AIAA JOURNAL
Vol. 47, No. 11, November 2009

Globally Convergent Optimization Algorithm Using Conservative Convex Separable Diagonal Quadratic Approximations

Albert A. Groenwold* and Derren W. Wood†
University of Stellenbosch, Stellenbosch 7602, South Africa
and
L. F. P. Etman‡ and Simon Tosserams§
Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands

DOI: 10.2514/1.41975

We implement and test a globally convergent sequential approximate optimization algorithm based on (convexified) diagonal quadratic approximations. The algorithm resides in the class of globally convergent optimization methods based on conservative convex separable approximations developed by Svanberg. At the start of each outer iteration, the initial curvatures of the diagonal quadratic approximations are estimated using historic objective and/or constraint function value information, or by building the diagonal quadratic approximation to the reciprocal approximation at the current iterate. During inner iterations, these curvatures are increased if no feasible descent step can be made. Although this conditional enforcement of conservatism on the subproblems is a relaxation of the strict conservatism enforced by Svanberg, global convergence is still inherited from the conservative convex separable approximations framework developed by Svanberg. A numerical comparison with the globally convergent version of the method of moving asymptotes and the nonconservative variants of both our algorithm and method of moving asymptotes is made.

Nomenclature

c_2	=	curvature
f_j	=	constraint function j, $j > 0$
$\tilde{f}_j$	=	approximate constraint function j, $j > 0$
f_0	=	objective function
$\tilde{f}_0$	=	approximate objective function
k	=	outer iteration number
$\mathcal{L}$	=	Lagrangian
l	=	inner iteration number
m	=	number of constraints
n	=	number of design variables
P_D	=	dual subproblem
P_P	=	primal subproblem
$\mathcal{R}$	=	real number
$\boldsymbol{x} \in \mathcal{R}^n$	=	primal (design) variables
$\check{x}_i$	=	lower bound of design variable i
$\hat{x}_i$	=	upper bound of design variable i
α	=	index; runs over 0 to m
β	=	intermediate result
γ	=	dual
ϵ	=	tolerance
$\boldsymbol{\lambda} \in \mathcal{R}^m$	=	dual variables
χ	=	multiplication constant

Received 5 November 2008; revision received 29 June 2009; accepted for publication 29 June 2009.

*Professor, Department of Mechanical and Mechatronics Engineering; albertg@sun.ac.za.

†Ph.D. Student, Department of Mechanical and Mechatronics Engineering; wderren@gmail.com.

‡Assistant Professor, Department of Mechanical Engineering, Post Office Box 513; l.f.p.etman@tue.nl. Member AIAA.

§Ph.D. Student, Department of Mechanical Engineering, Post Office Box 513; s.tosserams@tue.nl.

I. Introduction

IN STRUCTURAL optimization, sequential approximate optimization (SAO) methods are the optimization methods of choice, in particular when computationally demanding analysis models are part of the optimization loop. These methods typically use a tailored approximation with nonlinear curvature for not only the objective function, but each and every constraint. The specific SAO algorithm used is often selected purely as a result of experience with that algorithm for a specific class of programming problems, irrespective of the existence of proof of global convergence for that algorithm.

Well-known globally convergent algorithms from mathematical programming are sequential quadratic programming (SQP) algorithms. These methods are very efficient (where efficiency is expressed in terms of the number of function and gradient evaluations required for convergence and termination). The efficiency, however, derives from the use of second-order information, which is often unattractive in structural optimization, because second-order information requires the storage of the, in general, fully populated Hessian matrix. This is highly demanding of resources for the very large number of design variables often present in structural optimization.

To avoid the storage of coupling terms, most of the approximations popular in structural optimization are based on separable subproblems, for example, see the review paper by Barthelemy and Haftka [1]. In addition, several of these algorithms are also based on *convex* subproblems. (This is certainly the case for algorithms based on dual principles, because the combination of separability and convexity often allows for the formulation of highly efficient dual formulations.)

Proof of global convergence has never been demonstrated for many an implementation of popular engineering SAO algorithms. Examples include the very popular method of moving asymptotes (MMA) of Svanberg [2] (which is currently almost exclusively used in large topology optimization problems when multiple constraints are present), the well-known CONLIN algorithm of Fleury and Braibant [3], and the Dynamic-Q method of Snyman and Hay [4]. What is more, even optimality criterion methods (e.g., see Berke and Venkayya [5], Khot [6], Venkayya [7], and Rozvany [8]) lack a general proof of global convergence.

However, the foregoing does not mean that these algorithms cannot be rendered globally convergent. Two obvious strategies may be followed. The first is to cast these algorithms in a mathematical

trust-region framework, for example, see Conn et al. [9]. Alternatively, the algorithms may be cast into the framework of conservative convex and separable approximations (CCSA) developed by Svanberg [10]. The variable conservatism of the approximations takes over the role of the trust region to effect global convergence; it ensures that the optimal solution of the subproblem is a feasible solution of the original problem, with a lower objective value than the previous iterate. CCSA are aimed at nonlinear inequality constrained problems; the notion of conservatism does not exist for equality constraints, except if these constraints are linear.

The MMA approximation was the first separable approximation to be rendered globally convergent by casting the approximation into the CCSA framework [10,11]. (Herein, we will denote the resulting algorithm by GCMMA.) Yet, for several engineering problems of importance, for example, the topology optimization problem, the nonconservative variant of MMA is often preferred despite the risk of convergence failure. This may, in part, be attributed to the additional computational burden sometimes required when enforcing global convergence. Moreover, for the topology optimization applications, the nonconservative variant has proven to often terminate at good local minima (the problem is highly multimodal).

To contribute to the understanding and further development of globally convergent CCSA optimization algorithms, we herein cast the convex form of diagonal quadratic approximations into a dual SAO algorithm within Svanberg's CCSA framework [10]. This is probably the simplest variant in the CCSA class one can envision. The conservatism is effected through manipulation of the coefficients of the second-order diagonal terms. Nonconvex forms of these approximations were previously proposed by Snyman and Hay [4] and Groenwold et al. [12,13]. The diagonal quadratic approximation does not depend on moving asymptotes to effect conservatism as in MMA; no intervening variables are introduced.

A similar approximation to the diagonal quadratic approximations used herein was mentioned by Svanberg [10], which, in turn, may be viewed as a convexified instance of the approximations used by Snyman and Hay [4]. However, Svanberg [10] presented no numerical results for this variant within the CCSA framework. What is more, we herein use different methods to estimate the initial curvatures in each outer iteration, and we only enforce conservatism in the inner iteration loop *if a feasible descent step cannot be made*. The algorithm nevertheless inherits the global convergence proofs from the CCSA class of algorithms developed by Svanberg [10].

Our paper is arranged as follows. In Sec. II, we introduce the nonlinear programming problem that interests us in this paper. This is followed by a short introduction to sequential approximate optimization in Sec. III, where we introduce the diagonal quadratic approximations under consideration and the dual form used in our subproblems. In Sec. IV, we present the CCSA algorithm based on diagonal quadratic approximations, followed by numerical experiments in Sec. V. Finally, concluding remarks are offered in Sec. VI.

II. Problem Statement

Consider a nonlinear inequality constrained optimization problem P_{NLP} of the form

$$\begin{aligned} &\min f_0(\boldsymbol{x}) \\ &\text{subject to } f_j(\boldsymbol{x}) \le 0, \qquad j = 1, 2, \ldots, m \\ &\qquad \check{x}_i \le x_i \le \hat{x}_i, \quad i = 1, 2, \ldots, n \end{aligned} \tag{1}$$

where $f_0(\boldsymbol{x})$ is a real valued scalar objective function, and $f_j(\boldsymbol{x})$, $j = 1, 2, \ldots, m$ are m inequality constraint functions. Functions of $f_0(\boldsymbol{x})$ and $f_j(\boldsymbol{x})$ depend on the n real (design) variables $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}^T \in \mathcal{R}^n$, with $\check{x}_i$ and $\hat{x}_i$, respectively, indicating lower and upper bounds on variable x_i.

We restrict ourselves to nonlinear programming problems, viz., we assume that at least one of the functions $f_\alpha(\boldsymbol{x})$, $\alpha = 0, 1, 2, \ldots, m$ is nonlinear (although the remaining functions may be linear). The functions $f_\alpha(\boldsymbol{x})$, $\alpha = 0, 1, 2, \ldots, m$ are required to be (at least) once continuously differentiable. Because we cast the convex diagonal SAO algorithm into the CCSA framework, we will follow Svanberg's procedure to relax problem P_{NLP} to ensure that a feasible solution to this problem always exists [2]. This is briefly discussed in the Appendix. For the sake of brevity and clarity, however, we do not explicitly mention the penalty terms, except where their mentioning is necessary to prevent confusion.

III. Sequential Approximate Optimization

Sequential approximate optimization as a solution strategy for problem P_{NLP} seeks to construct successive approximate analytical subproblems $P[k]$, $k = 1, 2, 3, \ldots$ at successive approximations $\boldsymbol{x}^{\{k\}}$ to the solution $\boldsymbol{x}^*$. The solution of subproblem $P[k]$ is $\boldsymbol{x}^{\{k*\}} \in \mathcal{R}^n$, to be obtained using any suitable continuous programming method. Thereafter, $\boldsymbol{x}^{\{k+1\}} = \boldsymbol{x}^{\{k*\}}$, the minimizer of subproblem $P[k]$.

A. Approximate Primal Subproblem for Problem P_{NLP}

The approximate continuous primal subproblem $P_P[k]$ at $\boldsymbol{x}^{\{k\}}$ is written as

Primal approximate subproblem $P_P[k]$

$$\begin{aligned} &\min \tilde{f}_0(\boldsymbol{x}) \\ &\text{subject to } \tilde{f}_j(\boldsymbol{x}) \le 0, \qquad j = 1, 2, \ldots, m \\ &\qquad \check{x}_i \le x_i \le \hat{x}_i, \quad i = 1, 2, \ldots, n \end{aligned} \tag{2}$$

This primal problem has n unknowns, m constraints, and $2n$ side or bound constraints (when no slack or relaxation variables are introduced); it may be solved using many techniques for constrained nonlinear programming. Here, we understand the functions $\tilde{f}_\alpha(\boldsymbol{x})$ to represent the approximations at the point $\boldsymbol{x}^k$ to the original functions $f_\alpha(\boldsymbol{x})$ for $\alpha = 0, 1, \ldots, m$; the additional terms resulting from relaxation (see the Appendix) are simply added to primal approximate subproblem $P_P[k]$. For $\alpha = 0$, we understand that the objective function is approximated; for $1 \le \alpha \le m$, inequality constraint function j is approximated, which we will also refer to as the constraint functions f_j. It is assumed here that first-order conditions $\tilde{f}_\alpha(\mathbf{x}_k) = f_\alpha(\mathbf{x}_k)$ and $(\partial \tilde{f}_\alpha / \partial x_i)(\mathbf{x}_k) = (\partial f_\alpha / \partial x_i)(\mathbf{x}_k)$ are satisfied.

B. Diagonal Quadratic Approximation

The diagonal quadratic approximation we have previously presented [12] is expressed as

$$\begin{aligned} \tilde{f}_\alpha(\boldsymbol{x}) = f_\alpha(\boldsymbol{x}^{\{k\}}) &+ \sum_{i=1}^{n} \left(\frac{\partial f_\alpha}{\partial x_i} \right)^{\{k\}} (x_i - x_i^{\{k\}}) \\ &+ \frac{1}{2} \sum_{i=1}^{n} c_{2i_\alpha}^{\{k\}} (x_i - x_i^{\{k\}})^2 \end{aligned} \tag{3}$$

It is also understood that

$$\left(\frac{\partial f}{\partial x_i} \right)^{\{k\}} = \frac{\partial f}{\partial x_i}(\boldsymbol{x}^{\{k\}})$$

being the partial derivative of f with respect to x_i at the point $\boldsymbol{x}^{\{k\}}$. Approximation (3) is convex if $c_{2i_\alpha}^{\{k\}} \ge 0 \ \forall \ i$, whereas the approximation is strictly convex if the inequality holds for all i.

Some strategies for estimating the curvatures $c_{2i_\alpha}^{\{k\}}$ may be found in [12,14,15]. Herein, for the sake of brevity, we will consider only two. The first results in a spherical quadratic approximation, whereas the second yields the (separable) nonspherical diagonal quadratic approximation to the reciprocal approximation.

1. *Spherical Quadratic Approximation*

To construct a spherical quadratic approximation [4], we select $c_{2i_\alpha}^{\{k\}} \equiv c_{2_\alpha}^{\{k\}} \ \forall \ i$, which requires the determination of the single unknown $c_{2_\alpha}^{\{k\}}$, to be obtained by enforcing the condition

$$\tilde{f}_{\alpha}(\boldsymbol{x}^{\{k-1\}}) = f_{\alpha}(\boldsymbol{x}^{\{k-1\}}) \tag{4}$$

which implies that

$$c_{2_\alpha}^{\{k\}} = \frac{2[f_{\alpha}(\boldsymbol{x}^{\{k-1\}}) - f_{\alpha}(\boldsymbol{x}^{\{k\}}) - \nabla^T f_{\alpha}(\boldsymbol{x}^{\{k\}})(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})]}{\|\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}\|_2^2} \tag{5}$$

This results in the approximation proposed by Snyman and Hay [4]. An alternative condition for formulating a spherical quadratic approximation is presented in [16].

2. Quadratic Approximation to the Reciprocal Approximation

We will now derive a nonspherical diagonal quadratic approximation with n unknowns $c_{2i_0}^{\{k\}}$. (Herein, we will do so for the objective function f_0 only, but generalization to the constraints f_j is perfectly in order.)

Let us depart with the (separable) reciprocal approximation

$$\tilde{f}_R(\boldsymbol{x}) = f(\boldsymbol{x}^{\{k\}}) + \sum_{i=1}^{n}\left[1 - \left(\frac{x_i}{x_i^{\{k\}}}\right)^{-1}\right] x_i^{\{k\}} \left(\frac{\partial f}{\partial x_i}\right)^{\{k\}} \tag{6}$$

where we have, for the sake of convenience dropped, subscript zero. Taking the derivatives of the reciprocal approximation (6) with respect to (w.r.t.) the variables x_i, we, respectively, obtain the first and second derivatives of $\tilde{f}_R$ in the current point $\boldsymbol{x}^k$ as

$$\left(\frac{\partial \tilde{f}_R}{\partial x_i}\right)^{\{k\}} = \left(\frac{\partial f}{\partial x_i}\right)^{\{k\}} \tag{7}$$

and

$$\left(\frac{\partial^2 \tilde{f}_R}{\partial x_i^2}\right)^{\{k\}} = \frac{-2}{x_i^{\{k\}}}\left(\frac{\partial f}{\partial x_i}\right)^{\{k\}} \tag{8}$$

Hence, the quadratic Taylor series expansion to $\tilde{f}_R$ in the current point $\boldsymbol{x}^k$ is precisely the diagonal quadratic approximation (3), if we select the higher-order curvatures $c_{2i}^{\{k\}}$ in approximation (3) as Eq. (8), that is, we enforce

$$c_{2i}^{\{k\}} = \left(\frac{\partial^2 \tilde{f}_R}{\partial x_i^2}\right)^{\{k\}} = \frac{-2}{x_i^{\{k\}}}\left(\frac{\partial f}{\partial x_i}\right)^{\{k\}} \tag{9}$$

Similar curvature estimation approaches can be derived for approximation functions other than the reciprocal approximation (6). In [14], we have, for example, derived a similar expression for the quadratic Taylor series expansion to the exponential approximation. In [15], we have derived the quadratic Taylor series expansion to an even wider number of approximations, including the TANA-3 approximation [17,18] and the conservative approximation [3,19]. For the sake of clarity, herein, we will not elaborate on all these possibilities, but they are trivially included into approximation (3); only the higher-order curvatures $c_{2i}^{\{k\}}$ change.

3. Convexity of the Diagonal Quadratic Approximation

To obtain strictly convex primal subproblems, we enforce $c_{2i_\alpha}^{\{k\}} = \max(\epsilon_n > 0, c_{2i_\alpha}^{\{k\}})\ \forall\ i$ if $\alpha = 0$, and $c_{2i_\alpha}^{\{k\}} = \max(0, c_{2i_\alpha}^{\{k\}})\ \forall\ i$ if $\alpha > 0$, with ϵ_n selected rather arbitrarily as 10^{-5}. (Improved strategies are of course possible for linear objective functions, e.g., in weight minimization). For the constraints, we will use curvatures (5); for the objective function, either curvatures (5) or curvatures (9). Neither the spherical quadratic approximation nor the quadratic approximation to the reciprocal approximation are necessarily conservative; we will enforce this during inner loops in the sections to come.

C. Dual Statement

Rather than primal approximate subproblem $P_P[k]$, $k = 1, 2, 3, \ldots$, a dual approximate subproblem $P_D[k]$, $k = 1, 2, 3, \ldots$ may also be formulated. We start by defining the approximate dual function $\tilde{\gamma}(\boldsymbol{\lambda})$ as

$$\tilde{\gamma}(\boldsymbol{\lambda}) = \min_{\boldsymbol{x}}\left\{\tilde{f}_0(\boldsymbol{x}) + \sum_{j=1}^{m}\lambda_j \tilde{f}_j(\boldsymbol{x})\right\} \tag{10}$$

The minimizer of Eq. (10) for $\boldsymbol{\lambda}$ given will be denoted by $\boldsymbol{x}(\boldsymbol{\lambda})$, because it depends on $\boldsymbol{\lambda}$. We may thus write

$$\tilde{\gamma}(\boldsymbol{\lambda}) = \tilde{f}_0[\boldsymbol{x}(\boldsymbol{\lambda})] + \sum_{j=1}^{m}\lambda_j \tilde{f}_j[\boldsymbol{x}(\boldsymbol{\lambda})] \tag{11}$$

This allows for the formulation of the dual approximate subproblem as

Dual approximate subproblem $P_D[k]$

$$\max_{\boldsymbol{\lambda}}\left\{\tilde{\gamma}(\boldsymbol{\lambda}) = \tilde{f}_0[\boldsymbol{x}(\boldsymbol{\lambda})] + \sum_{j=1}^{m}\lambda_j \tilde{f}_j[\boldsymbol{x}(\boldsymbol{\lambda})]\right\} \qquad \text{subject to } \lambda_j \geq 0$$
$$j = 1, 2, \ldots, m \tag{12}$$

This bound-constrained problem requires the determination of the m unknowns λ_j only, subject to m nonnegativity constraints on the λ_j.

If primal subproblem (2) is strictly convex and separable and the approximation functions to f_0 and f_j are simple enough, it may be possible to find a simple (analytical) expression for the minimizing primal variables $\boldsymbol{x}(\boldsymbol{\lambda})$ in Eq. (11) in terms of the dual variables $\boldsymbol{\lambda}$. For the diagonal quadratic approximations (3), this is indeed the case. Introducing

$$\beta_i(\boldsymbol{\lambda}) = x_i^{\{k\}} - \left(c_{2i_0}^{\{k\}} + \sum_{j=1}^{m}\lambda_j c_{2i_j}^{\{k\}}\right)^{-1}\left(\frac{\partial f_0^{\{k\}}}{\partial x_i} + \sum_{j=1}^{m}\lambda_j \frac{\partial f_j^{\{k\}}}{\partial x_i}\right) \tag{13}$$

we obtain

$$x_i(\boldsymbol{\lambda}) = \begin{cases} \beta_i(\boldsymbol{\lambda}) & \text{if } \check{x}_i < \beta_i(\boldsymbol{\lambda}) < \hat{x}_i, \\ \check{x}_i & \text{if } \beta_i(\boldsymbol{\lambda}) \leq \check{x}_i, \\ \hat{x}_i & \text{if } \beta_i(\boldsymbol{\lambda}) \geq \hat{x}_i \end{cases} \tag{14}$$

for $i = 1, 2, \ldots, n$, being the final result that we require for solving dual problem (12) with the diagonal quadratic approximations (3). For details, the reader is referred to Falk [20] and our previous efforts [21]. The optimal point in the primal space of subproblem k is denoted $\boldsymbol{x}^{\{k*\}}$, which will converge to $\boldsymbol{x}^*$, a minimizer of problem P_{NLP}.

Dual approximate subproblem (12) may be solved numerically in many different ways. Conjugate gradient solvers in the Fletcher–Reeves tradition may, for example, be used. Our current implementation uses a limited memory BFGS variable metric solver [22,23], able to take the simple nonnegativity constraints into consideration.

D. Convergence Rate and Interpolation Order

It is clear that the approximate higher-order curvatures (5) and (9) in diagonal quadratic approximation (3) differ from the diagonal Hessian terms of the objective function f_0 and the constraint functions f_j (also see our elaboration in [12]). (Indeed, the Hessians of f_0 and the f_j will, in general, not even be diagonal.)

Hence, diagonal quadratic approximation (3) is in essence a linear approximation, corrected in some heuristic sense to approximately provide for an averaged higher-order effect; the term quadratic refers to the (maximum) order of the terms in the approximations $\tilde{f}_0$ and $\tilde{f}_j$, and not the order to which the derivatives of the approximation match those of the original functions f_0 and f_j. (A similar argument holds for the MMA approximations.)

This, however, does not mean that an algorithm based on approximation (3) will be linearly convergent. General statements about rate of convergence will require considerations of the nature of the f_0 and the f_j, and the corresponding approximate higher-order curvatures $c_{2i}^{\{k\}}$ used.

To elaborate, if f_0 and the f_j are spherical quadratic (or linear), an algorithm based on approximate higher-order curvatures (5) is exact.

On the other hand, consider a problem for which the f_0 and the f_j are reciprocal-like. Then, algorithms based on reciprocal intermediate variables (like MMA [2] and CONLIN [3]) may be expected to do very well, because the nonlinear response is linearized through the use of the reciprocal intermediate variables. For an elaboration on intermediate variables, the reader is referred to Haftka and Gürdal [24].

Reciprocal intermediate variables are, for example, very important in structural mechanics: for a statically determinate structure subject to stress and displacement constraints only, reciprocal intermediate variables yield approximations that are exact. Hence, an approximation based on the approximate higher-order curvatures (9) may, like MMA and CONLIN, be expected to also be effective for problems in structural mechanics, because the approximation used is exactly the quadratic Taylor expansion to the reciprocal approximation about the point where the approximation is constructed. (This, in fact, clearly positions SAO w.r.t. SQP methods; the former probably become attractive only when the latter become impractical due to computational considerations.)

Because elaborate convergence analyses for combinations of many different problems and approximate higher-order curvatures $c_{2i}^{\{k\}}$ are impractical, the performance of algorithms based on problem-specific approximations (like the one presented herein, MMA, and CONLIN) will probably have to be done on the basis of numerical performance for a class of problems. (An important example is the almost exclusive use of MMA for problems in structural topology optimization.)

Nevertheless, for any (partially) concave function, convex approximations like those used herein, and in MMA and CONLIN, reduce to the linear approximation, and the rate of convergence indeed becomes linear. What is more, inappropriate higher-order curvatures $c_{2i}^{\{k\}}$ may actually decrease the rate of convergence, rather than improve it.

Finally, for an algorithm based on the diagonal quadratic approximations considered herein, some knowledge of the objective and constraint functions is required to enable the specification of suitable higher-order curvatures $c_{2i}^{\{k\}}$. This is not necessary for MMA and CONLIN. Although this may possibly be considered a drawback of our method, it may also be considered a salient feature: the range of applicability of the algorithm is extended, albeit at the cost of requiring some insight into the original problem formulation. As argued, this, however, is not too difficult in many disciplines and fields of study, like structural optimization, least-squares problems, etc.

IV. Conservatism for the Diagonal Quadratic Approximation

A. Overview

As an alternative to trust-region methods, Svanberg has developed the concept of conservative convex and separable approximations to effect global convergence [10]. He implemented and demonstrated this numerically for the MMA algorithm, denoting the resulting algorithm a "globally convergent version of MMA."

CCSA methods are simple, but effective. Every approximation function in the CCSA subproblem is separable and convex, and consists of two terms (see Svanberg [10]): the first term, denoted v in [10], is responsible for matching the function gradient to the approximation gradient to guarantee first-order accuracy of the expansion. The second term, denoted w in [10], has zero function value and gradient at the approximation point and hence cannot affect the first-order approximation. In the framework developed by Svanberg, the purpose of the second term is primarily to stabilize the iteration.

Herein, we cast the convexified diagonal quadratic approximations presented in Sec. III.B into the CCSA framework of Svanberg [10]. This corresponds to choosing a linear form for v and a quadratic form for w. Accordingly, Svanberg introduced inner iterations for each outer iteration in the sequential approximate optimization strategy outlined in Sec. III: for an arbitrary outer iteration k, the solution $\boldsymbol{x}^{\{k*\}}$ is accepted as the starting point for outer iteration $k+1$ if *all* the approximation functions $\tilde{f}_j(\boldsymbol{x}^{\{k*\}})$ are conservative at $\boldsymbol{x}^{\{k*\}}$, viz., if $\tilde{f}_j(\boldsymbol{x}^{\{k*\}}) \geq f_j(\boldsymbol{x}^{\{k*\}})$, $j = 0, 1, 2, \ldots, m$. If any of the approximation functions $\tilde{f}_j(\boldsymbol{x}^{\{k*\}})$ are not conservative at $\boldsymbol{x}^{\{k*\}}$, inner iterations l are initiated, during which the curvatures $c_{2i_\alpha}^{\{k,l\}}$ of all the nonconservative approximations are iteratively increased until conservatism has been enforced at $\boldsymbol{x}^{\{(k,l)*\}}$, after which the solution $\boldsymbol{x}^{\{(k,l)*\}}$ is accepted as the starting point $\boldsymbol{x}^{\{k+1\}}$ for outer iteration $k+1$. The outer iterations k require function and gradient evaluations of the objective and the constraint functions, whereas the inner iterations l require only function evaluations.

Clearly then, CCSA methods simply aim to ensure that each step is a feasible descent step, which in itself is adequate to demonstrate that the sequence will terminate at a stationary point. However, the rate of convergence may be low, because the use of accurate second-order information including off-diagonal terms is in general impossible.

B. Feasible Descent Steps

In the CCSA framework of Svanberg [10], proof of convergence is demonstrated if *all* the approximation functions are conservative at successive solutions $\boldsymbol{x}^{\{k*\}}$ of the approximate subproblems. This includes approximations to constraints that are feasible, possibly even by a large margin.

However, the proof presented by Svanberg [10] simply demands that feasible descent steps are made; indiscriminately enforcing conservatism is not necessary. It is also easy to envision situations where this may not be desirable.

Denoting the approach of indiscriminately enforcing conservatism "strict conservatism," we propose "relaxed conservatism," in which *feasible* descent steps (in terms of the unrelaxed objective function f_0) are unconditionally accepted.

C. Curvature Initialization

If the higher-order curvatures $c_{2i}^{\{k\}}$ are estimated using curvatures (5) at the start of outer iteration k, some strategy is required when $k = 0$, because no historic information is then available. Snyman and Hay [4] have opted to set the initial curvatures $c_{2i}^{\{0\}}$ to zero, that is, to depart with the linear approximation. This typically results in a first outer iteration step that is limited by only the variable bounds (or the move limit used). This step is often highly unconservative, and we have opted to rather depart with a nonlinear approximation by (arbitrarily) selecting the initial curvatures $c_{2i}^{\{0\}}$ as unity.

If the quadratic Taylor series expansion to the reciprocal approximation is used in an algorithm, that is, when the higher-order curvatures are calculated using curvatures (9) at the start of outer iteration k, no special consideration is needed when $k = 0$; it is possible to construct the quadratic approximation to the reciprocal approximation in any point $\boldsymbol{x}^{\{k\}}$.

In part due to the two different methods we will use to calculate the higher-order curvatures $c_{2i}^{\{k\}}$, we will use a very simple heuristic to enforce conservatism during inner loops: we will use a constant multiplication factor $\chi > 1$ to increase the $c_{2i}^{\{k\}}$, and hence, increase the conservatism of approximation (3). However, more refined strategies (possibly with an additional computational advantage) are certainly possible.

D. Algorithmic Implementation and Considerations

We will now develop a rudimentary conservative SAO algorithm based on the separable diagonal quadratic approximations (3), cast in the CCSA framework of Svanberg [10].

For a given outer iteration k, conservatism of the approximations (3) may be effected by simply multiplying the $c_{2i_\alpha}^{\{k\}}$ by some constant $\chi > 1$ in an inner loop. [For the objective function, it may be recalled that the $c_{2i_0}^{\{k\}}$ are always selected to be positive, to ensure that the subproblems are strictly convex. For the constraints, it should be remembered that the $c_{2i_\alpha}^{\{k\}}$ (with $\alpha \geq 1$) may be zero at the start of an iteration, although this can only happen if constraint α is indeed linear.]

Accordingly, given an initial point $\boldsymbol{x}^{\{0\}}$, a conservative algorithm based on convex separable diagonal quadratic approximations proceeds as follows (using a FORTRAN-like pseudolanguage):

1) **Initialization:** Select positive constants ϵ_1, ϵ_2, ϵ_x, ϵ_n, $\hat{k}$, $\chi_1 > 1$, $\chi_2 > 1$. Set $k := 0$, $l := 0$.
2) **Function evaluations and sensitivity analysis:** Compute $f_\alpha(\boldsymbol{x}^{\{0\}})$, $\nabla f_\alpha(\boldsymbol{x}^{\{0\}})$, $\alpha = 0, 1, 2, \ldots, m$.
3) **Construct the approximate curvatures:** Calculate the initial outer curvatures $c_{2i_0}^{\{k\}} = \mathbf{max}(\epsilon_n > 0, c_{2i_0}^{\{k\}})$ $\forall\ i$ and $c_{2i_\alpha}^{\{k\}} = \mathbf{max}(0, c_{2i_\alpha}^{\{k\}})$ $\forall\ i$, $\alpha = 1, 2, \ldots, m$, using the strategies mentioned in Sec. III.B.
4) **Approximate optimization:** Construct local approximate subproblem $P_D[k]$ at $\boldsymbol{x}^{\{k\}}$. Solve this subproblem to arrive at $(\boldsymbol{x}^{\{k*\}}, \boldsymbol{\lambda}^{\{k*\}})$.
5) **Function evaluations:** Compute $f_\alpha(\boldsymbol{x}^{\{k*\}})$, $\alpha = 0, 1, 2, \ldots, m$.
6) **Test if $\boldsymbol{x}^{\{k*\}}$ is acceptable:**
   a) **Test if $\boldsymbol{x}^{\{k*\}}$ represents a feasible descent step:** IF $f_0(\boldsymbol{x}^{\{k*\}}) < f_0(\boldsymbol{x}^{\{(k-1)*\}})$ for $k > 0$, AND $\mathbf{max}\{f_\alpha(\boldsymbol{x}^{\{k*\}})\} \le 0$, $\alpha = 0, 1, 2, \ldots, m$, GO TO step 8,
   b) **Test if $\boldsymbol{x}^{\{k*\}}$ represents a conservative step:** IF $\tilde{f}_0(\boldsymbol{x}^{\{k*\}}) \ge (f_0(\boldsymbol{x}^{\{k*\}}) - \epsilon_1)$, AND $\tilde{f}_\alpha(\boldsymbol{x}^{\{k*\}}) \ge (f_\alpha(\boldsymbol{x}^{\{k*\}}) - \epsilon_2)$, $\alpha = 0, 1, 2, \ldots, m$, GO TO step 8.
7) **Initiate an inner loop to effect conservatism:**
   a) Set $l := l + 1$.
   b) IF $\tilde{f}_0(\boldsymbol{x}^{\{k*\}}) < (f_0(\boldsymbol{x}^{\{k*\}}) - \epsilon_1)$, set $c_{2i_0}^{\{k\}} := \chi_1 c_{2i_0}^{\{k\}}$.
   c) IF $\tilde{f}_\alpha(\boldsymbol{x}^{\{k*\}}) < (f_\alpha(\boldsymbol{x}^{\{k*\}}) - \epsilon_2)$, set $c_{2i_\alpha}^{\{k\}} := \chi_2 c_{2i_\alpha}^{\{k\}}$, $\alpha = 1, 2, \ldots, m$.
   d) GO TO step 4
8) **Move to the new iterate:** Set $\boldsymbol{x}^{\{k+1\}} := \boldsymbol{x}^{\{k*\}}$.
9) **Convergence test:** IF $\|\boldsymbol{x}^{\{k+1\}} - \boldsymbol{x}^{\{k\}}\| \le \epsilon_x$, OR $k = \hat{k}$, STOP.
10) **Sensitivity analysis:** Compute $\nabla f_\alpha(\boldsymbol{x}^{\{k+1\}})$, $\alpha = 0, 1, 2, \ldots, m$.
11) **Initiate an additional outer loop:** Set $k := k + 1$ and GO TO step 3.

Note that it is always possible to perform the minimization of approximate subproblem $P_D[k]$ at $\boldsymbol{x}^{\{k\}}$ in step 4, because primal subproblem $P_P[k]$ is strictly convex.[¶] In addition, the minimizer $\boldsymbol{x}^{\{k*\}}$ of $P_D[k]$ at $\boldsymbol{x}^{\{k\}}$ will always be (approximately) feasible, because the modified problem P_{NLP} represented by problem (A1) is always feasible for suitable values of θ_j (see the Appendix). In step 4, the relaxed approximate problem is minimized; in steps 6 and 7, the unrelaxed problem is implied.

As said, strict convexity of approximate subproblem $P_P[k]$ is effected by the inequality $c_{2i_0}^{\{0\}} > 0$; for the constraints, the requirement $c_{2i_j}^{\{0\}} \ge 0\ \forall\ j$ suffices. These conditions translate into a strictly convex objective function, subject to convex or strictly convex constraints; both conditions imply that the subproblems are strictly convex. (This property was used in developing the Falk dual.)

The parameters ϵ_1, ϵ_2 are introduced to relax the conservatism requirement, mainly to prevent excessive inner iterations due to inadequate precision.

However, an interesting variant of any CCSA algorithm may be obtained by selecting ϵ_1, ϵ_2 "large," and to then progressively tighten these tolerances as the iterations proceed. If problem P_{NLP} is, for example, multimodal, this strategy may allow for a "hill-climbing" ability (in terms of the original problem P_{NLP}) during initial and intermediate steps, while oscillations in the terminal phase are prevented, and termination is still guaranteed (if not to the global optimum). This strategy, however, seems to require some knowledge about the original problem P_{NLP} in prescribing sensible tightening rates for ϵ_1, ϵ_2, and termination is not guaranteed.

V. Numerical Experiments

We now present numerical results for the conservative algorithm we have developed, which we will denote algorithm GCDQ (which stands for globally convergent diagonal quadratic); we will denote the nonconservative implementation of the algorithm (i.e., without steps 6 and 7 in the algorithm in the previous subsection) by DQ (note that DQ still generates a series of convex subproblems). We compare our results with GCMMA: the conservative, globally convergent version of MMA [10].[**]

Unless otherwise stated, we will use $\epsilon_1 = 10^{-7}$ and $\epsilon_2 = 10^{-4}$ in GCDQ. In GCMMA, we will use all the default settings (any exceptions are explicitly mentioned). Superscript $*$ will indicate the approximate calculated optima, n_b the number of variables at a bound constraint, and k and l, respectively, indicate outer and inner iterations. Invariably, we use the simple spherical quadratic approximation implied by curvatures (5), except for topology optimization problems, when we resort to curvatures (9) for the approximate objective function $\tilde{f}_0$. Finally, in GCDQ, we use $\chi_1 = \chi_2 = 2$ (although we have not performed any experiments to determine if these values are even close to being optimal). For GCMMA, the reader is referred to the discussion of the parameters χ_1, χ_2 by Svanberg [10].

The algorithms are stopped when $\|\boldsymbol{x}^{\{k+1\}} - \boldsymbol{x}^{\{k\}}\| \le \epsilon_x$, with ϵ_x identical for all the algorithms. It is standard practice in algorithms based on conservatism to limit the maximum number of inner loops, but we will not do so in this study, because this (strictly speaking) renders the algorithms not necessarily convergent.

A. Two Nonconvex Programming Problems of High Dimensionality

We start with two nonconvex problems proposed by Svanberg [10]. Both are expressed in terms of the symmetric, fully populated $n \times n$ matrices $\boldsymbol{S}$, $\boldsymbol{P}$, and $\boldsymbol{Q}$, with elements given by

$$s_{ij} = \frac{2 + \sin(4\pi\vartheta_{ij})}{(1 + |i-j|)\,\ell_n(n)}, \qquad p_{ij} = \frac{1 + 2\vartheta_{ij}}{(1 + |i-j|)\,\ell_n(n)}$$
$$q_{ij} = \frac{3 - 2\vartheta_{ij}}{(1 + |i-j|)\,\ell_n(n)}$$

where

$$\vartheta_{ij} = \frac{i + j - 2}{2n - 2} \in [0, 1] \quad \forall\ i, j$$

and $n > 1$.

The first nonconvex problem is formulated as

$$\min_{\boldsymbol{x}} f_0(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{S} \boldsymbol{x}$$
$$\text{subject to } f_1(\boldsymbol{x}) = \frac{n}{2} - \boldsymbol{x}^T \boldsymbol{P} \boldsymbol{x} \le 0$$
$$f_2(\boldsymbol{x}) = \frac{n}{2} - \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} \le 0$$
$$-1 \le x_i \le 1$$

with starting point $\boldsymbol{x}^0 = (0.5, 0.5, \cdots, 0.5)^T$. The objective function $f_0(\boldsymbol{x})$ is strictly convex, but the nonlinear constraint functions $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$ are strictly concave. Numerical results are presented in Table 1.

The second nonconvex problem is formulated as

$$\min_{\boldsymbol{x}} f_0(\boldsymbol{x}) = -\boldsymbol{x}^T \boldsymbol{S} \boldsymbol{x},$$
$$\text{subject to } f_1(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{P} \boldsymbol{x} - \frac{n}{2} \le 0$$
$$f_2(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} - \frac{n}{2} \le 0,$$
$$-1 \le x_i \le 1$$

with starting point $\boldsymbol{x}^0 = (0.25, 0.25, \ldots, 0.25)^T$. This time, the objective function $f_0(\boldsymbol{x})$ is strictly concave, whereas the nonlinear constraint functions $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$ are strictly convex. Numerical results for this problem are presented in Table 2. Note that we have used $\epsilon_x = 5 \times 10^{-4}$ for both GCDQ and GCMMA; hence, our results

[¶] For the sake of clarity, we have used superscript $\{k\}$, rather than $\{k, l\}$, which is more correct, to represent the iteration counters. The meaning of the abbreviated notation, however, is clear; $c_{2i_j}^{\{k\}}$ should be read as $c_{2i_j}^{\{k,l\}}$, etc.

[**] For both MMA and GCMMA, we use the FORTRAN versions coded by Svanberg [2,11].

Table 1 First large nonconvex programming problem: results for GCMMA and GCDQ

	GCMMA						GCDQ					
n	f_0^*	n_b^*	λ_1^*	λ_2^*	k^*	l^*	f_0^*	n_b^*	λ_1^*	λ_2^*	k^*	l^*
1000	260.85	184	0.138	0.451	186	214	260.85	183	0.138	0.451	174	16
2000	523.51	354	0.147	0.442	202	240	523.51	351	0.147	0.442	194	9
5000	1312.05	841	0.156	0.431	257	304	1312.05	835	0.156	0.431	226	5
10,000	2626.76	1632	0.161	0.425	289	348	2626.76	1627	0.161	0.425	276	2
20,000	5256.56	3188	0.165	0.420	337	405	5256.56	3178	0.165	0.420	317	1

Table 2 Second large nonconvex programming problem: results for GCMMA and GCDQ

	GCMMA						GCDQ					
n	f_0^*	n_b^*	λ_1^*	λ_2^*	k^*	l^*	f_0^*	n_b^*	λ_1^*	λ_2^*	k^*	l^*
1000	−739.15	184	0.549	0.862	415	422	−739.15	183	0.549	0.862	351	97
2000	−1476.49	353	0.558	0.853	473	488	−1476.49	351	0.558	0.853	425	114
5000	−3687.95	839	0.569	0.844	590	615	−3687.95	838	0.569	0.844	556	147
10,000	−7373.24	1629	0.575	0.839	707	731	−7373.24	1629	0.575	0.839	684	186
20,000	−14, 743.44	3182	0.580	0.834	817	841	−14, 743.44	3184	0.580	0.835	806	237

for GCMMA differ slightly with those obtained by Svanberg [10], who used a different stopping criterion.

Tables 1 and 2 reveal that GCMMA, on average, requires slightly more outer iterations than GCDQ and also far more inner iterations than GCDQ. However, of equal importance is the fact that we have elsewhere [21] presented results for the nonconservative algorithms DQ and MMA (viz., the algorithms presented herein, but without conservative inner loops). The nonconservative algorithms are not guaranteed to converge or even terminate. Notwithstanding, both nonconservative algorithms converged to the optimal solutions requiring notably less computational effort than the globally convergent versions, for both problems.

In particular, it is shown in [21] that, for nonconvex programming problem 1, nonconservative algorithm DQ is far more efficient than both the conservative algorithms studied herein. For nonconvex programming problem 2, nonconservative algorithm MMA is by far more efficient than the conservative algorithms. This illustrates the computational penalty one may have to pay for having a globally convergent optimization algorithm.

Finally, in generating the results presented in Tables 1 and 2 for GCDQ, conservatism was only enforced when a feasible descent step could not be made, in accordance with the algorithmic implementation presented in Sec. IV (as opposed to GCMMA, where conservatism is unconditionally enforced in each iteration). In Table 3, results are again presented for the first large nonconvex programming problem, but this time, conservatism is enforced in each and every iteration for the objective and the constraint functions.

Clearly, the unconditional enforcement of conservatism may imply a considerable additional computational burden. However, the results presented for the first large nonconvex programming problem are by far the most pronounced; for all the other examples studied, the penalty of indiscriminately enforcing conservatism was far less important. In fact, when repeating the aforementioned experiment for the second large nonconvex programming problem, the computational requirements for relaxed and strict enforcement of conservatism are identical (not explicitly shown in tabulated form); this time, there is no advantage derived from relaxed conservatism.

Table 3 First large nonconvex programming problem: results for GCDQ, with conservatism enforced strictly and indiscriminately in each iteration (compare with the GCDQ results presented Table 1, in which conservatism is only enforced if a feasible descent step cannot be made)

	GCDQ					
n	f_0^*	n_b^*	λ_1^*	λ_2^*	k^*	l^*
1000	260.85	186	0.138	0.451	199	112
2000	523.51	353	0.147	0.442	226	121
5000	1312.05	839	0.156	0.431	270	143
10,000	2626.76	1630	0.161	0.425	319	167
20,000	5256.56	3184	0.165	0.420	362	176

B. Snake Problem

We now apply our algorithm to the so-called snake problem, proposed by Svanberg for "anyone who wants to test a new method for nonlinear optimization" [11].

Let d be a given positive integer, and let δ_s be a given small positive real number. For $i = 1, 2, \ldots, d$, let

$$\psi_i = \frac{(3i - 2d)\pi}{6d}$$

$$g_i(\boldsymbol{x}) = \frac{x_i^2 + x_{d+i}^2 - 1}{\delta_s}, \quad \text{and} \quad h_i(\boldsymbol{x}) = \frac{x_{2d+i} - 2x_i x_{d+i}}{\delta_s}$$

Then, consider the following problem in the variables $\boldsymbol{x} = (x_1, \ldots, x_{3d})^T$:

$$\min_{\boldsymbol{x}} f_0(\boldsymbol{x}) = \sum_{i=1}^{d} (x_i \cos\psi_i + x_{d+i} \sin\psi_i - 0.1x_{2d+1})$$

$$\text{subject to } \sum_{i=1}^{d} (x_i^2 + x_{2d+i}^2) \le d \qquad -2 \le g_i(\boldsymbol{x}) + g_i(\boldsymbol{x})^7 \le 2$$

$$i = 1, 2, \ldots, d \qquad -2 \le h_i(\boldsymbol{x}) + h_i(\boldsymbol{x})^7 \le 2, \qquad i = 1, 2, \ldots, d$$

$$-2 \le x_j \le 2, \qquad j = 1, 2, \ldots, 3d$$

For a short discussion of the problem, see Svanberg [11]. Svanberg considers the problem "rather difficult to solve" if the following feasible, but far from optimal, starting point $\boldsymbol{x}^{\{0\}}$ is chosen:

$$x_i^{\{0\}} = \cos\left(\psi_i + \frac{\pi}{12}\right)$$

$$x_{d+i}^{\{0\}} = \sin\left(\psi_i + \frac{\pi}{12}\right), \quad \text{and} \quad x_{2d+i}^{\{0\}} = \sin\left(2\psi_i + \frac{\pi}{6}\right)$$

$$i = 1, 2, \ldots, d$$

Results for selected values of d are presented in Table 4 for $\delta_s = 0.1$ (neither GCMMA nor GCDQ seem to be sensitive to the value of δ_s) and $\epsilon_x = 10^{-6}$. GCDQ not only requires fewer outer iterations than GCMMA, but also far fewer inner iterations than GCMMA.

Interestingly, for $d = 10$, Svanberg [11] reports the following:

> Two rather well-known codes based on SQP needed much more than 100 major iterations before the optimal solution had been obtained with similar accuracy.... Each such major iteration includes function and gradient calculations of the original

Table 4 Snake problem results for GCMMA and GCDQ

			GCMMA			GCDQ		
d	n	m	k^*	l^*	f_0^*	k^*	l^*	f_0^*
10	30	41	72	437	−10.02298	50	284	−10.02298
20	60	81	77	617	−19.95530	59	367	−19.95530
50	150	201	87	816	−49.75444	71	426	−49.75444
100	300	401	114	1175	−99.42045	78	482	−99.42045
200	600	801	135	1491	−198.75277	84	494	−198.75278

functions, the solution of a QP problem, and a line search with additional evaluations of the original functions.

Our own experiments corroborate this.

For this problem, there is again some advantage to be derived from using relaxed as opposed to strict conservatism in GCDQ, albeit not as dramatic as for the first large nonconvex problem. For $d = 10$, for example, the number of outer iterations k^* in GCDQ remain unchanged when strict conservatism is enforced, but the number of outer iterations l^* increases from 284 to 349.

Finally, with a suitably small move limit, nonconservative algorithm DQ converges to the optimum, at the cost of many outer iterations (not shown, and we have only tested this for $d = 10$). Nonconservative algorithm MMA does not terminate for $d = 10$.

C. Twelve-Corner Polytope Problem

Consider the 12-corner polytope problem in the 21 variables $(r_1, \ldots, r_{11}, v_1, \ldots, v_{10})$ proposed by Svanberg [25]. It is required to maximize the area f_0, subject to a single constraint f_1 on the circumference. The problem is formulated as

$$\min_{r_i, v_i} f_0(\boldsymbol{r}, \boldsymbol{v}) = -\frac{1}{2}\sum_{i=1}^{10}[r_i r_{i+1}\sin(v_i)]$$

$$\text{subject to } f_1(\boldsymbol{r}, \boldsymbol{v}) = r_1 + r_{11} + \sum_{i=1}^{10}[r_i^2 + r_{i+1}^2 - 2r_i r_{i+1}\cos(v_i)]^{1/2} - 60 \le 0$$

$$1 \le r_i \le 30, \qquad i = 1, 2, \ldots, 11 \qquad 1 \le v_i \le 45 \qquad i = 12, 13, \ldots, 21$$

The starting point is taken as $r_i^0 = 11$ and $v_i^0 = 18$ for all i. The optimal solution is $f_0^* = -75(2 + \sqrt{3}) = -279.9038$, with $r_i^* = 5\sin(15i)/\sin(15)$ and $v_i^* = 15$. We have used $\epsilon_x = 10^{-6}$. Numerical results are presented in Table 5; GCDQ performs better than GCMMA for this example.

It is again interesting to compare the conservative algorithms with their nonconservative counterparts. For this problem, nonconservative algorithm MMA does not converge [21]. Nonconservative algorithm DQ, however, does converge, and does so relatively quickly when compared to GCDQ (again see [21]); this again illustrates the computational burden that may result from enforcing global convergence (if the nonconservative algorithm converges at all, of course).

Table 5 Twelve-corner polytope: results for GCMMA and GCDQ

	f_0^*	f_1^*	λ_1^*	k^*	l^*
GCMMA	−279.9038	1.21×10^{-9}	9.330	424	485
GCDQ	−279.9038	7.07×10^{-8}	9.330	183	77

D. Optimal Design of Truss Structures

Next, we consider the classical minimum weight design problem of trusslike structures, in which the domain $\mathcal{X}$ is discretized using the finite element method. The optimal design problem is expressed as

$$\min_{\boldsymbol{x}} f_0(\boldsymbol{x}) = \sum_{i=1}^{n_{\text{el}}} \rho_i l_i x_i$$

$$\text{subject to } f_j(\boldsymbol{x}) = u_j(\boldsymbol{x}) - \bar{u}_j \le 0 \qquad j = 1, 2, \ldots, m$$

$$\boldsymbol{K}\boldsymbol{q} = \boldsymbol{r}$$

$$0 < \check{x}_i \le x_i \le \hat{x}_i \qquad i = 1, 2, \ldots, n \tag{15}$$

where f_0 represents the structural weight, ρ_i represents the mass density of element i, and l_i is a geometric parameter such that the product $l_i x_i$ is the volume of element i. The global assembled finite element displacement and force vectors are represented by $\boldsymbol{q} = \boldsymbol{q}(\boldsymbol{x})$ and $\boldsymbol{r}$, whereas $\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{x})$ represents the global finite element stiffness matrix. The constraints f_j represent the behavioral constraints, for example, limits on stress and displacement, etc., where the $\bar{u}_j$ represent prescribed allowable quantities of, for example, stress and displacement. (Often, the constraints are normalized). There are n_{el} elements in the finite element mesh. The vector of cross-sectional areas (the sizing design variables) is $\boldsymbol{x}$, with $\check{x}_i$ and $\hat{x}_i$ the bound constraints of variable x_i. In structural weight minimization, the number of inequality constraints m may be very large, with $m \gg n$. This makes constraint deletion strategies attractive. We have, however, not done this herein.

Numerical results for three convex example problems are presented in Tables 6 and 7. The former table also tabulates the problem bounds and the starting points used. For geometric details and loading conditions, the reader is referred to the literature, for example, see [26] and many others. There is little to choose between the conservative algorithms; GCMMA requires fewer outer iterations, whereas GCDQ requires fewer inner iterations. It is again clear that the nonconservative variants DQ and MMA require less computational effort than their conservative counterparts.

E. Topology Optimization: Messerschmitt-Bölkow-Blohm Beam

Finally, we consider the optimal topology design of the well-known Messerschmitt-Bölkow-Blohm (MBB) beam; the design domain is discretized using the finite element method. The MBB problem is described by, for example, Bendsøe and Sigmund [27].

The optimization problem is expressed as

$$\min_{\boldsymbol{\rho}} c(\boldsymbol{\rho}) = \boldsymbol{q}^T\boldsymbol{K}\boldsymbol{q} = \sum_{i=1}^{n_{\text{el}}} \rho_i^p \boldsymbol{q}_i^{eT}\boldsymbol{K}_0^e\boldsymbol{q}_i^e$$

$$\text{subject to } f_1 = v - \bar{v} = \sum_{i=1}^{n_{\text{el}}} \rho_i^e v_i^e - \bar{v} \le 0$$

$$\boldsymbol{K}\boldsymbol{q} = \boldsymbol{r}$$

$$\boldsymbol{0} < \check{\boldsymbol{\rho}} \le \boldsymbol{\rho} \le \boldsymbol{1} \tag{16}$$

where c represents compliance, $\boldsymbol{q} = \boldsymbol{q}(\boldsymbol{x})$ and $\boldsymbol{r}$ represent the global assembled finite element displacement and force vectors, and $\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{x})$ represents the global finite element stiffness matrix. There are n_{el} finite elements; superscript e indicates elemental quantities. The vector of elemental densities (the design variables) is denoted $\boldsymbol{\rho}$,

Table 6 Truss results for MMA and GCMMA

							MMA			GCMMA		
No.	Name	$\check{x}_i$	$\hat{x}_i$	$x_i^{\{0\}}$	n	m	f^*	k^*	l^*	f^*	k^*	l^*
1	3-bar truss	0.1	14.5	5.0	3	24	20.54	12	——	20.54	15	28
2	10-bar truss	0.5	33.0	5.0	10	28	4193.19	32	——	4193.19	29	81
3	36-bar truss	5.0	175.0	5.0	36	198	35,726.22	29	——	35,726.22	23	36

Table 7 Truss results for DQ and GCDQ

		DQ			GCDQ		
No.	Name	f^*	k^*	l^*	f^*	k^*	l^*
1	3-bar truss	20.54	9	——	20.54	11	8
2	10-bar truss	4193.19	23	——	4193.19	45	28
3	36-bar truss	35,726.22	30	——	35,726.22	36	11

Table 8 Topology optimization of the MBB beam using GCMMA and GCDQ, for $p = 1$

	f_0^*	k^*	l^*
GCMMA	165.884	68	10
GCDQ with curvatures (9)	165.884	58	0
GCDQ with curvatures (5)	165.982	35	47

whereas v represents the structural volume, with $\bar{v}$ a prescribed limit on volume. We use $\epsilon_x = 10^{-4}$. The SIMP (solid isotropic material with penalization) penalty parameter is p, which drives the solution to black and white, for example, see [27].

We optimize the beam using $\check{\rho}_i = 0.001 \ \forall \ i$, a prescribed volume fraction $\bar{v} = 0.5$, and we select $x_i^{\{0\}} = \bar{v} = 0.5 \ \forall \ i$. The mesh is discretized using 75×25 elements for half the beam, due to symmetry. We use standard isoparametric displacement-based four-node membrane finite elements (e.g., see [28]), often known as "Q4" elements. Finally, we use the sensitivity filter proposed by Sigmund [29] to regularize the solution; the filter radius used is 8% of the beam height.

First, we present results for the penalty parameter $p = 1$ in Table 8. Setting $p = 1$ results in a relaxed, strictly convex objective function, which allows for direct comparison between the GCMMA and GCDQ algorithms (for $p > 1$, the objective becomes multimodal [27]). The table illustrates that the same minimum is found using GCMMA and GCDQ, when using curvatures (9) in the latter algorithm to approximate the diagonal Hessian terms.

Next, we set $p = 3$. This time, algorithm GCDQ with curvatures (9) requires 30 outer iterations and 14 inner iterations to converge to the optimal topology depicted in Fig. 1. For the starting point $x_i^{\{0\}} = \bar{v} = 0.5 \ \forall \ i$, algorithms GCMMA and GCDQ with curvatures (5) are unable to converge to reasonable local minima; these algorithms terminate very quickly at highly nonoptimal local solutions.

Again, of significance is an observation not explicitly shown: even though the variable thickness sheet design problem (obtained with $p = 1$) is strictly convex,[††] nonconservative algorithm DQ with curvatures (5) never terminates, but oscillates rather severely. That is, when a spherical quadratic approximation with direct design variables is used, convergence has to be enforced. [Nonconservative algorithm DQ with curvatures (9), however, performs well indeed, e.g., see our previous efforts [14].] Nonconservative algorithm MMA also converges relatively quickly, this being attributed to the reciprocal-like behavior of the objective function and the constraints.

F. Discussion of Numerical Results

The numerical results suggest that the enforcement of global convergence in SAO algorithms may imply a penalty when expressed in terms of computational effort. For most of the problems studied, both GCDQ and GCMMA required significantly higher computational effort than their nonconservative counterparts DQ and MMA. This probably explains the popularity of algorithms that are not guaranteed to converge for problems that require huge computational resources, for example, the topology optimization problem or any problem requiring the solution of large finite element or computational fluid dynamics models. However, this may come at some cost: both DQ and MMA did fail to converge for some problems.

Relaxed conservatism made little difference, if any, as compared to classical or strict conservatism for several of the test problems studied. However, for the first large nonconvex test problem, for example, the advantages of relaxed conservatism were huge. Because testing for relaxed conservatism requires negligible computational overhead, we recommend that it is used in the CCSA class of algorithms.

[††]The compliance depends linearly on the thickness ρ, for example, see [27].

Fig. 1 Optimum topology found with the GCDQ algorithm for the MBB beam, with $p = 3$ (75×25 mesh, $\epsilon_x = 10^{-4}$, and no continuation on p).

It is also noticed that the enforcement of (strict) convexity may sometimes be detrimental to convergence rate. This was elaborated upon in an augmented Lagrangian setting for the nonconvex variant DQ in [13].

Finally, a cautionary word on the results we have obtained with GCMMA: we have used the default settings present in GCMMA, but the performance of the algorithm may be different if other settings are used. Also note that we have not limited the maximum number of inner iterations allowed in both GCDQ and GCMMA, to ensure that the algorithms are indeed conservative. In practice, this need, in general, not be done.

VI. Conclusions

We have cast a sequential approximate optimization algorithm based on diagonal quadratic approximations in the conservative, convex, separable approximation (CCSA) algorithmic framework of Svanberg [10]. The resulting algorithm, denoted GCDQ, is globally convergent and related to the globally convergent version of the MMA algorithm (GCMMA); both algorithms reside in the CCSA class. In addition, both take advantage of a highly efficient dual formulation. The main difference between GCDQ and GCMMA lies in the approximations used: GCDQ uses diagonal quadratic approximations, versus the reciprocal-like approximations with asymptotes used in GCMMA.

The diagonal quadratic approximation is the simplest possible approximation within the CCSA framework. Approximations in the CCSA framework are convex and separable and contain two terms: one to ensure that first-order accuracy is guaranteed and a second term for stabilization of the iterations. In GCDQ, the first term is simply the linear approximation, whereas the second is estimated using a problem-specific heuristic. However, contrary to GCMMA, we use the second term not only for stabilization, but we also attempt to capture the local curvature of the true nonlinear functions by this term. In addition, in GCDQ, the higher-order part is also used to guarantee convexity of the approximations.

Another difference is that, in GCMMA, conservatism is enforced unconditionally. However, in GCDQ, conservatism is only enforced if a feasible descent step cannot be made; we have denoted this strategy relaxed conservatism.

We have also presented two different methods to estimate the higher-order curvatures at the start of each outer iteration in GCDQ (but many other possibilities exist). The first estimates the curvatures by assuming that the true functions are spherical quadratic, through use of the function values already evaluated at the previous iterate $\boldsymbol{x}^{\{k-1\}}$.

The second method estimates the curvatures by constructing the quadratic Taylor series expansion to the reciprocal approximation about the current iterate $\boldsymbol{x}^{\{k\}}$. This approximation is aimed at problems for which reciprocal-like behavior of the approximations is crucial to the convergence behavior of the SAO algorithm, for example, in minimum compliance topology optimization, etc.

We conclude that GCDQ is an efficient member of the CCSA class of algorithms. Because no single algorithm will always outperform all other algorithms for a wide range of problem classes, availability of a suite of algorithms (and tailorability of these algorithms) seems attractive. Further investigations into the CCSA type of approximations, the initialization of the outer loop iterations, and new mechanisms to reduce the computational cost induced by the CCSA inner loops are considered fruitful directions for future research.

Appendix: Relaxed Form for Problem P_{NLP}

Following Svanberg [2,10], we relax problem P_{NLP} to obtain

$$\min f_0(\boldsymbol{x}) + \sum_{j=1}^{m}\left(\theta_j y_j + \frac{1}{2}\phi_j y_j^2\right) \qquad \text{subject to } f_j(\boldsymbol{x}) - y_j \le 0$$

$$j = 1,\ldots,m \qquad \check{x}_i \le x_i \le \hat{x}_i, \qquad i = 1,\ldots,n \qquad y_j \ge 0$$

$$j = 1,\ldots,m \tag{A1}$$

For suitable values of θ_j, ϕ_j, it is always possible to obtain a feasible solution[‡‡] to Eq. (A1), because at least one Karush-Kuhn-Tucker point will exist. The relaxation considered here was used by Svanberg [2,10] in MMA. In this study, it is understood that relaxed problem (A1) is actually used whenever problem P_{NLP} is indicated. For the sake of brevity, this is, however, not reflected. Throughout, we have used $\theta_j = 10^3$ and $\phi_j = 1 \forall\ j$ in both GCMMA and GCDQ. A note on this is in order: in all the examples presented, the values used for θ_j and ϕ_j had negligible influence on the results generated with algorithm GCDQ. GCMMA, on the other hand, revealed some sensitivity to this. For the (convex) 10-bar truss example, $\theta_j \ge 10^5$ was required in MMA before a feasible solution could be obtained.

Acknowledgments

The authors would like to acknowledge the very helpful comments of the reviewers, as well as those made by J. A. Snyman of the University of Pretoria, South Africa.

References

[1] Barthelemy, J.-F. M., and Haftka, R. T., "Approximation Concepts for Optimum Structural Design: A Review," *Structural Optimization*, Vol. 5, No. 3, 1993, pp. 129–144. doi:10.1007/BF01743349

[2] Svanberg, K., "The Method of Moving Asymptotes: A New Method for Structural Optimization," *International Journal for Numerical Methods in Engineering*, Vol. 24, No. 2, 1987, pp. 359–373. doi:10.1002/nme.1620240207

[3] Fleury, C., and Braibant, V., "Structural Optimization: A New Dual Method Using Mixed Variables," *International Journal for Numerical Methods in Engineering*, Vol. 23, No. 3, 1986, pp. 409–428. doi:10.1002/nme.1620230307

[4] Snyman, J. A., and Hay, A. M., "The Dynamic-Q Optimization Method: An Alternative to SQP?," *Computers and Mathematics with Applications*, Vol. 44, No. 12, 2002, pp. 1589–1598. doi:10.1016/S0898-1221(02)00281-X

[5] Berke, L., and Venkayya, V. B., "Review of Optimality Criteria Approaches to Structural Optimization," American Society of Mechanical Engineers, Fairfield, NJ, 1974, pp. 49–64.

[6] Khot, N. S., "Algorithms Based on Optimality Criteria to Design Minimum Weight Structures," *Engineering Optimization*, Vol. 5, No. 2, 1981, pp. 73–90. doi:10.1080/03052158108902436

[7] Venkayya, V. B., "Optimality Criteria: A Basis for Multidisciplinary Optimization," *Computational Mechanics*, Vol. 5, No. 1, 1989, pp. 1–21. doi:10.1007/BF01046875

[8] Rozvany, G. I. N., *Structural Design via Optimality Criteria: The Prager Approach to Structural Optimization*, Kluwer Academic, Dordrecht, The Netherlands, 1989.

[9] Conn, A. R., Gould, N. I. M., and Toint, P. L., *Trust-Region Methods*, MPS/SIAM Series on Optimization, Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[10] Svanberg, K., "A Class of Globally Convergent Optimization Methods Based on Conservative Convex Separable Approximations," *SIAM Journal on Optimization*, Vol. 12, No. 2, 2002, pp. 555–573. doi:10.1137/S1052623499362822

[11] Svanberg, K., "On a Globally Convergent Version of MMA," *Proceedings of the Seventh World Congress on Structural and Multidisciplinary Optimization* A0052, May 2007.

[12] Groenwold, A. A., Etman, L. F. P., Snyman, J. A., and Rooda, J. E., "Incomplete Series Expansion for Function Approximation," *Structural and Multidisciplinary Optimization*, Vol. 34, No. 1, 2007, pp. 21–40. doi:10.1007/s00158-006-0070-6

[13] Groenwold, A. A., Etman, L. F. P., Kok, S., Wood, D. W., and Tosserams, S., "An Augmented Lagrangian Approach to Non-Convex SAO Using Diagonal Quadratic Approximations," *Structural and Multidisciplinary Optimization*, Vol. 38, No. 4, May 2009, pp. 415–421. doi:10.1007/s00158-008-0304-x

[14] Groenwold, A. A., and Etman, L. F. P., *A Quadratic Approximation for Structural Topology Optimization*, *International Journal for Numerical Methods in Engineering* (submitted for publication), July 2008.

[15] Groenwold, A. A., Etman, L. F. P., and Wood, D. W., Approximated Approximations for SAO, *Structural and Multidisciplinary Optimization* (to be published); available online 10 June 2009. doi:10.1007/s00158-009-0406-0

[16] Wilke, D. N., Kok, S., and Groenwold, A. A., The Application of Gradient-Only Optimization Methods for Problems Discretized Using Non-Constant Methods, *Structural and Multidisciplinary Optimization* (to be published); available online 6 May 2009. doi:10.1007/s00158-009-0389-x

[17] Wang, L., and Grandhi, R. V., "Improved Two-Point Function Approximation for Design Optimization," *AIAA Journal*, Vol. 33, No. 9, 1995, pp. 1720–1727. doi:10.2514/3.12715

[18] Xu, S., and Grandhi, R. V., "Effective Two-Point Function Approximation for Design Optimization," *AIAA Journal*, Vol. 36, No. 12, 1998, pp. 2269–2275. doi:10.2514/2.337

[19] Starnes, J. H., Jr., and Haftka, R. T., "Preliminary Design of Composite Wings for Buckling, Stress and Displacement Constraints," *Journal of Aircraft*, Vol. 16, No. 8, 1979, pp. 564–570. doi:10.2514/3.58565

[20] Falk, J. E., "Lagrange Multipliers and Nonlinear Programming," *Journal of Mathematical Analysis and Applications*, Vol. 19, No. 1, 1967, pp. 141–159. doi:10.1016/0022-247X(67)90028-5

[21] Groenwold, A. A., and Etman, L. F. P., "Sequential Approximate Optimization Using Dual Subproblems Based on Incomplete Series Expansions," *Structural and Multidisciplinary Optimization*, Vol. 36, No. 6, 2008, pp. 547–570. doi:10.1007/s00158-007-0197-0

[22] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J., "L-BFGS-B: FORTRAN Subroutines for Large Scale Bound Constrained Optimization," Electrical Engineering and Computer Science Dept., Northwestern Univ., TR NAM-11, 1994.

[23] Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C., "A Limited Memory Algorithm for Bound Constrained Optimization," *SIAM Journal on Scientific Computing*, Vol. 16, No. 5, 1995, pp. 1190–1208. doi:10.1137/0916069

[24] Haftka, R. T., and Gürdal, Z., *Elements of Structural Optimization*, Solid Mechanics and Its Applications Vol. 11, 3rd ed., Kluwer Academic, Dordrecht, The Netherlands, 1991.

[25] Svanberg, K., "A Globally Convergent Version of MMA Without Line Search," edited by G. I. N. Rozvany, and N. Olhoff, Pergamon Press, Oxford, England, U.K., 1995, pp. 9–16.

[26] Ringertz, U. T., "On Methods for Discrete Structural Optimization," *Engineering Optimization*, Vol. 13, No. 1, 1988, pp. 47–64. doi:10.1080/03052158808940946

[27] Bendsøe, M. P., and Sigmund, O., *Topology Optimization: Theory, Methods and Applications*, Springer, Berlin, 2003.

[28] Cook, R. D., Malkus, D. S., Plesha, M. E., and Witt, R. J., *Concepts and Applications of Finite Element Analysis*, Wiley, New York, 2002.

[29] Sigmund, O., "On the Design of Compliant Mechanisms Using Topology Optimization," *Mechanics of Structures and Machines*, Vol. 25, No. 4, 1997, pp. 495–526.

[30] Perez, V. M., Eldred, M. S., and Renaud, J. E., "Solving the Infeasible Trust-Region Problem Using Approximations," AIAA Paper No. 2004-4312, 2004.

[31] Etman, L. F. P., Groenwold, A. A., and Rooda, J. E., "Sequential Approximate Optimization in an NLP Filter Framework," AIAA Paper No. 2006-7078, 2006.

T. Zang
Associate Editor

[‡‡]Relaxation may be effected in many different manners, for example, see the normal-tangential step approach used by Perez et al. [30] and also the current authors [31].