MMA-algorithm.

Algorithm 1. MMA (method of moving asymptotes)

```
Step 0: Choose $x^0 \in X$, $y^0_{eq,j}$ $(j = 1, \ldots, m_{eq})$, $y^0_{ie,j} \ge 0$
        $(j = 1, \ldots, m_{ie})$; compute $f(x^0), \nabla f(x^0), g_j(x^0)$,
        $\nabla g_j(x^0)$ $(j = 1, \ldots, m_{eq}), h_j(x^0), \nabla h_j(x^0)$
        $(j = 1, \ldots, m_{ie})$; let $k := 0$
Step 1: Compute $L_i^k$ and $U_i^k$ $(i = 1, \ldots, n)$ by some
        scheme; define $f^k(x)$, $g_j^k(x)$ $(j = 1, \ldots, m_{eq})$,
        $h_j^k(x)$, $(j = 1, \ldots, m_{ie})$ (cf. (1),(2) and (3))
Step 2: Solve (SP$^k$) let $(x^{k+1}, y_{eq}^{k+1}, y_{ie}^{k+1})$ be the
        solution, where $y_{eq}^{k+1}$ and $y_{ie}^{k+1}$ denote the
        corresponding vector of Lagrange multipliers
Step 3: If $x^{k+1} = x^k$ stop; $(x^k, y_{eq}^k, y_{ie}^k)$ is the solution
Step 4: Compute $f(x^{k+1})$, $\nabla f(x^{k+1})$, $g_j(x^{k+1})$,
        $\nabla g_j(x^{k+1})$ $(j = 1, \ldots, m_{eq}), h_j(x^{k+1})$, $\nabla h_j(x^{k+1})$
        $(j = 1, \ldots, m_{ie})$, let $k := k + 1$, goto step 1
```

The SCP-method differs from MMA mainly by an additional line-search with respect to the augmented Lagrangian function. With this modification it is possible to prove global convergence of the algorithm, cf. Zillober (2001b). Since MMA is always our default method we neglect the detailed presentation of SCP and refer the reader to Zillober (2001b,a).

3
Subproblem solution

To simplify the notation we neglect equality constraints in this section and rewrite problem (SP^k). It should be stressed that the subproblem solvers described in this section are not restricted to inequality constraints only. The new subproblem is then

$$\min \quad f^k(x)\,, \quad x \in \mathbb{R}^n\,,$$

$$\text{s.t.} \quad h_j^k(x) \le 0\,, \quad j = 1, \ldots, m\,,$$

$$\underline{x_i}' \le x_i \le \overline{x_i}'\,, \quad i = 1, \ldots, n\,. \qquad (6)$$

m is used for the number of constraints since we do not have to distinguish between equalities and inequalities. We define $X' := \left\{x : \underline{x_i}' \le x_i \le \overline{x_i}',\ i = 1, \ldots, n\right\}$.

The subproblems of MMA have been solved traditionally with a dual approach. For general nonlinear programming problems the solution of the dual problem is not easier than the solution of the primal problem (6), even harder. In the case of MMA, however, two properties of the approximation functions lead to the fact that the dual problem is much easier. Firstly, the convexity property yields that the solution of the dual directly corresponds to the solution of the primal. In general this provides only a bound. Secondly, the separability property leads to a much more easier evaluation of the dual objective function. In general, an evaluation of the dual objective is equivalent to the n-dimensional minimization of an unconstrained function or a function with respect to simple bound constraints. Here, this n-dimensional minimization splits into n one-dimensional minimizations which can be performed analytically. For details of this approach see Fleury (1989) or Svanberg (1987).

The main computational work in the solution of the dual problem has to be done finally by solution of a sequence of linear systems of the dimension $m \times m$. These linear systems are dense, independent of any structure in problem (OP) or (SP^k). This is not critical for problems with a moderate number of constraints. But for problems with a large number of constraints this leads to a computational overhead. For this situation there was a need to find an alternative. This has been found in the predictor-corrector interior point method which will not only cover the case of a large number of constraints but also efficiently solves subproblems with few constraints and maybe many variables. Moreover, structure in problem (OP) can be exploited.

We will now briefly describe the predictor-corrector interior point method. Details of the approach are described by Zillober (2001a).

First of all, nonnegative slacks are added wherever inequalities appear

$$\min \quad f^k(x)\,, \quad x \in \mathbb{R}^n\,,$$

$$\text{s.t.} \quad h_j^k(x) + c_j = 0\,, \quad j = 1, \dots, m\,, \tag{7}$$

$$-c_j + r_j = 0\,, \quad j = 1, \dots, m\,, \tag{8}$$

$$\underline{x_i}' - x_i + s_i = 0\,, \quad i = 1, \dots, n\,, \tag{9}$$

$$x_i - \overline{x_i}' + t_i = 0\,, \quad i = 1, \dots, n\,, \tag{10}$$

$$r, s, t \geq 0\,.$$

In the interior point algorithm we will have to ensure the strict positivity of the variables r, s and t and their duals. In consequence, their introduction allows the corresponding constraints to coincide with their bounds, i.e. $c_j = 0, x_i = \underline{x_i}'$ and $x_i = \overline{x_i}'$ are possible. The corresponding constraints may even be violated in intermediate iterations. However, we have to ensure additionally that x does not leave the region of the definition of f^k and h_j^k, i.e. $x \in X'$ has to be guaranteed.

For this modified problem we put the nonnegativity constraints corresponding to r, s and t into the objective function using a barrier formulation. Then we formulate the Kuhn–Tucker conditions:

$$\begin{aligned}
&\nabla f^k(x) + J(x)y - d_s + d_t = 0\\
&h^k(x) + c = 0\\
&y - d_r = 0\\
&d_r - \mu R^{-1}e = 0\\
&d_s - \mu S^{-1}e = 0\\
&d_t - \mu T^{-1}e = 0\\
&-c + r = 0\\
&\underline{x}' - x + s = 0\\
&x - \overline{x}' + t = 0
\end{aligned}$$

Here, y denotes the dual to (7), d_r is the dual of (8), d_s is the dual of (9) and d_t is the dual of (10). $e = (1, \dots, 1)^T$ is a vector of ones in the appropriate dimension and $\mu > 0$ denotes the barrier parameter. $J(x) = (\nabla h^k(x))^T$ is the transposed Jacobian of the constraints of (6). The capital letters denote diagonal matrices built by the components of the vectors, e.g. $R^{-1} := \text{diag}(1/r_1, \dots, 1/r_m)$.

We apply Newton's method to a set of equations derived from the Kuhn–Tucker conditions by a few trivial algebraic manipulations. For the right hand side, in the predictor step the terms containing the barrier parameter μ are neglected. We end up with

$$\begin{pmatrix}
\nabla_{xx}L^k(x,y) & J(x) & & & & & & -I & I\\
J^T(x) & & I & & & & & & \\
 & I & & & & & -I & & \\
 & & & D_r & & & R & & \\
 & & & & D_s & & & S & \\
 & & & & & D_t & & & T\\
 & & -I & I & & & & & \\
-I & & & & I & & & & \\
I & & & & & I & & &
\end{pmatrix}
\begin{pmatrix}\Delta x\\ \Delta y\\ \Delta c\\ \Delta r\\ \Delta s\\ \Delta t\\ \Delta d_r\\ \Delta d_s\\ \Delta d_t\end{pmatrix} =$$

$$-\begin{pmatrix}
\nabla f^k(x) + J(x)y - d_s + d_t\\
h^k(x) + c\\
y - d_r\\
D_r Re\\
D_s Se\\
D_t Te\\
-c + r\\
\underline{x}' - x + s\\
x - \overline{x}' + t
\end{pmatrix}. \tag{11}$$

$\nabla_{xx}L^k(x,y)$ is the Hessian of the Lagrangian with respect to x. This matrix is only dependent on x and y.

The matrix of (11) is hard to handle but since most of its terms are positive diagonal matrices we can eliminate Δc, Δr, Δs, Δt, Δd_r, Δd_s and Δd_t.

This elimination process provides analytical, easy to evaluate formulae for the eliminated components and the following linear system that remains to be solved

$$\begin{pmatrix}
\nabla_{xx}L^k(x,y) + S^{-1}D_s + T^{-1}D_t & J(x)\\
J^T(x) & -D_r^{-1}R
\end{pmatrix}
\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} =$$

$$\begin{pmatrix} b_1\\ b_2\end{pmatrix} \tag{12}$$

The evaluation of the right hand side is straightforward. Formulae are omitted because the structure of the matrix of (12) is the main issue to be discussed below.

To understand the structure of this linear system we focus on its input terms in more detail. Firstly, we observe that $\nabla_{xx}L^k(x,y)$ is a positive diagonal matrix

$$\nabla_{xx}L^k(x,y) = \nabla^2 f^k(x) + \sum_{j=1}^{m} y_j \nabla^2 h_j^k(x)\,.$$

The individual Hessians are all diagonal because of the separability property. The Hessians of the constraints are nonnegative because of the convexity, the Hessian of the objective is positive because of its special construction, cf. (5). If the algorithm is able to ensure that the corresponding Lagrange multipliers are nonnegative then $\nabla_{xx}L^k(x,y)$ is positive.

Secondly, we consider the nonzero structure of $J(x)$. For one particular component of the approximation (2) of a constraint function we have

$$\frac{\partial h_j^k(x)}{\partial x_i} = \begin{cases} \frac{\partial h_j(x^k)}{\partial x_i} \frac{\left(U_i^k - x_i^k\right)^2}{\left(U_i^k - x_i\right)^2}, & \text{if } i \in I_{j,+}^k \\ \frac{\partial h_j(x^k)}{\partial x_i} \frac{\left(x_i^k - L_i^k\right)^2}{\left(x_i - L_i^k\right)^2}, & \text{if } i \in I_{j,-}^k \end{cases} . \tag{13}$$

Since the second term is strictly positive in both cases, the nonzero structure of $J(x)$ at an arbitrary point is identical to the nonzero structure of the original constraints at the current main iteration point x^k. That means, sparsity in the original problem is preserved for the subproblems.

The matrix of (12) is indefinite, of full rank and can be considered as sparse since the upper left and lower right part are diagonal. Additional sparsity of $J(x)$ improves the sparsity.

It is possible to use this system to define the search directions for the interior point method. It can be a practical alternative for certain sparsity patterns of $J(x)$. But the more important cases follow subsequently.

We may observe that due to the special structure of (12), the diagonal matrices in the upper left and lower right part, we can easily eliminate either $\varDelta x$ or $\varDelta y$ from (12). In the first case the results is:

$$\left(J^T(x)\varTheta^{-1}J(x) + D_r^{-1}R\right)\varDelta y = b_3 . \tag{14}$$

In the second case we get:

$$\left(\varTheta + J(x)R^{-1}D_rJ^T(x)\right)\varDelta x = b_4 . \tag{15}$$

$\varTheta$ abbreviates the upper left part of (12):

$$\varTheta := \nabla_{xx}L^k(x,y) + S^{-1}D_s + T^{-1}D_t .$$

In both cases the eliminated variable is easy to compute as soon as the linear system (14) ((15) respectively) is solved.

The matrices in (14) and (15) are both positive definite. The matrix in (14) is of dimension $m \times m$, in (15) it is of dimension $n \times n$. This makes one of the major advantages of the interior point approach evident: to compute a search direction which is the major part of a subproblem solution it is possible to define either linear systems of dimension $m \times m$ or $n \times n$. The formulations are equivalent in the sense that all necessary data can be derived of the corresponding linear system solution. While problems with a large number of constraints where hardly to handle with the traditional dual subproblem solver, the interior point approach is suitable for both situations, problems with many variables and few constraints as well as for problems with few variables and many constraints. Moreover, since the sparsity of the Jacobian of the original constraints can be exploited, it is also possible to solve problems with many variables and many constraints as long as the problem has such a structure.

In Sect. 5.4 it is discussed how SCPIP handles the choice of (14) and (15) and the methods to solve the linear systems.

The corrector step of the interior point method uses the same matrices to define the search direction. Only the right-hand side changes. The results of the predictor step are used for its definition.

4 Infeasible subproblems

As mentioned in Sect. 2 we have to consider the case where the feasible region of a subproblem (SP^k) is empty. This can be the case even if the feasible region of (OP) is non-empty. Since the constraints of (SP^k) are of first order it is only possible if the current iteration point is infeasible. Most likely it happens if the infeasibility is large. To overcome this situation we introduce artificial variables which guarantee the existence of a feasible point. Since they are not related to the original problem we try to keep their influence as low as possible. In the best case they should vanish if they are not necessary, i.e. if the feasible region of (SP^k) is not empty. This can be obtained by adding these variables to the objective function. To accelerate the process we multiply them additionally by a penalty parameter.

Although this seems to be a purely heuristical approach theoretical convergence properties of SCP can be preserved in this case under mild regularity conditions. For problems without equality constraints this has been outlined in Zillober (1993).

Suppose w.l.o.g. that we have $h_j^k\left(x^k\right) > 0, j = 1, \ldots, m_1$ and $h_j^k\left(x^k\right) \le 0, j = m_1, \ldots, m$. We now introduce new (artificial) variables q_j $(j = 1, \ldots, m_1)$ and transform problem (6) in the following way:

$$\min \quad f^k(x) + \frac{1}{2}\sum_{j=1}^{m_1} \rho_j q_j^2 , \quad x \in \mathbb{R}^n ,$$

$$\text{s.t.} \quad h_j^k(x) - q_j h_j^k\left(x^k\right) \le 0 , \quad j = 1, \ldots, m_1 ,$$

$$h_j^k(x) \le 0 , \quad j = m_1, \ldots, m ,$$

$$\underline{x_i}' \le x_i \le \overline{x_i}' , \quad i = 1, \ldots, n ,$$

$$0 \le q_j \le M , \quad j = 1, \ldots, m_1 . \tag{16}$$

With this transformation the point (x^k, q^k) with $q_j^k = 1$, $j = 1, \ldots, m_1$ is a feasible point (provided the box-constraints are fulfilled). The penalty parameters ρ_j are positive parameters. We choose initially $\rho_j = 1$, $j = 1, \ldots, m_1$. For degenerate problems it is possible that the ρ_j have to be enlarged during the iteration, cf.

Zillober (1993). If this is necessary we multiply them by 10. The additional term in the objective function will force the artificial variables to be as low as possible. The constant M could be set to 1 to ensure at least one feasible point. For numerical reasons (ensuring interior points also for the artificial variables) we choose $M = 2$.

It is important to notice that the basic structure of (16) is the same as that of (6). The differentiability does not change, the feasible region is compact due to the box-constraints of the artificial variables and the separability property is also still given. Therefore, the interior point method introduced in Sect. 3 can be applied to problem (16) with only minor modifications.

It is important to notice that this procedure is done automatically within SCPIP such that a user does not have to be concerned about the problem of infeasible subproblems.

5 Implementational details

5.1 Choice of asymptotes

SCPIP contains several approaches to adapt the asymptotes L_i^k and U_i^k. We present here only one of these possibilities. It is generally applicable and has been proven to be a reliable strategy. For the other strategies see Zillober (2001c).

$$\begin{aligned} k = 0,1\ :\ L_i^k &= x_i^k - \gamma_1(\overline{x}_i - \underline{x}_i) \\ U_i^k &= x_i^k + \gamma_1(\overline{x}_i - \underline{x}_i) \end{aligned}$$

$$k = 2,3,\ldots : \text{If } \operatorname{sign}\left(x_i^k - x_i^{k-1}\right) = \operatorname{sign}\left(x_i^{k-1} - x_i^{k-2}\right)\ :$$
$$L_i^k = x_i^k - \gamma_2\left(x_i^{k-1} - L_i^{k-1}\right)$$
$$U_i^k = x_i^k + \gamma_2\left(U_i^{k-1} - x_i^{k-1}\right)$$
$$\text{If } \operatorname{sign}\left(x_i^k - x_i^{k-1}\right) \neq \operatorname{sign}\left(x_i^{k-1} - x_i^{k-2}\right)\ :$$
$$L_i^k = x_i^k - \gamma_3\left(x_i^{k-1} - L_i^{k-1}\right)$$
$$U_i^k = x_i^k + \gamma_3\left(U_i^{k-1} - x_i^{k-1}\right)$$

A suitable choice is $\gamma_1 = 0.5,\ \gamma_2 = 1.15,\ \gamma_3 = 0.7$.

5.2 Initialization

For Algorithm 1 we have to initialize the design variables x^0 and the Lagrange multipliers y_{eq}^0 and y_{ie}^0. For the design variables we expect input by the user. We only test this input whether it fulfills the box-constraints ($x^0 \in X$). If this is not the case we increase (or decrease, respectively) the corresponding components such that the violated box-constraints are fulfilled exactly.

For the Lagrange multipliers we do not expect initial values set by the user. To the experience of the author the initialization of the Lagrange multipliers is not crucial for the performance of MMA or SCP. Therefore, we set $y_{eq,j}^0 := 0\ \forall\, j = 1,\ldots,m_{eq}$ and $y_{ie,j}^0 := 0\ \forall\, j = 1,\ldots,m_{ie}$. However, by setting a certain input variable it is also possible to provide initial values for these two vectors.

We have to be more careful for the initialization of the interior point algorithm since it is more sensitive to certain initial values, in particular the values of the variables that have to fulfill the interior point condition (positivity condition). But due to the introduction of the variables r, s, t and their corresponding duals d_r, d_s and d_t we shifted the problem of the interior point initialization from the original variables appearing in Algorithm 1 to artificial variables that are not seen by a user. For example, variables s and d_s are initialized as $s_i := \max\{x_i - \underline{x_i}', \beta\}$ and $(d_s)_i := \beta$; β is an internal constant. Since a scaling procedure is applied to the subproblems, $\beta := 1$ is a well working value that is not problem dependent. More details on the initialization of the interior point algorithm can be found in Zillober (2001a).

5.3 Stopping criteria

Within SCPIP there are two possibilities to define stopping criteria. The first one is based on the mathematical optimality conditions for problem (OP). It is the simultaneous fulfilling of $\|\nabla_x L(x,y)\|_{\max} \le \epsilon$, $|g_j(x)| \le \epsilon$, $j = 1,\ldots,m_{eq}$ and $h_j(x) \le \epsilon$, $j = 1,\ldots,m_{ie}$; L denotes the Lagrangian of (OP). Notice that the box-constraints are always fulfilled due to the construction of the algorithm; ϵ is a constant that has to be set by the user.

The second possibility to define a stopping criterion is a relaxed convergence check based on the iteration progress. Four conditions have to be fulfilled simultaneously.

1. $|g_j(x)| \le \epsilon,\ j = 1,\ldots,m_{eq}$ and $h_j(x) \le \epsilon,\ j = 1,\ldots,m_{ie}$. This is the feasibility condition as in the first possibility.
2. $\left\|\frac{x^k - x^{k-1}}{x^k}\right\|_{\max} \le \epsilon_1$. The relative change in the design variables.
3. $\left|f\left(x^k\right) - f(x^{k-1})\right| \le \epsilon_2$. The absolute change in the objective function.
4. $\frac{\left|f\left(x^k\right) - f(x^{k-1})\right|}{\left|f\left(x^k\right)\right|} \le \epsilon_3$. The relative change in the objective function.

The values for $\epsilon, \epsilon_1, \epsilon_2$ and ϵ_3 can be set independently.

If a user wants to stop the algorithm whenever the progress in the objective function is less than one percent while attaining feasibility he could choose e.g. $\epsilon := 10^{-7}, \epsilon_3 := 0.01, \epsilon_1 :=$inf, $\epsilon_2 :=$inf, where inf is a large number representing infinity. This would lead to the fact that criteria 2 and 3 are neglected.

5.4 Choice of linear system solution

To cover the variety of possible large-scale cases SCPIP provides several possibilities to solve the linear systems arising in (14) and (15):

- A dense Cholesky solver
- A sparse Cholesky solver
- A conjugate gradient solver

Together, there are six possibilities to define and solve the linear systems. As a rough orientation we summarize in which situation which combination of the definition of the linear system and the corresponding linear system solver should be chosen.

Formulation (14) should be chosen together with the subsequent linear systems solvers in the following cases:

Cholesky, dense	$m < n$, m moderate or m large, matrix in (14) is dense and sufficient storage is available
Cholesky, sparse	$m < n$, m large, matrix in (14) is sparse
Conjugate gradients	$m < n$, m large and available storage is not sufficient for a decomposition

Similarly, formulation (15) should be chosen together with the subsequent linear systems solvers in the following cases:

Cholesky, dense	$n < m$, n moderate or n large, matrix in (15) is dense and sufficient storage is available
Cholesky, sparse	$n < m$, n large, matrix in (15) is sparse
Conjugate gradients	$n < m$, n large and available storage is not sufficient for a decomposition

The meaning of the term *moderate* depends on the underlying hardware and what is accepted by the user for the time of a subproblem solution in relation to a function and/or gradient evaluation of (OP).

Each of these situations can be chosen by the user of SCPIP by setting two integer parameters. But it is also possible to set the parameters such that the program decides itself which combination to choose. This is dependent on the relation of m to n, possible sparsity in the matrices of (14) and (15) and the available storage for a possible decomposition of the matrices. It should be mentioned that the case where sparsity of $J(x)$ is given and m as well as n are large is not explored enough up to now to claim that the decision rules of SCPIP are good in this case. Sophisticated preconditioning techniques for the conjugate gradient solver are also still in work.

In most finite element software systems that provide optimization tools (as well as in the author's environment, cf. Sect. 6) the equilibrium conditions are solved in the finite element part, i.e. their results are input for the optimization routine. In consequence, the nodal displacements u are always computed in direct dependence of the design variables x. The Jacobian matrix of the nodal displacements and the displacement dependent functions (e.g. stresses) is then in general a dense matrix because any nodal displacement is dependent on the sizes of all elements.

If the equilibrium conditions would be incorporated to the optimization model as equality constraints, u and x could be handled as independent variables. This approach is known in the literature as *simultaneous analysis and design.* The Jacobians of all displacement dependent constraints would then be sparse matrices. Although the optimization problem dimensions would increase we could solve larger problems since the sparsity can be handled very efficiently by SCPIP. We would like to formulate these arguments in order to convince finite element software developers about the advantages and to be able to proof this in the near future in a nonacademic environment. The data structures of SCPIP are already designed for sparse problems, i.e. SCPIP expects the Jacobian of the constraints in a sparse storage format.

A drawback of this approach would be the fact that intermediate designs can violate the equilibrium conditions because the optimizer does not necessarily fulfill all the constraints in intermediate iterations.

5.5 Active-set strategy

SCPIP contains also a simple active-set strategy. Gradients are requested only for those inequality constraints that fulfill the condition $h_j\left(x^k\right) \geq -\text{ACTRES}$, where ACTRES is a positive number to be set by the user. Only these constraints are passed to the subproblem which can lead to a considerable reduction of the number of constraints in a subproblem. Choosing a large value for ACTRES avoids the active-set strategy.

5.6 Strict convexity of f^k

In Sect. 2 the approximation of the objective function was defined in (1) and the choice of the $\alpha_{0,i}^k(x_i), i = 1, \ldots, n$ in (5). Without the usage of the term containing the τ_i, the second derivative with respect to a component $i \in I_{0,+}^k$ is

$$\frac{\partial^2 f^k(x)}{\partial x_i^2} = 2\frac{\partial f\left(x^k\right)}{\partial x_i}\frac{\left(U_i^k - x_i^k\right)^2}{\left(U_i^k - x_i\right)^3}\,. \tag{17}$$

This term is strictly positive in D^k provided the first derivative $\frac{\partial f\left(x^k\right)}{\partial x_i}$ is positive. If this is the case we do not need the additional term containing τ_i. $f^k(x)$ is strictly convex in X' which is sufficient for the proof of global convergence, cf. Zillober (2001b). In general this cannot be guaranteed, i.e. we have to consider $\frac{\partial f(x^k)}{\partial x_i} = 0$. Then the second derivative term as above vanishes. Therefore we add a term that is independent of $\frac{\partial f(x^k)}{\partial x_i}$ to ensure the positivity of the second derivative of $f^k(x)$ in X' without destroying the first-order approximation property. This is achieved by the term $\tau_i(x_i - x_i^k)^2$ as used in (5). (17) then changes to

$$\frac{\partial^2 f^k(x)}{\partial x_i^2} = 2\left(\frac{\partial f\left(x^k\right)}{\partial x_i} + \tau_i\right) \frac{\left(U_i^k - x_i^k\right)^2}{\left(U_i^k - x_i\right)^3} .$$

For the practical implementation we have a stronger demand. We ask for

$$\frac{\partial f\left(x^k\right)}{\partial x_i} + \tau_i \geq \overline{\tau} > 0$$

to ensure a certain degree of convexity for the approximation of the objective.

The same principle is valid for the components $i \in I_{0,-}^k$.

6 Examples

We present three examples in order to show the advantages of the interior point approach in SCPIP. A sizing example with few variables and many constraints, an example of topology design in its standard formulation with many variables and few constraints and an example of topology design in a different formulation with many variables and many constraints.

More examples can be found in Zillober (2001a). An industrial project with applications of SCPIP can be found in Zillober and Vogel (2000) (ship design).

The examples have been computed on a INTEL Pentium PC with 450 MHz and 128 MB RAM.

6.1 Example 1

The first example has been computed with the shell optimization program POPT which is an add-on module to the wide-spread finite element program ANSYS. It should be noted that due to limited access on the source code the times reported below do not include the time for the start-up of the problem (formulation of stiffness matrix etc.) and the function and gradient evaluation at the initial design. More details of this example and the corresponding computations can be found in Zillober (2001a).

The example is a tube construction. It is fixed at the right border. There are two load cases. The first load case are forces applying at the nodes on the upper end of the tube directed against the curvature of the tube. These forces are indicated in Fig. 1 by many small arrows, one per element node. The tips of these arrows form the shifted circle at the top of the tube. The second load case is a pressure force from inside the tube applying at each element.

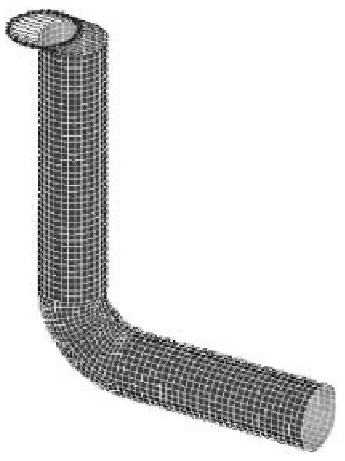
Fig. 1 Example 1: tube

The tube is meshed with 2976 finite elements and 17856 degrees of freedom. The structure is divided in 36 areas where each area is assigned a thickness parameter. The objective function is the weight of the structure, the constraints are one stress constraint with upper and lower bound for each element, i.e. 11 904 constraints for the two load cases.

The time for one function and gradient evaluation was approximately 500 CPU-seconds. SCPIP solves this problem within 5336 CPU-seconds and 9 function and gradient evaluations besides the evaluations at the initial design. SCPIP chose automatically approach (15) and the dense Cholesky solver. Manual selection of approach (14) as well as the selection of the dual subproblem solver lead to no results for the first subproblem within one night of real time. The active set strategy lead to about 6000 constraints in the first subproblem.

6.2 Example 2

The second example is a problem of topology design based on the power-law approach, the so-called MBB-halfbeam taken from Sigmund (2001). For that purpose the Matlab code of Sigmund (2001) has been translated to Fortran77. The ground structure has been meshed with 390×260 two-dimensional square 4-node finite elements. It is fixed at the left edge in x-direction and the lower right node in y-direction. A force is applied at the upper left node. The objective is to minimize the compliance of the structure with respect to a volume constraint and equilibrium. During the optimization process a mesh-independency filter like in Sigmund (2001) is used. The formulation of the problem is as follows:

$$\min_x \quad u^T K(x) u\,,$$

$$\text{s.t.} \quad V(x) \le V_{\max}\,,$$

$$K(x)u = p\,,$$

$$0 < x_{\min} \le x_i \le 1\,, \qquad i = 1,\ldots,n_e\,; \tag{18}$$

x_i is the relative density in element i, thus x is the vector of design variables. n_e is the number of finite elements in the discretization. K is the global stiffness matrix of the power-law approach, V is the volume of the structure, u is the global displacement vector and p is the force vector. $x_{\min}$ is a positive, but very low lower bound on the relative densities to avoid singularities. We chose $x_{\min} = 0.001$ and $V_{\max}$ as 50% of the volume of the ground structure.

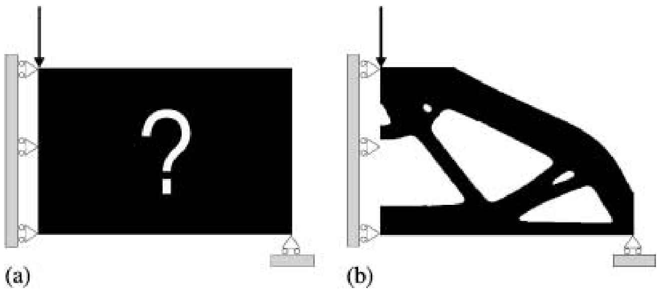

Fig. 2 Example 2. (a) topology design, ground structure, (b) topology design, result

Thus, the optimization problem has 101 400 variables and one constraint besides the box-constraints since the equilibrium condition is solved within a finite element analysis in an outer loop.

SCPIP solved the problem within 31 iterations and 138 000 CPU-seconds. Stopping criterion was an upper bound of 0.1% on the relative change in the objective function. SCPIP chose automatically approach (14). Approach (15) lead to no comparable results. In this example the traditional dual approach is also efficient. It should be emphasized that the largest part of the CPU-time refers to the computation of the mesh-independency filter, not to the finite element analyses or the subproblem solutions. This should be an encouragement to look for alternative strategies for mesh independent solutions. The structure of the solution is also sensitive to the size of the filter radius. Moreover the fine discretization should be viewed as a component of a numerical study for large scale structural optimization problems. It is not necessary for each topology optimization problem to use such a fine discretization.

6.3 Example 3

The third example is also an example of topology design but it is based on a slightly different formulation of the volume constraint and subject to constraints on the element compliances. The structure is fixed in x-direction at the left edge and in y-direction at the lower edge. Moreover, there are damping layers of fixed material in the upper and lower horizontal direction (indicated by grey). The design domain is the domain between these fixed layers. Forces are distributed equally at the upper edge in negative y-direction. The discretization is 270×180, i.e. 48 600 finite elements are used.

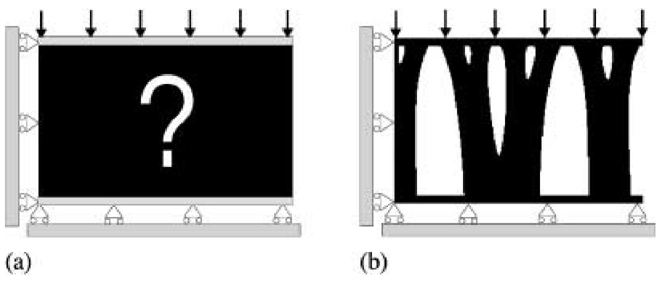

Fig. 3 Example 3. (a) topology design, ground structure, (b) topology design, result

The volume constraint is modified such that the amount of material used in the final structure is variable. To make such a formulation useful it is necessary to penalize the use of material in the objective function. The reason for this modification of the volume constraint is the introduction of constraints on the element compliances in order to ensure feasible designs. Thus, it is possible to use more material to get feasible designs but it is also possible to reduce the amount of material if the constraints are not active.

The new problem formulation reads as follows:

$$\min_x \quad u^T K(x) u + \rho\delta^2\,,$$

$$\text{s.t.} \quad V(x) \le \delta V_0\,,$$

$$K(x)u = p\,,$$

$$u_i^T K_i(x) u_i \le c_i\,, \quad i = 1,\ldots,n_e\,,$$

$$0 \le \delta \le 1\,,$$

$$0 < x_{\min} \le x_i \le 1\,, \qquad i = 1,\ldots,n_e\,; \tag{19}$$

V_0 is the amount of material in the ground structure, δ is the new optimization variable for a flexible use of material, ρ is a fixed penalty parameter to make δ as low as possible; u_i is the displacement vector corresponding to element i, K_i is the element stiffness matrix of element i, c_i is the bound for the i-th element compliance.

In our example the c_i are equal for all elements such that we can expect that few of these constraints will be active. Again, the equilibrium conditions are solved separately such that the optimization problem has 48 601 variables and 48 601 constraints besides the box-constraints.

SCPIP solves this problem in 41 iterations and 43 350 CPU-seconds to the stopping criterion as above. At the optimum 4 of the local constraints are active. During the iteration process up to 8 local constraints have been active. The optimal value of δ at the optimum is 0.527, that means 52.7% of the initial material of the design domain is used. This value is sensitive to the choice of ρ which has been set to 10^4 in this case. In all iterations approach (14) has been chosen for the solution of the subproblems.

7 Conclusion

The Fortran code SCPIP for the solution of structural optimization problems has been introduced. Its advantages in the variable formulation of subproblems have been shown. SCPIP is suitable for the solution of large optimization problems which is demonstrated by three examples. Moreover, structure in the problem data as sparsity can be exploited. Future developments are to fix decision rules in case of sparse Jacobians of constraints and the development of sophisticated preconditioning techniques for the conjugate gradient linear system solver. Both in order to be able to solve problems that are still larger than the problems handled presently.

Acknowledgements The author is grateful to CAD-FEM GmbH, Marktplatz 2, D-85567 Grafing, for the possibility to use the POPT program for finite element computations, to Frank Vogel, inuTech GmbH, Fürther Straße 212, D-90429 Nürnberg, for providing structural optimization examples and to Ole Sigmund, Department of Solid Mechanics, Technical University of Denmark, DK-2800 Lyngby, for providing his topology optimization code and examples.

References

Fleury, C. 1989: CONLIN: an efficient dual optimizer based on convex approximation concepts. *Struct. Optim.* **1**, 81–89

Nocedal, J.; Wright, S. 1999: *Numerical optimization.* Berlin, Heidelberg, New York: Springer

Sigmund, O. 2001: A 99 line topology optimization code written in Matlab. *Struct. Multidisc. Optim.* **21**, 120–127

Svanberg, K. 1987: The method of moving asymptotes – a new method for structural optimization. *Int. J. Num. Meth. Eng.* **24**, 359–373

Zillober, C. 1993: A globally convergent version of the method of moving asymptotes. *Struct. Optim.* **6**, 166–174

Zillober, C. 2001a: A combined convex approximation – interior point approach for large scale nonlinear programming. *Optim. Engng.* **2**, 51–73

Zillober, C. 2001b: Global convergence of a nonlinear programming method using convex approximations. *Numer. Algorithms* **27**, 256–289

Zillober, C. 2001c: Software manual for SCPIP 2.2. *Technical Report TR01-2*, Informatik, Universität Bayreuth, WWW: www.uni-bayreuth.de/departments/math/~czillober/papers/tr01-2.ps

Zillober, C.; Vogel, F. 2000: Solving large scale structural optimization problems. In: Sienz, J. (ed.) *Proc. 2-nd ASMO UK/ISSMO Conf. on Engineering Design Optimization*, pp. 273–280. University of Swansea, Wales