On diagonal QP subproblems for sequential approximate optimization

L.F.P. Etman, Albert A. Groenwold, and J.E. Rooda

Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, the Netherlands,
l.f.p.etman@tue.nl, j.e.rooda@tue.nl
Department of Mechanical Engineering, University of Stellenbosch, Stellenbosch, South Africa,
albertg@sun.ac.za

1. Abstract
Sequential approximate optimization (SAO) methods aimed at structural optimization often use reciprocal-like approximations in a dual setting; the method of moving asymptotes (MMA) and CONLIN are important and telling examples. Using a different (if related) approach, we have previously demonstrated that replacement of the reciprocal approximation by its own quadratic or second order Taylor series expansion may result in equally efficient dual SAO algorithms.

In this paper, we show that the quadratic treatment is not only to the benefit of dual SAO statements, but may also make (sparse) sequential quadratic programming (SQP) methods viable for the solution of large-scale structural optimization problems. We solve several test problems by means of a series of diagonal Lagrange-Newton or QP subproblems generated from the quadratic treatment of the objective function and constraint function approximations. We conclude that the proposed SAO method using a QP statement is a very promising alternative when the Falk dual methods become expensive, which may for instance happen for problems with both the number of design variables and the number of constraints very high. As expected, dual methods however remain the most efficient for problems with only a few constraints.

2. Keywords: Diagonal quadratic approximations, reciprocal intervening variables, quadratic programming (QP) subproblems.

3. Introduction
Sequential approximate optimization (SAO) techniques based on nonlinear but convex separable approximation functions have proven to be very effective in structural optimization. Well-known examples are the convex linearization (CONLIN) [23, 12] algorithm, and the method of moving asymptotes (MMA) [25]. Both these algorithms are based on reciprocal or reciprocal-like approximations, and generate a series of convex nonlinear programming (NLP) approximate subproblems. The reciprocal-like approximations are obtained via the substitution of reciprocal intermediate variables into a first-order (linear) Taylor series expansion, and fairly accurately match the nonlinearity present in many an objective and constraint function encountered in structural optimization. They provide for reasonably accurate subproblem approximations without using Hessian information, while the separability and convexity of the objective and constraint function approximations may also be used to develop instances of the dual formulation of Falk [10]. The Falk dual is computationally very efficient, in particular for applications with many design variables and relatively few constraints.

Several other separable approximations based on intervening variables have been proposed, see for example the overviews provided in [2], [13], and [20]. The intervening variables yield approximations that are nonlinear and possibly non-convex in terms of the original or direct variables. The resulting subproblems are typically solved in their primal form by means of an appropriate mathematical programming algorithm. Even though the subproblem may be separable, either the non-convexity or the inability to arrive at an analytical primal-dual relation may hinder the utilization of the Falk dual. This partly explains why these often highly accurate nonlinear approximations, are not as widely used in large-scale structural optimization as the reciprocal type of approximations.

Recently, we have found that (diagonal) quadratic Taylor series approximations to the original nonlinear approximations may perform equally well, or sometimes even better, than subproblems based on the original nonlinear approximations [16, 19, 17]. In [16], we demonstrate that the diagonal quadratic approximation to the reciprocal and exponential approximate objective functions can successfully be used
in topology optimization. In [19, 17], this observation is generalized even further: diagonal quadratic approximations to arbitrary nonlinear approximate objective and constraint functions are built; the resulting subproblems are convexified (if necessary), and cast into the dual statement of Falk. The form of the dual subproblem does not depend on the specific nonlinear approximations or intervening variables used, and all the approximated approximations can be used simultaneously in a single dual statement. In [19, 17], quadratic approximations to the reciprocal, exponential, CONLIN, MMA, and TANA approximations are developed.

The quadratic treatment of nonlinear approximations based on intervening variables is not only to the benefit of dual SAO statements but may also make sequential quadratic programming (SQP) methods viable for the solution of large-scale structural optimization problems. SQP methods build a Hessian of second-order derivative information of the Lagragian function. The storage and update of this Hessian matrix becomes burdensome for large-scale structural optimization problems with many design variables, such as in topology optimization. For this reason, [11] proposes an SQP method with a diagonal Hessian and develops methods to efficiently calculate the diagonal second order derivatives in a finite element environment. [8] studies quasi-Newton and finite difference estimates to the diagonal Hessian terms, using historic information generated at the previously visited iterate. A related approach to estimate diagonal Hessian terms for quadratic approximations using historic information is the spherical quadratic approximation [24].

By means of quadratic Taylor series approximations to nonlinear approximations based on some intervening variable, it is possible to develop a (crude) estimate of the diagonal Hessian information (curvatures) without using historic information or an exact calculation of the diagonal Hessian terms. The diagonal curvature estimates directly follow from the selected intervening variables, which means that at every given iteration point, only function value and gradient information is required, like in most popular primal and dual SAO methods.

The idea explored herein is to use the estimated curvatures in an SQP-like approach, generating a series of diagonal Lagrange-Newton QP subproblems. The distinct advantage of formulating the approximate subproblem as a (convex) QP problem is that QPs are typically solvable for very many design variables and constraints. (Note that our proposed SQP approach is different to a two-level approach [30] which generates an inner loop of QP subproblems to solve the nonlinear intervening-variables based NLP subproblems defined in the outer iteration loop.)

We then investigate the computational efficiency of SAO based on a series of diagonal QP subproblems, generated by replacing the reciprocal approximation to the objective and/or the constraint functions by its own quadratic approximation. We also compare the resulting method (herein denoted SAO-QP) with the diagonal quadratic dual method we have previously proposed [19, 17], which generates a series of dual bound-constrained subproblems (herein to be denoted dual-SQP).

Our paper is outlined as follows: in Section 4, we derive the Falk dual and QP subproblems using the diagonal quadratic approximations. Furthermore, we outline how the second order curvatures can be obtained from reciprocal approximations. In Section 5, we present numerical results for three large-scale (structural) optimization problems: the cam design problem, Vanderplaats’ multi-segmented cantilever beam, and minimum compliance topology optimization. Concluding remarks are offered in Section 6.

4. SAO using diagonal quadratic approximations

4.1. Problem statement
We consider the inequality constrained (structural) optimization problem

\[
\begin{align*}
\min & \quad f_0(x) \\
\text{subject to} & \quad f_j(x) \leq 0, \quad j = 1, 2, \ldots, m, \\
& \quad \bar{x}_i \leq x_i \leq \check{x}_i, \quad i = 1, 2, \ldots, n,
\end{align*}
\]

(1)

where \(f_0(x)\) is a real valued scalar objective function, and the \(f_j(x), j = 1, 2, \ldots, m\) are inequality constraint functions. \(f_0(x)\) and \(f_j(x)\) depend on the \(n\) real (design) variables \(x = \{x_1, x_2, \ldots, x_n\}^T \in \mathbb{R}^n\). \(\bar{x}_i\) and \(\check{x}_i\) respectively indicate lower and upper bounds of continuous real variable \(x_i\). Evaluation of the \(f_j(x), j = 0, 1, \ldots, m\), is assumed to require an expensive numerical analysis.

4.2. Sequential approximate optimization
Sequential approximate optimization as a solution strategy for problem (1) seeks to construct succes-
sive approximate subproblems $P[k]$, $k = 1, 2, 3, \ldots$ at successive iteration points $x^{(k)}$. That is, we seek suitable (analytical) approximation functions $\hat{f}_j$ that are inexpensive to evaluate. The solution of subproblem $P[k]$ is denoted $x^{(k+1)} \in \mathbb{R}^n$, to be obtained using any suitable continuous programming method. The minimizer of subproblem $P[k]$ is $x^{(k+1)}$, to become the starting point $x^{(k+1)}$ for the subsequent approximate subproblem $P[k + 1]$.

To guarantee that the sequence of iteration points approaches the solution $x^*$, one may cast the SAO method used into a trust region framework [1], or in the framework of conservative convex separable approximations (CCSA) [26]. In the trust region framework, an allowed search domain is defined around $x^{(k)}$, and incorporated via additional bound constraint(s) into the approximate subproblem formulation. The size of the search subregion can be increased and decreased iteratively to enforce termination. Alternatively, in the CCSA framework, the conservatism of the objective and constraint function approximations can be increased to enforce convergence. This implies that $x^{(k+1)}$ is only conditionally accepted to become the new iterate. If $x^{(k+1)}$ is rejected, the subproblem is re-solved with a reduced trust region, or with increased conservatism.

For the sake of clarity, the results presented herein will be generated using a constant infinity norm trust region. In fact, $\tilde{x}_i$ and $\hat{x}_i$ are set equal to the lower and upper bounds of the trust region hyper-box, and we allow for the unconditional acceptance of iterates.

4.3. Diagonal quadratic function approximations

To build the approximate subproblems, we generate diagonal quadratic Taylor approximations $\hat{f}_j$ to the objective function and constraint functions $f_j$, such that

$$\hat{f}_j(x) = f_j(x^{(k)}) + \sum_{i=1}^{n} \left( \frac{\partial f_j}{\partial x_i} \right) x_i^{(k)} - x_i^{(k)} + \frac{1}{2} \sum_{i=1}^{n} c_{2i}^{(k)} (x_i - x_i^{(k)})^2,$$  \hspace{1cm} (2)

for $j = 0, 1, 2, \ldots, m$ and with the $c_{2i}^{(k)}$ the approximate second order diagonal Hessian terms or curvatures.

4.4. Dual subproblem

In the first approximate optimization subproblem we will consider, we replace the true objective function $f_0$, as well as all the true constraint functions $f_j$, $j = 1, 2, \ldots, m$ in (1) by the diagonal quadratic approximation functions $\hat{f}_j$ given by (2). The primal problem then has $n$ unknowns, $m$ constraints, and $2n$ side or bound constraints. We enforce strict convexity of every subproblem $P[k]$ by setting

$$c_{2i}^{(k)} = \max\{c_0 > 0, c_{2i}^{(k)}\},$$
$$c_{2i}^{(k)} = \max\{0, c_{2i}^{(k)}\}, \quad j = 1, 2, \ldots, m.$$  \hspace{1cm} (3)

We then invoke the dual of Falk [10] to obtain the following approximate dual subproblem $P_D[k]$:

$$\max\{\gamma(\lambda) = \hat{f}_0^{(k)}(x(\lambda)) + \sum_{j=1}^{m} \lambda_j \hat{f}_j^{(k)}(x(\lambda))\},$$
subject to $\lambda_j \geq 0, \quad j = 1, 2, \ldots, m.$  \hspace{1cm} (4)

with $\hat{f}_0^{(k)}$ and $\hat{f}_j^{(k)}, \quad j = 1, \ldots, m$, the diagonal quadratic approximations at iteration point $x^{(k)}$, and the primal-dual relationship between variables $x_i, i = 1, 2, \ldots, n$ and $\lambda_j, j = 1, 2, \ldots, m$ given by:

$$x_i(\lambda) = \begin{cases} \beta_i(\lambda) & \text{if } \tilde{x}_i^{(k)} < \beta_i(\lambda) < \hat{x}_i^{(k)}, \\
\tilde{x}_i^{(k)} & \text{if } \beta_i(\lambda) \leq \tilde{x}_i^{(k)}, \\
\hat{x}_i^{(k)} & \text{if } \beta_i(\lambda) \geq \hat{x}_i^{(k)}, \end{cases}$$  \hspace{1cm} (5)

with

$$\beta_i(\lambda) = \tilde{x}_i^{(k)} - \left( c_{2i0}^{(k)} + \sum_{j=1}^{m} \lambda_j c_{2ij}^{(k)} \right)^{-1} \left( \frac{\partial f_0^{(k)}}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial f_j^{(k)}}{\partial x_i} \right).$$  \hspace{1cm} (6)

For details, the reader is referred to our previous efforts [14].
This bound constrained problem requires the determination of the \( m \) unknowns \( \lambda_j \), subject to \( m \) non-negativity constraints on the \( \lambda_j \). Clearly, this is advantageous if \( m \ll n \). The subproblems may be solved using any suitable first order or second order method able to handle discontinuous second derivatives; we have used l-BFGS-b [31]. If the subproblems happen to be infeasible, we actually use a bounded form [29] of dual approximate subproblem \( P_l[k] \), but relaxation [25, 26, 18] may equally well be used.

4.5. Approximate subproblem in QP form
Since the approximations (2) are (diagonal) quadratic, the subproblems are easily recast as a quadratic program \( P_{QP}[k] \), written as
\[
egin{align*}
\min_s \quad & f_0^{(k)}(s) = f_0^{(k)} + \nabla^T f_0^{(k)} s + \frac{1}{2} s^T Q^{(k)} s \\
\text{subject to} \quad & f_j^{(k)}(s) = f_j^{(k)} + \nabla^T f_j^{(k)} s \leq 0, \quad j = 1, 2, \ldots, m, \\
& \bar{x}_i \leq x_i \leq \hat{x}_i, \quad i = 1, 2, \ldots, n,
\end{align*}
\]
with \( s = (x - x^{(k)}) \) and \( Q^{(k)} \) the Hessian matrix of the approximate Lagrangian.

Using the diagonal quadratic objective function and constraint function approximations \( f_j, j = 0, 1, \ldots, m \), the approximate Lagrangian \( \mathcal{L}^{(k)} \) is
\[
\mathcal{L}^{(k)} = f_0(x^{(k)}) + \sum_{j=1}^{m} \lambda_j^{(k)} f_j(x^{(k)}),
\]
This gives:
\[
Q_{ii} = c_{2i}^{(k)} + \sum_{j=1}^{m} \lambda_j^{(k)} c_{2ij},
\]
and \( Q_{il} = 0 \) \( \forall i \neq l \), \( i, l = 1, 2, \ldots, n \). As for the dual subproblem, we apply convexity conditions (3), to arrive at a strictly convex QP subproblem with a unique minimizer.

The quadratic programming problem requires the determination of the \( n \) unknowns \( x_i \), subject to \( m \) linear inequality constraints. Efficient QP solvers can typically solve problems with a very large numbers of design variables \( n \) and constraints \( m \). Obviously, it is imperative that the diagonal structure of \( Q \) is exploited when the QP subproblems are solved. Hence, we will use the commercial Numerical Algorithms Group (NAG) QP solver e04nqf.

4.6 Estimating the higher order curvatures
Key to our approach is to obtain the approximate higher order curvatures \( c_{2i}^{(k)} \) without the user providing or the algorithm storing explicit second order derivative information. To do so, a very simple strategy is to use the so-called spherical quadratic approximation, for which the development is as follows: select \( c_{2i}^{(k)} \) \( \equiv c_{2i}^{(k)} \) \( \forall i \). This requires the determination of the single unknown \( c_{2i}^{(k)} \), easily obtained by enforcing (for example) the condition
\[
f_j(x^{(k-1)}) = f_j(x^{(k)}),
\]
which implies that
\[
c_{2j}^{(k)} = \frac{2[f_j(x^{(k-1)}) - f_j(x^{(k)}) - \nabla^T f_j(x^{(k)})(x^{(k-1)} - x^{(k)})]}{||x^{(k-1)} - x^{(k)}||^2_2}.
\]
Snyman and Hay [24] have used nonconvex forms of the spherical quadratic approximation. In this paper we use strictly convex forms be enforcing convexity conditions (3). Other ‘classical’ strategies for estimating the \( c_{2i}^{(k)} \) may be found in [13].

To estimate the approximate higher order curvatures, we have recently also proposed a novel approach [17, 19] in which we generate diagonal quadratic approximations to a nonlinear first-order approximation based on some arbitrary intervening variable. For several structural optimization applications, it is known that intervening variables may yield nonlinear approximations of reasonable accuracy. Consider the first-order Taylor series expansion in terms of intervening variables \( y_i(x_i) \)
\[
\tilde{f}(y) = f(y^{(k)}) + \sum_{i=1}^{n} \frac{\partial f^{(k)}}{\partial y_i}(y_i - y_i^{(k)}),
\]

4
where we assume that \( y_i(x_i) \) is a continuous and monotonic expression of intervening variable \( y_i \) in terms of direct variables \( x_i \).

For the second order terms, (diagonal) quadratic approximation (2) to the first-order expansion (12) yields

\[
 c_{2i}^{(k)} = \left( \frac{\partial^2 \tilde{f}_1}{\partial x_i^2} \right)^{(k)} = \left( \frac{\partial f}{\partial x_i} \right)^{(k)} \left( \frac{\partial x_i}{\partial y_i} \right)^{(k)} \left( \frac{\partial^2 y_i}{\partial x_i^2} \right)^{(k)},
\]

(13)

For the popular reciprocal intervening variables \( y_i = 1/x_i \), we obtain

\[
 c_{2i}^{(k)} = \frac{-2}{x_i^{(k)}} \left( \frac{\partial f}{\partial x_i} \right)^{(k)}.
\]

(14)

In other words: using the approximate curvatures (14) in (2) implies that (2) becomes the quadratic approximation to the reciprocal approximation in the point \( x^{(k)} \).

For other diagonal quadratic approximations to a number of different well known approximations, including the exponential, TANA, and MMA approximations, see [17, 19].

5. Numerical results

We now compare the numerical performances of an SAO algorithm based on Falk dual subproblem (4) and an SAO algorithm based on QP subproblem (7) for three test problems. We respectively denote the two methods by 'dual-SAO' and 'SAO-QP'. We terminate the iterations when \( \| x^{(k)} - x^{(k-1)} \| \leq \epsilon \), where \( \| \cdot \| \) indicates the Euclidian or 2-norm, and we have used a 20\% move limit throughout (but it often is advantageous for QP problems to not do so).

Both the Falk dual-SAO implementation and the SAO-QP implementation are based on BLAS. Results were generated on a single core of a 64-bit 3.73 GHz Intel-based Dell machine with 8 GB of memory; the operating system was OpenSuse 11.1, while the code was implemented in double-precision FORTRAN, and we have used the open-source FORTRAN-4.2 complier.

We define \( h = \max\{f_j\}, j = 1, 2, \cdots, m \), while \( l^* \) and \( u^* \) respectively indicate the number of design variables on the lower and upper bounds at the solution \( x^* \). \( k^* \) denotes the required number of iterations. The reported CPU effort is in seconds. Finally, \% S-S denotes the percentage of the total CPU time spent on solving the subproblems. For CPU < 0.05, we do not report \% S-S since the time keeping is not accurate enough, and also not sufficiently repeatable).

5.1 Cam design problem

This problem [7] aims to maximize the area of a valve opening for one rotation of a convex cam, with constraints on the curvature and on the radius of the cam.

We assume that the shape of the cam is circular over an angle of \( 6/5\pi \) of its circumference, with radius \( r_{\text{min}} \). The design variables \( r_i, i = 1, \ldots, n \), represent the radius of the cam at equally spaced angles distributed over an angle of \( 2/5\pi \). We maximize the area of the valve opening by maximizing

\[
 f_0(r) = \pi r_v^2 \left( \frac{1}{n} \sum_{i=1}^{n} r_i \right),
\]

subject to constraints on the \( r_i \). The design parameter \( r_v \) is related to the geometry of the valve. We also require that \( r_{\text{min}} \leq r_i \leq r_{\text{max}} \) \( \forall \ i \). The requirement that the cam be convex is expressed by requiring that

\[
 2r_{i-1}r_{i+1} \cos(\theta) - r_i(r_{i-1} + r_{i+1}) \leq 0, \quad i = 0, \ldots, n + 1,
\]

where \( r_{-1} = r_0 = r_{\text{min}}, r_{n+1} = r_{\text{max}}, r_{n+2} = r_n, \) and \( \theta = 2\pi/5(n+1) \). The curvature requirement is expressed by requiring

\[
 -\alpha - \left( \frac{r_{i+1} - r_i}{\theta} \right) \leq 0, \quad i = 0, \ldots, n,
\]

and

\[
 \left( \frac{r_{i+1} - r_i}{\theta} \right) - \alpha \leq 0, \quad i = 0, \ldots, n.
\]

Our formulation results in \( 3n + 4 \) constraints; this is a slight departure from [7], where \( 2n \) constraint were formulated (our implementation requires that the constraints are written in negative-null form). We use
Table 1: The cam design problem: computational effort for SAO-QP

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>k</th>
<th>f₀*</th>
<th>h*</th>
<th>CPU</th>
<th>% S-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>19</td>
<td>6</td>
<td>4.49332</td>
<td>0.00 × 10⁻⁰⁰</td>
<td>0.00</td>
<td>—</td>
</tr>
<tr>
<td>10</td>
<td>34</td>
<td>6</td>
<td>4.38600</td>
<td>4.41 × 10⁻¹⁵</td>
<td>0.01</td>
<td>—</td>
</tr>
<tr>
<td>20</td>
<td>64</td>
<td>5</td>
<td>4.32794</td>
<td>0.00 × 10⁻⁰⁰</td>
<td>0.01</td>
<td>—</td>
</tr>
<tr>
<td>40</td>
<td>124</td>
<td>5</td>
<td>4.30095</td>
<td>8.82 × 10⁻¹⁵</td>
<td>0.01</td>
<td>—</td>
</tr>
<tr>
<td>80</td>
<td>244</td>
<td>5</td>
<td>4.28694</td>
<td>1.23 × 10⁻¹²</td>
<td>0.02</td>
<td>—</td>
</tr>
</tbody>
</table>

Table 2: The cam design problem: computational effort for dual-SAO

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>k</th>
<th>f₀*</th>
<th>h*</th>
<th>CPU</th>
<th>% S-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>19</td>
<td>15</td>
<td>4.49331</td>
<td>1.64 × 10⁻⁰⁸</td>
<td>0.04</td>
<td>—</td>
</tr>
<tr>
<td>10</td>
<td>34</td>
<td>15</td>
<td>4.38600</td>
<td>4.84 × 10⁻⁰⁸</td>
<td>0.16</td>
<td>94.87</td>
</tr>
<tr>
<td>20</td>
<td>64</td>
<td>49</td>
<td>4.32794</td>
<td>1.83 × 10⁻⁰⁷</td>
<td>1.01</td>
<td>96.79</td>
</tr>
<tr>
<td>40</td>
<td>124</td>
<td>240</td>
<td>4.30096</td>
<td>1.01 × 10⁻⁰⁷</td>
<td>10.28</td>
<td>98.79</td>
</tr>
<tr>
<td>80</td>
<td>244</td>
<td>1673</td>
<td>4.28546</td>
<td>2.98 × 10⁻⁰⁶</td>
<td>183.64</td>
<td>99.20</td>
</tr>
</tbody>
</table>

Table 3: The cam design problem: computational effort for SAO-QP with n and m large

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>k</th>
<th>f₀*</th>
<th>h*</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>2404</td>
<td>5</td>
<td>4.27427</td>
<td>1.99 × 10⁻¹³</td>
<td>3.28</td>
</tr>
<tr>
<td>1000</td>
<td>3004</td>
<td>6</td>
<td>4.27399</td>
<td>9.94 × 10⁻¹²</td>
<td>6.77</td>
</tr>
<tr>
<td>1200</td>
<td>3604</td>
<td>7</td>
<td>4.27380</td>
<td>4.79 × 10⁻¹²</td>
<td>12.76</td>
</tr>
</tbody>
</table>

Numerical results are presented in Tables 1 and 2 for SAO-QP and dual-SAO respectively. We have used εₓ = 10⁻⁴, and spherical quadratic approximations for the objective function f₀ and all of the 3n + 4 constraint functions f_j. The results emphasize that the SAO-QP is far superior to dual-SAO, presumably largely since m ≫ n.

Next, in Table 3, we present results for SAO-QP for n = 800, 1000, 1200 respectively. For these settings, Reference [7] reports results for several state-of-the-art SQP solvers. The reported CPU times are of the same order of magnitude as those obtained with SAO-QP (not explicitly shown). Note however that the results in [7] were obtained using an Intel 1.8GHz machine.

5.2 Vanderplaats’ cantilever beam

Consider the optimal sizing design of the tip-loaded multi-segmented cantilever beam proposed by Vanderplaats [27]. The beam is of fixed length l, is divided into p segments, and is subject to geometric, stress and a single displacement constraint. The geometry has been chosen such that a very large number of the constraints are active or ‘near-active’ at the optimum.

The objective function is formulated in terms of the design variables b_i and h_i as

\[ \min f₀(b, h) = \sum_{i=1}^{k} b_i h_i l_i, \]

with l_i constant for given k. We enforce the bound constraints 1.0 ≤ b_i ≤ 80, and 5.0 ≤ h_i ≤ 80 (the
upper bounds were arbitrarily chosen; they are required in the dual formulation, and also needed to allow for the notion of a ‘move limit’). The stress constraints are

$$\frac{\sigma(b,h)}{\bar{\sigma}} - 1 \leq 0, \quad i = 1, 2, p,$$

while the linear geometric constraints are written as

$$h_i - 20b_i \leq 0, \quad i = 1, 2, p.$$

The tip displacement constraint is

$$\frac{u_{\text{tip}}(b,h)}{\bar{u}} - 1 \leq 0.$$

The constraints are rather easily written in terms of the design variables $b$ and $h$, e.g. see Vanderplaats [27]. Note that the constraints are normalized; this is sound practice in primal algorithms. However, this may not be desirable in algorithms based purely on dual statements. We therefore scale the last constraint by $10^3$ (to have all the dual variables of roughly the same order; this is easily determined for the problems of low dimensionality).

Using consistent units, the geometric and problem data are as follows [27]: we use a tip load of $P = 50000$, a modulus of elasticity $E = 2 \times 10^7$, a beam length $l = 500$, while $\bar{\sigma} = 14000$, and $\bar{u} = 2.5$. The starting point is $b_i = 5.0$ and $h_i = 60$ for all $i$, while $\epsilon_x = 10^{-5}$. We use the quadratic approximation to the reciprocal approximation for the objective and all the constraint values. (Since the objective is linear, even better results may be obtained by spherical quadratic approximations for the objective).

The problem is expressed in terms of $2p$ design variables, and $2p + 1$ constraints. The lessons learned from the cam design problem may seem to suggest that SAO-QP will again be superior, since $m \approx n$. However, this is not the case: this time, dual-SAO is superior (although not overly so), see Tables 4 and 5 for respectively SAO-QP and dual-SAO. However, not shown is that the computational effort required with SAO-QP is rather insensitive to the scaling of the constraints; for the Falk dual subproblems, ineffective scaling can increase the computational effort by up to an order of magnitude(!)

<table>
<thead>
<tr>
<th>$k$</th>
<th>$n$</th>
<th>$m$</th>
<th>$k^*$</th>
<th>$f_0^*$</th>
<th>$h^*$</th>
<th>CPU</th>
<th>% S-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>50</td>
<td>51</td>
<td>14</td>
<td>63781.07</td>
<td>1.47 $\times 10^{-11}$</td>
<td>0.03</td>
<td>—</td>
</tr>
<tr>
<td>50</td>
<td>100</td>
<td>101</td>
<td>14</td>
<td>63704.47</td>
<td>1.36 $\times 10^{-12}$</td>
<td>0.05</td>
<td>—</td>
</tr>
<tr>
<td>100</td>
<td>200</td>
<td>201</td>
<td>14</td>
<td>63678.10</td>
<td>2.13 $\times 10^{-14}$</td>
<td>0.12</td>
<td>75.86</td>
</tr>
<tr>
<td>200</td>
<td>400</td>
<td>401</td>
<td>14</td>
<td>63669.20</td>
<td>2.27 $\times 10^{-13}$</td>
<td>0.40</td>
<td>91.09</td>
</tr>
<tr>
<td>400</td>
<td>800</td>
<td>801</td>
<td>15</td>
<td>63666.01</td>
<td>6.40 $\times 10^{-14}$</td>
<td>2.25</td>
<td>93.96</td>
</tr>
<tr>
<td>800</td>
<td>1600</td>
<td>1601</td>
<td>15</td>
<td>63665.34</td>
<td>9.95 $\times 10^{-13}$</td>
<td>17.13</td>
<td>96.75</td>
</tr>
<tr>
<td>1600</td>
<td>3200</td>
<td>3201</td>
<td>15</td>
<td>63665.17</td>
<td>6.39 $\times 10^{-14}$</td>
<td>89.35</td>
<td>97.73</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$k$</th>
<th>$n$</th>
<th>$m$</th>
<th>$k^*$</th>
<th>$f_0^*$</th>
<th>$h^*$</th>
<th>CPU</th>
<th>% S-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>50</td>
<td>51</td>
<td>14</td>
<td>63781.07</td>
<td>4.28 $\times 10^{-06}$</td>
<td>0.02</td>
<td>—</td>
</tr>
<tr>
<td>50</td>
<td>100</td>
<td>101</td>
<td>14</td>
<td>63704.47</td>
<td>2.99 $\times 10^{-05}$</td>
<td>0.05</td>
<td>—</td>
</tr>
<tr>
<td>100</td>
<td>200</td>
<td>201</td>
<td>14</td>
<td>63678.10</td>
<td>1.80 $\times 10^{-05}$</td>
<td>0.14</td>
<td>94.29</td>
</tr>
<tr>
<td>200</td>
<td>400</td>
<td>401</td>
<td>15</td>
<td>63669.20</td>
<td>1.37 $\times 10^{-05}$</td>
<td>0.47</td>
<td>94.92</td>
</tr>
<tr>
<td>400</td>
<td>800</td>
<td>801</td>
<td>15</td>
<td>63666.01</td>
<td>5.23 $\times 10^{-06}$</td>
<td>2.04</td>
<td>92.53</td>
</tr>
<tr>
<td>800</td>
<td>1600</td>
<td>1601</td>
<td>15</td>
<td>63665.34</td>
<td>6.67 $\times 10^{-06}$</td>
<td>9.60</td>
<td>94.50</td>
</tr>
<tr>
<td>1600</td>
<td>3200</td>
<td>3201</td>
<td>16</td>
<td>63665.17</td>
<td>1.27 $\times 10^{-06}$</td>
<td>32.73</td>
<td>93.50</td>
</tr>
</tbody>
</table>
Table 6: The MBB beam: average computational effort per iteration for SAO-QP (terminated after 100 iterations)

<table>
<thead>
<tr>
<th>mesh</th>
<th>$n$</th>
<th>$f_0^*$</th>
<th>$f_1^*$</th>
<th>$l^*$</th>
<th>$u^*$</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>$75 \times 25$</td>
<td>1875</td>
<td>204.35</td>
<td>$-1.66 \times 10^{-15}$</td>
<td>491</td>
<td>674</td>
<td>2.714</td>
</tr>
<tr>
<td>$150 \times 50$</td>
<td>7500</td>
<td>208.78</td>
<td>$-2.11 \times 10^{-16}$</td>
<td>1811</td>
<td>2537</td>
<td>62.204</td>
</tr>
<tr>
<td>$300 \times 100$</td>
<td>30000</td>
<td>211.45</td>
<td>$-8.31 \times 10^{-16}$</td>
<td>7191</td>
<td>9978</td>
<td>537.386</td>
</tr>
</tbody>
</table>

Table 7: The MBB beam: average computational effort per iteration for dual-SAO (terminated after 100 iterations)

<table>
<thead>
<tr>
<th>mesh</th>
<th>$n$</th>
<th>$f_0^*$</th>
<th>$f_1^*$</th>
<th>$l^*$</th>
<th>$u^*$</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>$75 \times 25$</td>
<td>1875</td>
<td>204.35</td>
<td>3.32 $\times 10^{-13}$</td>
<td>491</td>
<td>674</td>
<td>0.602</td>
</tr>
<tr>
<td>$150 \times 50$</td>
<td>7500</td>
<td>208.78</td>
<td>$-3.11 \times 10^{-08}$</td>
<td>1811</td>
<td>2537</td>
<td>2.086</td>
</tr>
<tr>
<td>$300 \times 100$</td>
<td>30000</td>
<td>211.45</td>
<td>6.78 $\times 10^{-13}$</td>
<td>7191</td>
<td>9978</td>
<td>14.426</td>
</tr>
</tbody>
</table>

5.3 Minimum compliance topology optimization
Finally, we study the minimum compliance topology optimization MBB problem described by, for example Bendsøe and Sigmund [5], see Figure 1. Using the popular SIMP (solid isotropic material with penalization) material model [4, 21] the problem can be expressed as:

$$
\begin{align*}
\min_{\rho} f_0(\rho) &= q^T K q = \sum_{i=1}^{n_{el}} (\rho_i)^p q_i^e K_0^p q_i^e, \\
\text{subject to } & f_1 \equiv \nu - \bar{\nu} = \sum_{i=1}^{n_{el}} \rho_i^e \nu_i^e - \bar{\nu} \leq 0, \\
& K q = r, \\
& 0 < \rho \leq 1,
\end{align*}
$$

where $c$ represents compliance, $q = q(\rho)$ and $r$ represent the global assembled finite element displacement and force vectors, while $K = K(\rho)$ represents the global finite element stiffness matrix. There are $n_{el}$ finite elements; superscript $e$ indicates elemental quantities. $\rho$ is the vector of elemental densities (the design variables), while $\nu$ represents the structural volume, with $\bar{\nu}$ a prescribed limit on volume.

![Figure 1: The MBB beam (unit thickness; plane stress)](image)

In our SAO approach to address this topology optimization problem, we will use the quadratic replacement to the reciprocal approximation for the nonlinear approximate objective function $f_0(\rho)$, and direct variables for the (linear) constraint function $f_1(\rho)$. Note that direct application of the ever-popular optimality criterion (OC) method to the topology optimization problem, combined with a heuristic updating scheme based on a damping factor of 0.5, results in a formulation which is identical to a rudimentary dual SAO algorithm with subproblems based on reciprocal intervening variables [5, 15].

We use $\rho_i = 0.001 \forall i$, a prescribed volume fraction $\bar{\nu} = 0.5$, a constant move limit $\delta_{\infty} = 0.2$, and we select $\rho_i^{(0)} = \bar{\nu} = 0.5 \forall i$. Sigmund’s filter strategy is used [22], with the filter radius taken as 8% of the
Figure 2: Optimal topologies obtained for the MBB beam, using continuation on $p$. (There are no visible differences between results obtained with dual-SAO and SAO-QP, since the constraints are linear)

beam height. The optimization runs are terminated after 100 iterations. All discretizations reported are for half the beam, due to symmetry. We use standard isoparametric displacement based four node finite elements (e.g. see [6]).

Since the classical minimum compliance topology optimization problem yields a nonlinear objective function and a linear constraint, dual subproblems (4) and QP subproblem 7 are identical, and indeed yield identical iteration paths (to machine precision). This is evident from Tables 6 and 7, for respectively SAO-QP and dual-SAO. The interesting aspect is the computational effort (we report here the average computational effort per iteration for each algorithm; the reported CPU times exclude the time to evaluate the objective and the single constraint function and the derivatives thereof).

As expected, dual-SAO is by far superior to SAO-QP, in that it is far less expensive to solve the convex dual than the QP subproblems, since there is only one constraint, thereby confirming the long tradition of using dual methods in classical minimum compliance topology optimization.

Finally, the optimal topologies obtained for the three mesh discretizations considered are depicted in Table 2; these are obviously independent of mesh refinement.

6. Conclusion
We have proposed an SAO method based on a series of diagonal Lagrange-Newton or QP subproblems, as an alternative to the popular dual SAO methods in structural optimization, and we have shown that it is possible to estimate the diagonal Hessian information by means of quadratic Taylor series approximations to the intervening variables-based nonlinear approximations commonly used in the dual approaches. This means that at every given iterate, only function value and gradient information is used and no Hessian information is stored or updated. The computational efficiency of the proposed method, denoted SAO-QP, was investigated for several test problems and compared with the related dual SAO method that is based on diagonal quadratic approximations of the objective and the constraint functions.

We conclude that the proposed SAO-QP method is a very promising alternative when the commonly used Falk dual methods become very expensive. This may for instance happen in structural optimization problems for which both the number of design variables and the number of constraints are very high. As expected, the dual method remains the most efficient for problems with only a few constraints. The success achieved in solving large-scale structural optimization problems by means of the proposed diagonal QP approach may also shed new light on efforts to make mathematical trust-region SQP methods effective for large-scale structural optimization problems. We hope to address this in future research.

7. Acknowledgments
This work was in part done during the second author’s visit to Eindhoven in October and November 2008. He would like to express his sincerest gratitude towards his hosts for accommodating him. This research was supported by the Dutch Technology Foundation STW, applied science division of NWO and the Technology Program of the Ministry of Economic Affairs of the Netherlands.

8. References


