Stable computation of interior point solutions for a class of nonlinear convex programming problems

Christian Zillober
Mathematisches Institut, Universität Bayreuth
D-95440 Bayreuth, Germany
Christian.Zillober@Uni-Bayreuth.De

Abstract: In the application of an interior point method to certain convex smooth nonlinear programming problems numerical difficulties in the neighborhood of the solution had to be observed. The reason are cancellation errors in the computation of active components. These errors are not critical for the computation of the variables themselves, but they are also used for some right hand sides in subsequent computations and then lead to problems in stable computation of a solution. There are known techniques to eliminate such critical variables. In this paper, a new possibility to remove this problem is proposed, with the advantage that knowledge on the corresponding dual variables is not lost. It is outlined that an underlying implementation can be changed easily to handle the problem above.

1 Introduction

Applying any of the modern interior point methods to a convex programming problem, users have to deal with a more and more growing instability in computing new steps when the sequence of iteration points approaches optimality. The reasons are mainly cancellation errors and additionally increasing condition number of the linear system that defines the search direction.
In this paper we propose to eliminate the components of critical variables that converge to 0 to avoid the fact that cancellation errors lead subsequently to erroneous right hand sides and in consequence to wrong results. As a second effect the growing of the condition number of the linear systems, that have to be solved in each iteration, will be bounded. The increasing condition number itself does not automatically lead to numerical problems if a stable linear system solver is used and the necessary accuracy of the solution is not too high. It is important to notice that the cancellation errors described below have nothing to do with inexact linear system solutions. The errors would also appear if we would have arithmetically exact linear system solutions.
We focus on the numerical stabilization of a single step of the primal-dual interior point method in a certain neighborhood of the solution. It is not claimed that the proposed method is the only possibility to improve stability but it is a quite natural way.
We will also answer the question, what has to be done if the convergence of a variable was

only temporarily and the optimal point is not 0, i.e. how to undo the proposed modifications in case of a wrong assumption.
In general, the proposed method is suitable for any of the state-of-the-art interior point methods applied to a convex programming problem.
There are a lot of papers dealing with related questions in linear programming. For nonlinear programming there is much less literature.
The problem of setting certain variables to zero arises, e.g., if in linear programming a basis of an optimal interior point solution is needed (crossover). In this context it is important to have a good strategy to choose candidates of variables to enter the basis. I.e., the problem is that a fixed number of variables has to be set to 0 and all the others not. It likely happens that variables with larger values are better candidates than others with lower values. Therefore, one can not simply take the variables with the lowest values. This problem and solutions are described e.g. in [Gill et al., 1986]. Sophisticated strategies and a comparison of different approaches can be found in [El-Bakry et al., 1994]. Closely related is the problem of a dual active set strategy, cf. [Tone, 1993].
A motivation to eliminate variables could be the reduction of the dimension of the problem. There, it is also important to have good strategies to choose candidates. This problem is also covered by [El-Bakry et al., 1994].
In [Wright, 1998] certain cancellation errors and ill-conditioning of the linear systems are separated. The influence of this kind of cancellation, mainly inaccurate values of active constraints, is investigated. [Freund and Jarre, 1996] deals with the problem of avoiding increasing ill-conditioning by a modified strategy to eliminate variables.
For the application described in this paper the only motivation to care about variable elimination is the stable computation of solutions. Thus, we do not have the problem of how to decide as early as possible which variable could be a candidate for elimination, or how to obtain as much candidates as possible. Moreover, since certain convexity properties appear, we are faced with an unique solution which gives us more security in choosing variables for elimination.
Although it is not relevant for our applications, we mention how the proposed changes affect the linear algebra procedures and in particular what could be done to save computational work of earlier iterations.
The outline of the paper is as follows. In section 2 we will explain the method of moving asymptotes, the environment where the convex programming problem of our concern arises. In section 3 the primal-dual interior point method is introduced. In section 4 observations of numerical problems that have been made will be shown and it is explained how it is possible to overcome this situation. Section 5 answers the questions what has to be done if a convergence assumption that has been made in section 4 was wrong and how to deal with modified structures of the linear systems. In the last section numerical examples will be presented.

2 Method of moving asymptotes

The method of moving asymptotes (MMA; [Svanberg, 1987]) is a method to solve nonlinear programming problems. A globally convergent extension is the method sequential convex programming (SCP; [Zillober, 1999]). Although the methods are applicable for general nonlinear programming problems, the main field of application is the framework of mechanical

design problems (structural optimization). The reason for the excellent performance in this environment is the approximation scheme that works very well for constraints dependent on nodal displacements, as e.g. stresses or strains.
In this section we will introduce the method only briefly in order to develop the subproblems that have to be solved via the interior point method. For detailed descriptions as well as a convergence theory, see [Svanberg, 1987] and [Zillober, 1999].
We consider the following general nonlinear programming problem:

$$\begin{array}{lll} \min & f(x) & x \in \mathbb{R}^n \\ \text{s.t.} & h_j(x) \leq 0, & j = 1, ..., m \\ & l_i \leq x_i \leq u_i, & i = 1, ..., n. \end{array} \tag{P1}$$

The functions f and h_j $(j = 1, ..., m)$ have to be defined only on $X := \{x \mid l_i \leq x_i \leq u_i,\ i = 1, ..., n\}$ and are assumed to be at least twice continuously differentiable in the interior. The feasible region is assumed to be non-empty.
The objective function of the problem will be approximated by a uniformly convex function, constraints by convex functions and the box-constraints will be allowed to shrink. Then, (P1) will be replaced by a sequence of subproblems using these approximations.
The approximation scheme for a constraint is as follows:

$$\begin{aligned} \tilde{h}_j(x) := h_j(x^k) &+ \sum_{i,+} \left.\frac{\partial h_j}{\partial x_i}\right|_{x^k} \left(\frac{(U_i - x_i^k)^2}{U_i - x_i} - (U_i - x_i^k) \right) \\ &- \sum_{i,-} \left.\frac{\partial h_j}{\partial x_i}\right|_{x^k} \left(\frac{(x_i^k - L_i)^2}{x_i - L_i} - (x_i^k - L_i) \right) \end{aligned} \tag{1}$$

$\sum_{i,+}\left(\sum_{i,-}\right)$ means summation over all components i where the partial derivative $\frac{\partial h_j}{\partial x_i}$ $\left(\frac{\partial f}{\partial x_i},\right.$ resp.) at the expansion point x^k is non-negative (negative). For the objective function the same approximation is used, but one more term is added to ensure strict convexity.
$\tilde{f}$ and $\tilde{h}_j$ $(j = 1, ..., m)$ are defined on

$$\tilde{D} := \{x \mid L_i < x_i < U_i,\ i = 1, ..., n\}\,.$$

L_i and U_i are parameters to be chosen with $L_i < x_i^k < U_i,\ i = 1, ..., n$. Consequently, we would have to use superscripts for the asymptotes L_i and U_i, too. These are omitted since it should always be clear to which iteration they belong.
This means, constraints and the objective function are linearized with respect to transformed variables $\frac{1}{U_i - x_i}$ and $\frac{1}{x_i - L_i}$. In the objective function an additional term is added. These so defined functions have the following properties:

$\tilde{f}$ and $\tilde{h}_j$ are first–order approximations of f and h_j, resp. I.e.

$$\begin{aligned} &\tilde{f}(x^k) = f(x^k),\ \tilde{h}_j(x^k) = h_j(x^k) \\ &\nabla\tilde{f}(x^k) = \nabla f(x^k),\ \nabla\tilde{h}_j(x^k) = \nabla h_j(x^k) \end{aligned}$$

for all $j = 1, ..., m$. Additionally, $\tilde{h}_j$ $(j = 1, ..., m)$ are convex, i.e. can be strictly convex, $\tilde{f}$ is strictly convex and $\tilde{f}$ and $\tilde{h}_j$ $(j = 1, ..., m)$ are separable.

One particular subproblem of (P1) looks then as follows:

$$\begin{array}{lll} \min & \tilde{f}(x) & x \in \mathbb{R}^n \\ \text{s.t.} & \tilde{h}_j(x) \leq 0, & j = 1, ..., m \\ & l'_i \leq x_i \leq u'_i, & i = 1, ..., n. \end{array} \qquad (P^k_{sub})$$

where $l'_i := \max\{l_i, x^k_i - \omega(x^k_i - L_i)\}$ and $u'_i := \min\{u_i, x^k_i + \omega(U_i - x^k_i)\}$, $\omega \in]0;1[$ fixed. I.e., $l_i \leq l'_i \leq x^k \leq u'_i \leq u_i$ and x^k is the expansion point. For further usage we define

$$X' := \{x \mid l'_i \leq x_i \leq u'_i,\ i = 1, ..., n\}.$$

Notice, that (P^k_{sub}) is a convex program with strictly convex objective function and bounded feasible region, i.e. its solution is unique provided the feasible region is non-empty. For empty feasible regions there are techniques to enlarge it. We omit this rather technical stuff here. It is sufficient to know, that all the convexity properties keep valid.
The solution of (P^k_{sub}) will be the issue of the following sections. In the rest of this section we will formulate the outer algorithm MMA.

Algorithm 1 *MMA (Method of moving asymptotes)*

Step 0 : *Choose* $x^0 \in X$, $y^0_j \geq 0, j = 1, ..., m$; *compute* $f(x^0), \nabla f(x^0)$, $h_j(x^0)$, $\nabla h_j(x^0)$, $j = 1, ..., m$; *let* $k := 0$

Step 1 : *Compute* L^k_i *and* U^k_i $(i = 1, ..., n)$ *by some scheme; define* $\tilde{f}(x)$, $\tilde{h}_j(x)$, $j = 1, ..., m$ *(cf. (1))*

Step 2 : *Solve* $(P^k_{sub})(x^k)$ *with algorithm 2; let* (x^{k+1}, y^{k+1}) *be the solution, where* y^{k+1} *denotes the corresponding vector of Lagrange multipliers*

Step 3 : *If* $x^{k+1} = x^k$ *stop;* (x^k, y^k) *is the solution*

Step 4: *Compute* $f(x^{k+1})$, $\nabla f(x^{k+1})$, $h_j(x^{k+1})$, $\nabla h_j(x^{k+1})$, $j = 1, ..., m$, *let* $k := k + 1$, *goto step 1*

The SCP algorithm is based on the MMA algorithm. It uses additionally after each solution of the subproblem (step 2) a line search procedure with respect to the so called augmented Lagrangian function in order to stabilize the global convergence behavior. We will not outline this algorithm here because it is not necessary to know it in detail in this paper.

3 The primal-dual interior point method

In this section we introduce the primal-dual interior point method to solve the MMA subproblems (P^k_{sub}). In the corresponding implementation a related method is used, the predictor-corrector method. The problems of our concern are not restricted to one of these methods. Thus, we chose the approach that is a little bit more simple for the matter of presentation. For a survey of interior point approaches we recommend to the reader the books [Wright, 1997] and [Vanderbei, 1996].
We proceed from problem (P^k_{sub}) and modify it a little bit in order to prepare it for the application of the primal-dual method. For that purpose we add nonnegative slack variables wherever inequalities appear:

$$
\begin{array}{lll}
\min & \tilde{f}(x) & x \in \mathbb{R}^n \\
\text{s.t.} & \tilde{h}_j(x) + c_j = 0, & j = 1, ..., m \\
& -c_j + r_j = 0, & j = 1, ..., m \\
& l'_i - x_i + s_i = 0, & i = 1, ..., n \\
& x_i - u'_i + t_i = 0, & i = 1, ..., n \\
& r, s, t \geq 0
\end{array}
$$

The slacks s and t are introduced because variable x is formally free and allowed to violate its bounds. Slack r is not absolutely necessary, but with its usage we allow c and in consequence the dual of the original subproblem constraints to get 0. If we do not use slack r we need c and the dual variable y to be positive. The computational amount is touched only slightly by this modification.
The positivity is only demanded for the new introduced variables r, s and t, variables that do not appear outside the subproblems, i.e. in the main loop.
For the next step we add barrier terms corresponding to the slack variables and build the Lagrangian of the so modified problem:

$$
\begin{aligned}
L_\mu(x, y, c, r, s, t, d_r, d_s, d_t) = \\
\tilde{f}(x) - \mu \sum_{j=1}^{m} \ln r_i - \mu \sum_{i=1}^{n} \ln s_i - \mu \sum_{i=1}^{n} \ln t_i + y^T(\tilde{h}(x) + c) \\
+ d_r^T(-c + r) + d_s^T(l' - x + s) + d_t^T(x - u' + t)
\end{aligned}
$$

where $\tilde{h} := (\tilde{h}_1, ..., \tilde{h}_m)^T$ and y, d_r, d_s, d_t are the dual variable vectors to the corresponding constraints. μ is a positive homotopy parameter to be discussed later. Formally, in this formulation all variables are free, the barrier formulation, however, needs r, s and t to be positive.
The necessary optimality condition $\nabla L_\mu = 0$ reads then:

$$
\begin{array}{lll}
\nabla_x : & \nabla \tilde{f}(x) + Jy - d_s + d_t & = 0 \\
\nabla_y : & \tilde{h}(x) + c & = 0 \\
\nabla_c : & y - d_r & = 0 \\
\nabla_r : & D_r Re - \mu e & = 0 \\
\nabla_s : & D_s Se - \mu e & = 0 \\
\nabla_t : & D_t Te - \mu e & = 0 \\
\nabla_{d_r} : & -c + r & = 0 \\
\nabla_{d_s} : & l' - x + s & = 0 \\
\nabla_{d_t} : & x - u' + t & = 0
\end{array}
\qquad \text{(KKT)}
$$

where $R = \text{diag}\,(r_1, ..., r_m); S, T, D_r, D_s, D_t$ resp., $e = (1, 1, ..., 1)^T$ in the appropriate dimension and $J \in \mathbb{R}^{n,m}$ denotes the Jacobian of the constraints.

The linear system to compute a Newton step for the solution of this systems reads as follows:

$$\begin{pmatrix} \nabla_{xx}L_\mu & J & & & & & -I & I \\ J^T & & I & & & & & \\ & I & & & & -I & & \\ & & & D_r & & R & & \\ & & & & D_s & & S & \\ & & & & & D_t & & T \\ & & -I & I & & & & \\ -I & & & & I & & & \\ I & & & & & I & & \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta c \\ \Delta r \\ \Delta s \\ \Delta t \\ \Delta d_r \\ \Delta d_s \\ \Delta d_t \end{pmatrix} = -\nabla L_\mu \qquad \text{(S1)}$$

$\nabla_{xx}L_\mu = \nabla^2 \tilde{f}(x) + \frac{d}{dx}Jy$ is a positive diagonal matrix because of the convexity and separability properties. This is one important point to make the interior point approach attractive for our problem.
We will now briefly state the principles of the primal-dual interior point algorithm, which has been chosen for the solution of (P^k_{sub}).

Algorithm 2 *Primal-dual interior point algorithm*

Step 0: *Choose* $x^0 \in X', c^0 \geq 0,\ r^0, s^0, t^0, d^0_r, d^0_s, d^0_t > 0,\ y^0,\ \epsilon$ *(solution tolerance). Let* $k := 0$.

Step 1: *Let* $\mu^0 := \max\limits_{\substack{j=1,\dots,m \\ i=1,\dots,n}} \{r^0_j (d^0_r)_j, s^0_i (d^0_s)_i, t^0_i (d^0_t)_i\}$.

Step 2: *Solve* (S1) *for the primal-dual search direction.*

Step 3: *Compute independent primal and dual stepsizes: Let* $\kappa := 10 \cdot \epsilon_{mach}$, *where* ϵ_{mach} *is the machine precision.*

$$\tilde{\delta}_{primal} := \min_{\substack{j=1,\dots,m \\ i=1,\dots,n}} \left\{ \left(\tfrac{r^k_j - \kappa}{-\Delta r_j},\ \text{where } r^k_j + \Delta r_j < \kappa \right), \left(\tfrac{s^k_i - \kappa}{-\Delta s_i},\ \text{where } s^k_i + \Delta s_i < \kappa \right), \left(\tfrac{t^k_i - \kappa}{-\Delta t_i},\ \text{where } t^k_i + \Delta t_i < \kappa \right) \right\};$$

$$\tilde{\delta}_{dual} := \min_{\substack{j=1,\dots,m \\ i=1,\dots,n}} \left\{ \left(\tfrac{(d^k_r)_j - \kappa}{-(\Delta d_r)_j},\ \text{where } (d^k_r)_j + (\Delta d_r)_j < \kappa \right), \left(\tfrac{(d^k_s)_i - \kappa}{-(\Delta d_s)_i},\ \text{where } (d^k_s)_i + (\Delta d_s)_i < \kappa \right), \left(\tfrac{(d^k_t)_i - \kappa}{-(\Delta d_t)_i},\ \text{where } (d^k_t)_i + (\Delta d_t)_i < \kappa \right) \right\}.$$

$\delta_{primal} = \min\{1, 0.99995 \cdot \tilde{\delta}_{primal}\}$, $\delta_{dual} = \min\{1, 0.99995 \cdot \tilde{\delta}_{dual}\}$.

Step 4: *Update:*

$$x^{k+1} := x^k + \delta_{primal} \cdot \Delta x, \quad y^{k+1} := y^k + \delta_{dual} \cdot \Delta y, \quad c^{k+1} := c^k + \delta_{primal} \cdot \Delta c,$$
$$r^{k+1} := r^k + \delta_{primal} \cdot \Delta r, \quad s^{k+1} := s^k + \delta_{primal} \cdot \Delta s, \quad t^{k+1} := t^k + \delta_{primal} \cdot \Delta t,$$
$$d^{k+1}_r := d^k_r + \delta_{dual} \cdot \Delta d_r, \quad d^{k+1}_s := d^k_s + \delta_{dual} \cdot \Delta d_s, \quad d^{k+1}_t := d^k_t + \delta_{dual} \cdot \Delta d_t.$$
$\mu^{k+1} = \mu^{k+1}(\mu^k, x^{k+1}, \dots, d^{k+1}_t)$.
Let $k := k + 1$. *If* $||\nabla L_0|| < \epsilon$ *stop, else goto step 2.*

Remark 1
- *For the initialization in Step 0 one can take the values of the previous main (MMA-) iteration. One possibly has to take care of the positivity condition.*

- *The factor 0.99995 in step 3 prevents the components of the critical vectors from getting too close to 0.*

We will now consider step 2 of the algorithm above, i.e., we will analyze how the linear system (S1) can be solved for the primal-dual search direction in practice.
For this purpose, we eliminate some variables of the linear system in order to get one that can be handled easier. Notice, that all matrices in this section that are inverted are positive diagonal matrices and thus are invertable.
Choose equation 4 of (S1): $D_r\Delta r + R\Delta d_r = \mu e - D_r R e$;

$$\Longrightarrow \Delta d_r = \mu R^{-1} e - d_r - R^{-1} D_r \Delta r$$

Analogously we get for the equations 5 and 6 of (S1):

$$\begin{aligned} \Delta d_s &= \mu S^{-1} e - d_s - S^{-1} D_s \Delta s \\ \Delta d_t &= \mu T^{-1} e - d_t - T^{-1} D_t \Delta t \end{aligned} \tag{2}$$

Substitute in equation 1 of (S1):
$\nabla_{xx} L_\mu \Delta x + J\Delta y - \Delta d_s + \Delta d_t = -\nabla \tilde{f}(x) - Jy + d_s - d_t$;

$$\Longrightarrow \nabla_{xx} L_\mu \Delta x + J\Delta y + S^{-1} D_s \Delta s - T^{-1} D_t \Delta t = -\nabla \tilde{f}(x) - Jy + \mu S^{-1} e - \mu T^{-1} e \tag{3}$$

Substitute in equation 3 of (S1):

$$\Delta y + R^{-1} D_r \Delta r = -y + \mu R^{-1} e \tag{4}$$

Further eliminations in the equations 7, 8 and 9 of (S1) yield:

$$\begin{aligned} \Delta r &= \Delta c + c - r \\ \Delta s &= \Delta x + x - l' - s \\ \Delta t &= -\Delta x - x + u' - t \end{aligned}$$

Substitution in (3) and (4) yields:

$$\begin{aligned} &(\nabla_{xx} L_\mu + S^{-1} D_s + T^{-1} D_t)\Delta x + J\Delta y = \\ &\quad -\nabla \tilde{f}(x) - Jy + \mu S^{-1} e - \mu T^{-1} e + d_s - d_t - S^{-1} D_s (x - l') + T^{-1} D_t (u' - x) =: \gamma_1 \end{aligned} \tag{5}$$

$$\Delta y + R^{-1} D_r \Delta c = -y + \mu R^{-1} e + d_r - R^{-1} D_r c$$

The last equation is used to eliminate Δc:

$$\Delta c = D_r^{-1} R(-y + \mu R^{-1} e + d_r - R^{-1} D_r c - \Delta y) = D_r^{-1} R(-y + \mu R^{-1} e + d_r - \Delta y) - c$$

This has to be inserted in equation 2 of (S1):

$$J^T \Delta x - D_r^{-1} R \Delta y = -\tilde{h}(x) - D_r^{-1} R(-y + \mu R^{-1} e + d_r) =: \gamma_2$$

Putting all together we get the following linear system:

$$\begin{pmatrix} \nabla_{xx}L_\mu + S^{-1}D_s + T^{-1}D_t & J \\ J^T & -D_r^{-1}R \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} \tag{S2}$$

This system is indefinite and has full rank provided that J is of full rank (remember $\nabla_{xx}L_\mu > 0$). It's dimension is $(n+m)\times(n+m)$. The upper left and lower right part are diagonal. Thus, the matrix can be considered as sparse. Additional sparsity of J improves the situation. It is now possible to solve this system with a sparse indefinite linear system solver.
This is the first possibility to compute the search direction in practice. There are two more ways. For the second one we define

$$\Theta := \nabla_{xx}L_\mu + S^{-1}D_s + T^{-1}D_t.$$

We proceed from (S2) by eliminating Δx from equation 1:

$$\Delta x = \Theta^{-1}(\gamma_1 - J\Delta y)$$

Inserting in the second equation yields:

$$(J^T\Theta^{-1}J + D_r^{-1}R)\Delta y = J^T\Theta^{-1}\gamma_1 - \gamma_2 \tag{S3}$$

This system is positive definite. It's dimension is $m \times m$. If J is sparse then we can hope that the same is true for the matrix in (S3). Unfortunately, this cannot be ensured. In the context of linear programming this case was extensively examined. If J has at least one dense column then the matrix is dense, too. On the other hand, there are techniques to overcome this situation by splitting dense columns. It is beyond the scope of this paper to outline these techniques. The reader is referred to the book [Wright, 1997] and the references cited therein. Notice, that it is worthwhile to think about sparse positive definite systems if J is dense overall. Then, of course, the matrix in (S3) is also dense.
The third approach is similar to the second, but here we eliminate Δy instead of Δx. We proceed from (S2) by eliminating Δy from equation 2:

$$\Delta y = R^{-1}D_r(J^T\Delta x - \gamma_2)$$

Inserting in the first equation yields:

$$(\Theta + JR^{-1}D_rJ^T)\Delta x = \gamma_1 + JR^{-1}D_r\gamma_2 \tag{S4}$$

As in the last subsection, this system is positive definite. It's dimension is $n \times n$. The remarks about sparsity are also valid here replacing columns by rows. The impact of dense rows or columns is examined more detailed in the book [Vanderbei, 1996].
These three possibilities to compute a search direction give us more flexibility in the implementation. This is important for the solution of large scale problems. E.g. many sizing problems in mechanical design have a moderate number of variables and a large number of constraints. In this case, approach (S4) is the better choice. Vice versa, for topology optimization problems we have usually many variables and few constraints, such that (S3) is the better variant. For a detailed comparison see [Zillober, 2000].
In our implementation (S2) is solved by a sparse $LU-$decomposition. We use this approach if a sparse matrix J leads to dense matrices in (S3) or (S4). The other two linear systems are solved by a dense or a sparse Cholesky-decomposition, dependent on the sparsity structure. Let us notice, that the numerical problems described in the next section arise in all the three formulations.

4 Numerical problem

In this section we outline a method to overcome numerical difficulties arising by cancellation. As a side effect we additionally avoid the unbounded increase of the condition number of the linear systems. For the matter of simplicity we choose one particular component (s_1) and show how the algorithm of section 3 can be modified in order to get a more stable method. We will drop the iteration index in this section because all variables belong to the same iteration.

Let us assume w.l.o.g. that s_1 converges to 0. For small values of s_1 it likely happens that

$$\Delta s_1 \approx -s_1$$

without leading to a stepsize δ_{primal} sufficiently far away from 1 (this is not surprising if 0 is the optimal value). I.e.,

$$s_1 + \delta_{primal} \Delta s_1 \approx 0.$$

The result is cancellation in computing the new value of s_1 (we will illustrate the behaviour by a numerical example in section 6). The problem is not s_1 itself. Under the above assumption it is close to 0 and usually accurate enough, but in the next iteration we use the inverse of this perturbed value of s_1 to compute the diagonal part and right hand side of (S2), (S3) and (S4) (cf. (5)), and to compute the value of Δd_s in (2). Notice, that this problem has nothing to do with errors in computing the solution of the linear systems (S2), (S3) and (S4). We can assume exact values for Δs_1.
The question is now, if that is only a numerical problem. The answer is no, it has also a theoretical background. One of the Karush-Kuhn-Tucker conditions in the optimum says ((KKT) for $\mu = 0$)

$$s_1 (d_s)_1 = 0.$$

Thus, supposing $s_1 = 0$, this equation is not useful to compute the optimal value of $(d_s)_1$. So it is not obvious why the equation

$$s_1 (d_s)_1 = \mu$$

of (KKT) should be useful (in a numerical sense) to compute $(d_s)_1$ for small values of μ.
One possibility to solve the problem of numerical instability is, to use in the situation above shorter steps, e.g.

$$(\delta_{primal})_{\text{modified}} = 0.8 \cdot \delta_{primal}.$$

This could help to avoid cancellation errors, but it would also slow down convergence. And it does not solve the basic problem of computing accurate values of $(d_s)_1$.

We propose the following method: suppose, we observe after some iterations that s_1 converges to 0. We believe that the optimal value of s_1 is 0, if it is below some predefined constant ϵ_{zero} (the situation if this assumption is not true is discussed below). We will use this knowledge to avoid the computation of lower and lower values for s_1 which would result in a loss of accuracy for the other variables.

$s_1 = 0$ yields that the condition $s_1(d_s)_1 = 0$ gives no further information. Thus, in (KKT) the condition $s_1(d_s)_1 = \mu$ is dropped. Additionally, s_1 is dropped in the condition $\nabla_{d_s} L_\mu = 0$. $s_1 = 0$ implies also $t_1 = u'_1 - l'_1$ and $(d_t)_1 = 0$, but due to consistency reasons we will not use this fact immediately.
Let's denote by $\overline{s} := (s_2, ..., s_n), \overline{d_s}, \overline{x}, ...$ resp. The necessary optimality conditions have then changed to:

$$\begin{array}{rll}
\nabla_x : & \nabla \tilde{f}(x) + Jy - d_s + d_t & = 0 \\
\nabla_y : & \tilde{h}(x) + c & = 0 \\
\nabla_c : & y - d_r & = 0 \\
\nabla_r : & D_r Re - \mu e & = 0 \\
\nabla_{\overline{s}} : & \overline{D_s} \overline{S} e - \mu e & = 0 \\
\nabla_t : & D_t T e - \mu e & = 0 \\
\nabla_{d_r} : & -c + r & = 0 \\
\nabla_{(d_s)_1} : & l'_1 - x_1 & = 0 \\
\nabla_{\overline{d_s}} : & l'_i - x_i + s_i & = 0 \quad (i = 2, ..., n) \\
\nabla_{d_t} : & x - u' + t & = 0
\end{array}$$

The new linear system to compute a Newton step looks like:

$$\begin{pmatrix}
[\ \nabla_{xx} L_\mu\ ] & J & & & & & [\ -I\ ] & I \\
[\ J^T\ ] & & I & & & & & \\
& I & & & & -I & & \\
& & & D_r & & R & & \\
& & & & \overline{D_s} & & \overline{S} & \\
& & & & & D_t & & T \\
& & -I & I & & & & \\
-1 & & & & & & & \\
& -I & & & I & & & \\
[\ I\ ] & & & & & I & &
\end{pmatrix}
\begin{pmatrix}
\Delta x_1 \\ \Delta \overline{x} \\ \Delta y \\ \Delta c \\ \Delta r \\ \Delta \overline{s} \\ \Delta t \\ \Delta d_r \\ \Delta (d_s)_1 \\ \Delta \overline{d_s} \\ \Delta d_t
\end{pmatrix} =$$

$$- \begin{pmatrix}
\nabla \tilde{f}(x) + Jy - d_s + d_t \\
\tilde{h}(x) + c \\
y - d_r \\
D_r Re - \mu e \\
\overline{D_s} \overline{S} e - \mu e \\
D_t T e - \mu e \\
-c + r \\
l'_1 - x_1 \\
\overline{l}' - \overline{x} + \overline{s} \\
x - u' + t
\end{pmatrix} \qquad (6)$$

Brackets $[\cdot]$ denote that the contents refers to more than one column.

Elimination of $\Delta d_r, \Delta\overline{d_s}$ and Δd_t in the fourth, fifth and sixth equation of (6) yields:

$$\begin{aligned}\Delta d_r &= \mu R^{-1}e - d_r - R^{-1}D_r\Delta r\\ \Delta\overline{d_s} &= \mu\overline{S}^{-1}e - \overline{d_s} - \overline{S}^{-1}\overline{D_s}\Delta\overline{s}\\ \Delta d_t &= \mu T^{-1}e - d_t - T^{-1}D_t\Delta t\end{aligned}$$

Substitute in first and third equation of (6):

$$\nabla_{xx}L_\mu\Delta x + J\Delta y + \begin{pmatrix} -\Delta(d_s)_1 \\ \overline{S}^{-1}\overline{D_s}\Delta\overline{s}\end{pmatrix} - T^{-1}D_t\Delta t = -\nabla\tilde{f}(x) - Jy + \begin{pmatrix}(d_s)_1\\ \mu\overline{S}^{-1}e\end{pmatrix} - \mu T^{-1}e$$

$$\Delta y + R^{-1}D_r\Delta r = -y + \mu R^{-1}e$$

The new system is:

$$\begin{pmatrix} [\ \nabla_{xx}L_\mu\ ] & J & & & \begin{pmatrix}(0,...,0)\\ \overline{S}^{-1}\overline{D_s}\end{pmatrix} & -T^{-1}D_t & -e_1\\ [\quad J^T\quad] & & I & & & & \\ & I & & R^{-1}D_r & & & \\ & & -I & I & & & \\ -1 & & & & & & \\ & -I & & & I & & \\ [\quad I\quad] & & & & & I & \end{pmatrix}\begin{pmatrix}\Delta x_1\\ \Delta\overline{x}\\ \Delta y\\ \Delta c\\ \Delta r\\ \Delta\overline{s}\\ \Delta t\\ \Delta(d_s)_1\end{pmatrix} =$$

$$-\begin{pmatrix}\nabla\tilde{f}(x) + Jy - \begin{pmatrix}(d_s)_1\\ \mu\overline{S}^{-1}e\end{pmatrix} + \mu T^{-1}e\\ \tilde{h}(x) + c\\ y - \mu R^{-1}e\\ -c + r\\ l_1' - x_1\\ \overline{l}' - \overline{x} + \overline{s}\\ x - u' + t\end{pmatrix} \tag{7}$$

where e_1 is the first unit vector of $\mathbb{R}^n$.
Further elimination of $\Delta r, \Delta\overline{s}$ and Δt in the fourth, sixth and seventh equation of (7) yields:

$$\Delta r = \Delta c + c - r \tag{8}$$

$$\Delta\overline{s} = \Delta\overline{x} + \overline{x} - \overline{l}' - \overline{s} \tag{9}$$

$$\Delta t = -\Delta x - x + u' - t \tag{10}$$

Substitution in the first and third equation of (7) results in:

$$\left(\nabla_{xx}L_\mu + \begin{pmatrix}0, & .....,0\\ . & \\ . & \overline{S}^{-1}\overline{D_s}\\ 0, & \end{pmatrix} + T^{-1}D_t\right)\Delta x + J\Delta y - \Delta(d_s)_1 e_1 =$$

$$-\nabla\tilde{f}(x) - Jy + \begin{pmatrix}(d_s)_1\\ \mu\overline{S}^{-1}e - \overline{S}^{-1}\overline{D_s}(\overline{x} - \overline{l}') + \overline{d_s}\end{pmatrix} - \mu T^{-1}e - d_t + T^{-1}D_t(u' - x) =: \gamma_3$$

$$\Delta y + R^{-1}D_r\Delta c = -y + \mu R^{-1}e + d_r - R^{-1}D_r c$$

The last equation is used to eliminate Δc:

$$\Delta c = D_r^{-1}R(-y + \mu R^{-1}e + d_r - \Delta y) - c$$

This has to be inserted in equation 2 of (7):

$$J^T\Delta x - D_r^{-1}R\Delta y = -\tilde{h}(x) - D_r^{-1}R(-y + \mu R^{-1}e + d_r) =: \gamma_4$$

The new linear system has then changed to:

$$\begin{pmatrix} \begin{bmatrix} \nabla_{xx}L_\mu + \begin{pmatrix} 0, & .....,0 \\ \cdot & \\ \cdot & \overline{S}^{-1}\overline{D_s} \\ 0, & \end{pmatrix} + T^{-1}D_t \end{bmatrix} & J & -e_1 \\ \begin{bmatrix} & J^T & \end{bmatrix} & -D_r^{-1}R & \\ -1 & & \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta\overline{x} \\ \Delta y \\ \Delta(d_s)_1 \end{pmatrix} = \begin{pmatrix} \gamma_3 \\ \gamma_4 \\ x_1 - l_1' \end{pmatrix} \tag{11}$$

Preparing the last step we eliminate now Δx_1 in the third equation of (11):

$$\Delta x_1 = l_1' - x_1 \tag{12}$$

Insert in (11):

$$\begin{pmatrix} \begin{pmatrix} (0, \ldots\ldots\ldots\ldots\ldots, 0) \\ \overline{\nabla_{xx}L_\mu} + \overline{S}^{-1}\overline{D_s} + \overline{T}^{-1}\overline{D_t} \end{pmatrix} & J & -e_1 \\ \overline{J^T} & -D_r^{-1}R & \end{pmatrix} \begin{pmatrix} \Delta\overline{x} \\ \Delta y \\ \Delta(d_s)_1 \end{pmatrix} = \begin{pmatrix} \gamma_3 - ((\nabla_{xx}L_\mu)_{11} + t_1^{-1}(d_t)_1)(l_1' - x_1)e_1 \\ \gamma_4 - a_1(l_1' - x_1) \end{pmatrix} \tag{13}$$

where a_1 denotes the first column of J^T and $\overline{J^T} \in \mathbb{R}^{m,n-1}$ results from J^T by deleting the first column.
We finish by eliminating $\Delta(d_s)_1$ in (13):

$$\Delta(d_s)_1 = a_1^T\Delta y - (\gamma_3 - ((\nabla_{xx}L_\mu)_{11} + t_1^{-1}(d_t)_1)(l_1' - x_1)e_1)_1 \tag{14}$$

The resulting linear system is:

$$\begin{pmatrix} \overline{\nabla_{xx}L_\mu} + \overline{S}^{-1}\overline{D_s} + \overline{T}^{-1}\overline{D_t} & \overline{J} \\ \overline{J^T} & -D_r^{-1}R \end{pmatrix} \begin{pmatrix} \Delta\overline{x} \\ \Delta y \end{pmatrix} = \begin{pmatrix} \overline{\gamma_3} \\ \gamma_4 - a_1(l_1' - x_1) \end{pmatrix} \tag{15}$$

This system has now exactly the form of (S2) except for the dimension. The first row and the first column of (S2) have been deleted.
To get the equivalent of (S3) we let $\overline{\Theta} := \overline{\nabla_{xx}L_\mu} + \overline{S}^{-1}\overline{D_s} + \overline{T}^{-1}\overline{D_t}$ and eliminate $\Delta\overline{x}$:

$$\Delta\overline{x} = \overline{\Theta}^{-1}(\overline{\gamma_3} - \overline{J}\Delta y)$$

Inserting in (15) yields:

$$(\overline{J^T}\overline{\Theta}^{-1}\overline{J} + D_r^{-1}R)\Delta y = -\gamma_4 + a_1(l_1' - x_1) + \overline{J}^T\overline{\Theta}^{-1}\overline{\gamma_3} \tag{16}$$

In this case, the dimensions of the corresponding linear systems (16) and (S3) are equal.
For the equivalent of (S4) we eliminate Δy in (15):

$$\Delta y = -R^{-1}D_r(\gamma_4 - a_1(l_1' - x_1) - \overline{J^T}\Delta\overline{x}) \tag{17}$$

Inserting this in (15) yields:

$$(\overline{\Theta} + \overline{J}R^{-1}D_r\overline{J}^T)\Delta\overline{x} = \overline{\gamma_3} + \overline{J}R^{-1}D_r(\gamma_4 - a_1(l_1' - x_1)) \tag{18}$$

The dimension of this system is one less than the dimension of (S4).
It is important to note, that the systems (15), (16) and (18) are not derived by the assumption that x_1 is known and therefore the computation of Δx_1 is not necessary. In this case we would not know how to compute the optimal value of $(d_s)_1$. It is derived by the development of the case that x_1 is fixed and its dual is not known. Notice, that if x_1 is sufficiently close to 0 it does not automatically follow that $(d_s)_1$ is sufficiently close to its optimal value. Usually this is not the case.
It is obvious what has to be done if we consider other components of r, s and t rather than s_1. The information of the corresponding x-component or c-component is deleted and a similar formula as (14) is derived for the dual.

5 Modified linear systems and undoing variable fixing

There are two important questions remaining: firstly, how to compute the solution of (15), (16) or (18) and secondly, what happens, if the fixing of s_1 to 0 is not true.

In the author's application [Zillober, 2000] the time to solve linear systems like (15), (16) or (18) is not crucial, since it is part of an explicit model of the underlying nonlinear programming problem. The time is wasted for the evaluation of functions and gradients of the original problem. Moreover, J is not constant and can change its nonzero structure from one iteration to the next. Thus, the corresponding matrices can be decomposed whenever its structure changes.
In other applications, as e.g. linear programming, without the modifications of this paper, the matrix of (15), (16) or (18) does not change its structure in the overall solution process since the Jacobian J is constant. The matrix is decomposed symbolically at the beginning and this decomposition does not change. Therefore, users might not be very happy with a modified structure. To use the old symbolic decomposition further on one can modify (15) as follows. Add a column and a row with a 1 in the new diagonal position:

$$\begin{pmatrix} 1 & & \\ & \overline{\nabla_{xx}L_\mu} + \overline{S}^{-1}\overline{D_s} + \overline{T}^{-1}\overline{D_t} & \overline{J} \\ & \overline{J^T} & -D_r^{-1}R \end{pmatrix} \begin{pmatrix} \Delta w \\ \Delta\overline{x} \\ \Delta y \end{pmatrix} = \begin{pmatrix} 0 \\ \overline{\gamma_3} \\ \gamma_4 - a_1(l_1' - x_1) \end{pmatrix} \tag{19}$$

w is a one-dimensional dummy variable and has no influence on the other variables. (19) has now the same dimension and the same basic properties of (S2). Only the off-diagonals of the

first row/column vanished. In this case many solvers can use the symbolic decomposition of (S2) to compute the numerical solution of (19). Thus, it is not necessary to perform a new symbolic decomposition.
In the case of a positive definite system (16) or (18) it is also not necessary to perform a new symbolic decomposition. One can use the Sherman-Morrison-Woodbury update formula to compute the solution of the modified problem using the symbolic decomposition of the unmodified problem. It is beyond the scope of the paper to outline this approach in detail. We refer the reader to [Duff et al., 1986].

The question how to choose ϵ_{zero} is, of course, problem dependent. In our application [Zillober, 2000], l_i' and u_i' are scaled to 0 and 1, resp. $(i = 1, ..., n)$. Moreover, these are reasonable values closely connected to a realistic variation of the variables. For the choice of $\epsilon_{zero} = 10^{-8}$ the assumption that a variable below this threshold is 0 in the optimum was false, had not to be observed in any case, provided that the starting point of the barrier variables was not too close to 0. On the other hand, the numerical problems described above did not arise above this threshold.
Nevertheless, we have to consider the case that the assumption is wrong, i.e. $s_1 \neq 0$ in the optimum. We assume in the following that in step 3 of algorithm 2 stepsize 1 is chosen after the fixing of s_1. Otherwise the following happens not in one step but in a few iterations.
Independent of the solution of (15), (12) forces x_1 to become l_1' and in the following iterations $\Delta x_1 = 0$. I.e. x_1 is fixed to l_1'. Subsequently, (10) forces t_1 to be fixed as $u_1' - l_1'$ and the condition $t_1(d_t)_1 = 0$ will force $(d_t)_1$ to converge to 0. Due to the construction of (8), (9) and (10) convergence difficulties do not occur for $\Delta r, \Delta s, \Delta t, \Delta d_r, \Delta d_s$ and Δd_t in (KKT).
The second equation of (15) is a modification of the second equation of (KKT), i.e. $\nabla_y L = 0$, the primal feasibility condition. This equation will force x to fulfill this condition, if possible. Thus, if we assume that $s_1 = 0$ is not optimal, the error has to occur in the computation of Δy and subsequently in y and $(d_s)_1$.
We tested this case choosing values of $\epsilon_{zero} = 10^{-2}$ and 10^{-3}. For some examples we observed that a barrier variable was fixed, but that the true value was positive in the optimal point. In this case we observed either that the condition $\nabla_y L = 0$ in (KKT) could not be fulfilled and additionally $\nabla_x L$ did not converge to 0, or primal feasibility could be obtained and only $\nabla_y L = 0$ could not be fulfilled.
Thus, if one of these two cases has to be observed it is a signal to control the fixing of the barrier variables and decrease ϵ_{zero} (in the worst case to 0 which means that the traditional method is chosen).
Once more we would like to emphasize that for a proper choice of ϵ_{zero} as above this is a rather exceptional case!

6 Examples

We present two examples in this section. The first one is an example of industrial strength solved by the method of section 2. The second example is a simple example of a collection of academic test problems. It has been chosen in order to show that the numerical problems described in this paper are not dependent on the difficulty of a certain instance.
It should be mentioned that the problems described here had not to be observed in all test cases. Sometimes we get a sufficient accuracy of the solution with the traditional method.

On the other hand, the two examples here are only a selection of a large number of examples where the old method failed. They are far away from unimportant exceptions.

6.1 Example 1

The example chosen here is the optimization of a bulk head of an aircraft. After discretization with finite elements an optimization problem with 56 variables and 113 constraints has been defined to minimize the weight of the structure. The problem could be solved up to optimality within 20 main iterations. Details of the example can be found in [Zillober and Vogel, 2000]. Without the modifications of this paper the first subproblem could not be solved. Several components had to be set to zero. For two of them the corresponding duals diverged afterwards. The problem had to be solved up to an accuracy of $||\nabla L_\mu|| < 10^{-8}$. The first 14 iterations are identical. Afterwards we observe for one particular variable (the values are rounded):

	Without modifications			With modifications		
	Value of variable			Value of variable		
Iteration	primal	dual	$\|\|\nabla L_\mu\|\|$	primal	dual	$\|\|\nabla L_\mu\|\|$
14	$2.08 \cdot 10^{-8}$	1890.94812	$4.56 \cdot 10^{-3}$	$2.08 \cdot 10^{-8}$	1890.94812	$4.56 \cdot 10^{-3}$
15	$2.11 \cdot 10^{-10}$	1890.94807	$4.62 \cdot 10^{-5}$	0	1890.94810	$1.84 \cdot 10^{-2}$
16	$2.13 \cdot 10^{-12}$	1890.94899	$8.98 \cdot 10^{-4}$	0	1890.94809	$2.18 \cdot 10^{-4}$
17	$1.89 \cdot 10^{-13}$	1890.91896	$2.91 \cdot 10^{-2}$	0	1890.94809	$2.22 \cdot 10^{-6}$
18	$2.23 \cdot 10^{-15}$	1892.24990	1.30	0	1890.94809	$2.29 \cdot 10^{-8}$
19	$2.23 \cdot 10^{-15}$	1994.51553	103.57	0	1890.94809	$1.63 \cdot 10^{-9}$
...						
50	$2.23 \cdot 10^{-15}$	2097.88413	206.94			

The primal variable does not change after iteration 18 because the correction is below machine precision ($\approx 2.22 \cdot 10^{-16}$). The dual and hence the residual of the Lagrangian diverge after the cancellation error occurs. For the modified algorithm smooth convergence can be observed.

6.2 Example 2

We consider problem # 36 of [Hock and Schittkowski, 1981], a harmless problem with $n = 3$ and $m = 1$, solved by the method described in section 2 within eight main iterations up to an accuracy of $9.7 \cdot 10^{-9}$ for the norm of the gradient of the Lagrangian of (P1). In the solution of the subproblem of the first main iteration the problems described below had to be observed. We neglect here all the details that are unnecessary for our purposes.
In the optimal solution two components of r, s and t were zero, r_1 and t_2. The problem had to be solved up to an accuracy of $||\nabla L_\mu|| < 10^{-8}$. The optimal value of $(d_t)_2$ was ≈ 88.5224601. For $(d_r)_1$ no problems had to be observed. Using the modifications described in this paper the algorithm terminated after iteration 12. In iteration 10, r_1 and t_2 were fixed to 0.
For the unmodified algorithm the iterations up to the first fixing, i.e. the first 9 iterations, are identical. Afterwards we observed the following changes of the variables:

	Without modifications			With modifications		
Iteration	t_2	$(d_t)_2$	$\|\|\nabla L_\mu\|\|$	t_2	$(d_t)_2$	$\|\|\nabla L_\mu\|\|$
9	$5.38 \cdot 10^{-8}$	88.5225666	$6.70 \cdot 10^{-7}$	$5.38 \cdot 10^{-8}$	88.5225666	$6.70 \cdot 10^{-7}$
10	$5.43 \cdot 10^{-10}$	88.5224612	$3.34 \cdot 10^{-8}$	0	88.5224612	$9.42 \cdot 10^{-7}$
11	$5.49 \cdot 10^{-12}$	88.5224526	$7.46 \cdot 10^{-6}$	0	88.5224602	$1.18 \cdot 10^{-7}$
12	$5.55 \cdot 10^{-14}$	88.5223979	$6.21 \cdot 10^{-5}$	0	88.5224601	$7.51 \cdot 10^{-10}$
13	$2.22 \cdot 10^{-15}$	88.4485981	$7.39 \cdot 10^{-2}$			
...						
50	$2.22 \cdot 10^{-15}$	84.494	4.028			

t_2 does not decrease after iteration 13 because the update Δt_2 is below the relative machine precision.
We observe that the algorithm without modifications converges first of all, but near the solution the computations get more and more defective. In each of these steps we have to observe a cancellation error as described in the earlier sections. The consequence is that the dual of t_2, $(d_t)_2$, can not be computed accurately leading to an increase of the norm of ∇L_μ. The algorithm terminates finally with the maximum number of iterations.

Conclusion

Convergence problems in the region of the optimal solution of a convex programming problem, solved by an interior point method, lead to the observation of cancellation errors. By an elimination procedure of critical variables we fixed this problem without losing knowledge on their duals. Additionally, increasing condition numbers of the linear systems that have to be solved could be avoided. This lead to a stabilization of the algorithm and to better convergence properties. The question of undoing such a fixing of variables is also discussed although it seems to be of less practical importance.

References

Duff, I.S., Erisman, A.M., and Reid, J.K. (1986). Direct Methods for Sparse Matrices. New York: Oxford University Press.

El-Bakry, A.S., Tapia, R.A., and Zhang, Y. (1994). A study of indicators for identifying zero variables in interior point methods. SIAM Review, **36**(1), 45–72.

Freund, R.W., and Jarre, F. (1996). A QMR-based interior-point algorithm for solving linear programs. Mathematical Programming, Series B, **76**, 183–210.

Gill, P.E., Murray, W., Saunders, M.A., Tomlin, J.A., and Wright, M.H. (1986). On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method. Mathematical Programming, **36**, 183–209.

Hock, W., and Schittkowski, K. (1981). Test Examples for Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical Systems, No. 187. Springer–Verlag, Berlin, Heidelberg, New York.

Svanberg, K. (1987). The Method of Moving Asymptotes – a new method for structural optimization. Int. J. Num. Meth. Eng., **24**, 359–373.

Tone, K. (1993). An active-set strategy in an interior point method for linear programming. Mathematical Programming, **59**, 345–360.

Vanderbei, R.J. (1996). Linear Programming: Foundations and Extensions. Kluwer Academic Publishers.

Wright, M.H. (1998). Ill-conditioning and computational error in interior methods for nonlinear programming. SIAM J. Optimization, **9**(1), 84–111.

Wright, S.J. (1997). Primal-Dual Interior-Point Methods. SIAM publications.

Zillober, Ch. (1999). Global convergence of a nonlinear programming method using convex approximations. Tech. rept. TR99-1. Informatik, Universität Bayreuth, WWW: www.uni-bayreuth.de/departments/math/∼czillober/papers/tr99-1.ps.

Zillober, Ch. (2000). A combined convex approximation - interior point approach for large scale nonlinear programming. Tech. rept. TR00-1. Informatik, Universität Bayreuth, WWW: www.uni-bayreuth.de/departments/math/∼czillober/papers/tr00-1.ps.

Zillober, Ch., and Vogel, F. (2000). Adaptive strategies for large scale optimization problems in mechanical engineering. *In:* Proceedings of the 2000 WSES Conference on Optimization and Applications (to appear).