A Practical Algorithm for General Large Scale Nonlinear Optimization Problems *

Paul T. Boggs † Jon W. Tolle ‡ Anthony J. Kearsley §

April 27, 1994

Abstract

We provide an effective and efficient implementation of a sequential quadratic programming (SQP) algorithm for the general large scale nonlinear programming problem. In this algorithm the quadratic programming subproblems are solved by an interior point method that can be prematurely halted by a trust region constraint. The resulting solution is shown to be a descent direction for a merit function that we have developed. A dynamic procedure for adjusting the merit function parameter is described, as well as procedures for adjusting the trust region radius. Numerical results are presented.

Key words: nonlinear programming, interior point, SQP, merit function, trust region, large scale

---

†Applied and Computational Mathematics Division, National Institute of Standards and Technology, Gaithersburg, MD 20899

‡Mathematics Department, University of North Carolina, Chapel Hill, NC 27599

§Department of Computational & Applied Mathematics, Rice University, Houston, Texas 77251-1892

1. Introduction

In a recent paper, [BTK], the authors suggested a new algorithmic approach for solving large, nonlinear, constrained optimization problems. This proposed procedure is, in essence, a sequential quadratic programming method that uses an interior point algorithm for solving the quadratic subproblems and achieves global convergence through the application of a type of trust region technique. The trust region strategy is based on a special merit function for measuring progress towards a solution that was developed in earlier research, [BT89] and [BTK91], on merit functions for equality- and inequality-constrained optimization problems. Over the past two years this approach has been extensively tested on a variety of large problems, including standard test problems and problems of engineering and scientific origin, ranging in size from several hundred to several thousand variables with up to several thousand constraints. Specific strategies have been developed for handling the parameters that occur in the algorithm and for dealing with the non-trivial pathologies (e. g., linearly dependent active constraint gradients or inconsistent linearized constraints in the quadratic subproblem) that often occur in large scale problems. Here we present the results of these efforts. Based on our numerical experience we are confident that this algorithm provides an efficient means for attacking a large, sparse, nonlinear program with equality and/or inequality constraints. Although no rigorous comparisons have been made with the limited number of other algorithms designed for this class of problems, our algorithm has been successful on problems that have caused difficulties for other algorithms and, consequently, we are encouraged to believe that it is at least competitive at the current stage in the development of methods for solving these large problems.

Below we give an outline of our procedure and in the succeeding sections we provide more specific detail on the component parts, the strategies and safeguards, and the numerical tests. We provide very little in the way of theoretical justification for our approach in this paper; where statements that can be justified theoretically are made, references to articles in which the necessary theorems and proofs may be found are included. The exception occurs in Section 3 where important properties of an inexact solution of the quadratic subproblem are proven.

We assume the general nonlinear programming problem to be of the form

$$\begin{array}{l} \min\limits_{x} \ f(x) \\ \text{subject to:} \ \ g(x) \leq 0 \end{array} \qquad (NLP)$$

where $f : \mathcal{R}^n \to \mathcal{R}^1$, and $g : \mathcal{R}^n \to \mathcal{R}^m$. We have not included nonlinear equality constraints in our description here in order to avoid distracting technicalities.

Their presence creates no theoretical difficulties; they are included in our code and in some of the problems we tested. The modifications necessary for their insertion can be inferred from [BT89].

The sequential quadratic programming method (SQP) is the backbone of our algorithm. At each step, we have an iterate, x^k, denoting the current approximation to the solution of (NLP). In addition to the x-iterate we also maintain an iterate, $z^k \in \mathcal{R}^m$, which measures the infeasibility of x^k, that is constrained to be nonnegative. At this stage (NLP) is modeled by a quadratic program of the form

$$\begin{array}{ll} \min_{\delta} \nabla f(x^k)^{\mathrm{t}}\delta + \frac{1}{2}\delta^{\mathrm{t}} B^k \delta \\ \text{subject to: } \nabla g(x^k)^{\mathrm{t}}\delta + g(x^k) \leq 0. \end{array} \qquad (QP)$$

Here B^k is taken to be an appropriate approximation to the Hessian of the Lagrangian for (NLP), i.e.,

$$B^k \approx \nabla^2_{xx}\ell(x^k, \lambda^k),$$

where

$$\ell(x, \lambda) = f(x) + g(x)^{\mathrm{t}}\lambda.$$

(See Section 4 for a discussion of the choice of B^k used in our numerical experiments.) In this form (QP) generates a step that provides a search direction for improving the current iterate. There are two significant points to be made concerning this phase of our algorithm. First, we apply an interior point quadratic program solver to (QP) to obtain this step; more specifically, we use the method found in [BDRW91] where solutions are calculated by solving a sequence of low dimensional quadratic programs. Pertinent details of this solver and its properties relative to its use in our SQP method can be found in Section 2. Second, we do not try to solve (QP) with complete accuracy at each iteration; rather, we often terminate the interior point method prematurely. In particular, we halt the quadratic program solver when the steplength exceeds a "trust region radius" that is modified at each iteration according to how well the improvement in our merit function is predicted. Thus our algorithm can be said to be a "truncated Newton method" in the sense of [EW94] (see also [DES82]). This particular merit function and a more useful "working version" are discussed in Section 3 and our strategy for updating the trust region radius is given in Section 4.

The output of the (QP) solver is a vector that determines the direction of the step in the x-variable, which in turn yields a step direction for the "slack" variable z as explained in Section 3. The combined step direction of these two variables is a descent direction for the working version of the merit function and also for constraint infeasibility; thus we can choose the steplength in this direction to decrease the merit function and/or the infeasibility of the iterate. The choice of steplength determines the new iterate x^{k+1} and also the new value z^{k+1}. The strategy for choosing the steplength and other algorithmic details, including the modifications and safeguards necessary to make an implementation practical, are given in Section 4.

The results of our numerical tests are contained in Section 5. These demonstrate the overall effectiveness of the procedure and highlight the beneficial effect of our trust region strategy and other procedures. Finally, in Section 6 we briefly consider weaknesses in the current version of the algorithm and suggest possible avenues of research that will improve its efficiency.

For a discussion of the theoretical and practical questions related to large scale nonlinear programming see the recent surveys [Col] and [CGT92b].

2. An Interior Point QP Solver

Interior point methods for linear programming have been demonstrated to be very successful, especially on large problems, and recent research has lead to their extension to quadratic programs. A particular method, the method of optimizing over low-dimensional subspaces, has performed well on linear programs and has been extended to the quadratic programming case (see [BDRW94], [BDRW91] and [DBRW91] and the references contained therein). This method, for which good numerical results for quadratic programs have been reported, has properties that make it particularly compatible with the SQP algorithm we are describing in this paper. A brief description of the essential features of this method and their importance for our purposes follows. The many details of the actual algorithm that are not reported here may be found in the above references.

The quadratic program that we solve, (QP), has the form

$$\begin{aligned} &\min_{\delta}\ c^{\mathrm{t}}\delta + \tfrac{1}{2}\delta^{\mathrm{t}}Q\delta \\ &\text{subject to: } \ A^{\mathrm{t}}\delta + b \le 0 \end{aligned} \tag{2.1}$$

where $c, \delta \in \mathcal{R}^n$, $Q \in \mathcal{R}^{n\times n}$, $A \in \mathcal{R}^{n\times m}$, and $b \in \mathcal{R}^m$. The assumptions on (2.1) that are necessary to apply the interior point algorithm are that the problem be bounded, that A have full column rank, and that there exist feasible points (i.e., that the constraints be consistent). Note that Q can be indefinite and that no assumption of a full-dimensional interior is required.

An important consideration in our use of the subproblem (QP) is how to obtain an initial feasible point, since we do not want to require that a feasible point be given. The algorithm uses a "Big M" method to construct the Phase I problem

$$\begin{aligned} &\min_{\delta,\theta}\ c^{\mathrm{t}}\delta + \tfrac{1}{2}\delta^{\mathrm{t}}Q\delta + M\theta \\ &\text{subject to:}\quad A^{\mathrm{t}}\delta + b - e\theta \le 0 \end{aligned} \tag{2.2}$$

where e is a vector of all ones and θ is the "artificial" variable. Clearly for θ^* large enough the point $(\delta, \theta) = (0, \theta^*)$ is feasible for (2.2) and if M is sufficiently large the algorithm applied to (2.2) will reduce θ until the artificial variable is negative, at which point the current value of δ is feasible and the $M\theta$ and $e\theta$ terms are dropped. If no such value of the artificial variable can be found, then (2.2) is not consistent and the algorithm stops. In this case, one can show that the optimal point satisfies

$$\theta = \min_{\delta}\ \max_{j}\{A_j^{\mathrm{t}}\delta - b_j\}.$$

In the implementation the variable θ is treated essentially the same as the components of δ. Consequently, in the brief description of the algorithm that follows the dependence on θ is incorporated into the formulation given in (2.1).

The defining characteristic of the algorithm is that it proceeds by solving a sequence of low-dimensional subspace approximations to (2.1). In our application we follow the reported results in which the dimension of the subspace is taken as three. The following is an outline of the O3D (for Optimizing over 3-Dimensional subspaces) version of the algorithm.

O3D Algorithm for Quadratic Programming

1. Given a feasible point, $\bar{\delta}^0$; set $j := 0$.

2. Generate 3 independent search directions $p_i,\ i = 1, \dots, 3$ and let P^j be the matrix whose columns are p_i.

3. Form and solve the restricted quadratic program

$$\begin{aligned} &\min_{\zeta}\ c^{\mathrm{t}}\bar{\delta} + \tfrac{1}{2}\bar{\delta}^{\mathrm{t}}Q\bar{\delta} \\ &\text{subject to:}\quad A^{\mathrm{t}}\bar{\delta} \le b \end{aligned}$$

   where $\bar{\delta} = \bar{\delta}^j + P^j\zeta$ and $\zeta \in \mathcal{R}^3$. Call the solution ζ^*.

4. Set $\bar{\delta}^{j+1} := \bar{\delta}^j + \rho P^j\zeta^*$ for an appropriate value of the steplength $\rho \in (0, 1)$.

5. If stopping criteria are met, exit.

6. Go to 2. (At this step, if the component of the vector $\bar{\delta}$ corresponding to the artificial variable θ has become negative, it is eliminated from the problem.)

The search directions in Step 2 are solutions to

$$\left[AD^2A^{\mathrm{t}} + Q/\beta\right] p_i = t_i, \qquad i = 1, \ldots, 3, \tag{2.3}$$

where β is a scalar depending on the current iterate,

$$D = \operatorname{diag}\{1/r_k,\ k = 1, \ldots, m\}$$

with $r_k = -(A_k\tilde{\delta}^j + b)_k$, and the t_i are particular values chosen such that one of these directions is always a descent direction with respect to the objective function. The steplength ρ is set to the minimum of 99% of the distance to the boundary or the distance to the minimum of the objective function in the final direction.

The form of the matrix in (2.3) allows for efficient exploitation of the sparsity. Note that if Q is positive semi-definite, then the matrix in (2.3) is positive definite for all interior points; otherwise, it may not be. In the latter case a modification in [GMW81] is used, i.e., a positive amount is added to the diagonal of the matrix in (2.3) to ensure that the Cholesky factorization has "safely" positive entries on the diagonal. In our application of this algorithm, using this procedure obviates the need for the matrix B^k to be positive definite, which in turn allows us to use an analytic (or a finite difference approximation) to the Hessian of the Lagrangian.

The standard criterion for convergence of the algorithm is that at least one of the following hold: (a) the relative change in two successive values of the objective function is small; (b) the relative difference between the primal and the dual objective function values is small; or (c) the difference between two successive iterates is small. For use in our SQP algorithm we have added: (d) the length of the solution vector exceeds a specified value. This additional condition has been implemented to allow for trust region strategies; in particular, this procedure will cause the algorithm to halt if (QP) is unbounded. In any case, the terminal vector will be a useful direction in the context of our purposes as will be discussed in the next section.

3. The Merit Function

A merit function for (NLP) is typically a scalar-valued function that has an unconstrained minimum at x^*, the solution to (NLP). As a reduction in this function implies that progress is being made towards the solution, it can be used to determine an appropriate steplength in a given search direction for the constrained problem.

In [BT84] and [BT89] a merit function for equality-constrained problems was derived that has important properties *vis-a-vis* the steps generated by the sequential quadratic programming algorithm. In [BTK91] the authors developed a modified version of this function for inequality-constrained problems by applying the ideas to the slack variable version of (NLP):

$$\begin{array}{l} \min\limits_{x,s} f(x) \\ \text{subject to:} \quad g(x) + \frac{1}{4}S^2 e = 0 \end{array} \tag{3.1}$$

where

$$S = \operatorname{diag}\{s_1, \ldots, s_m\}.$$

As the resulting merit function contained terms involving s_i^2, and not s_i, it was natural to rephrase this merit function in terms of $z_i = \frac{1}{4}s_i^2$. This led to the atypical situation in which the merit function is *constrained*, i.e., the constrained minimum of the merit function corresponds to the solution of (NLP). This merit function has the form

$$\psi_d(x,z) = f(x) + \bar{\lambda}(x,z)^{\mathrm{t}}\bar{c}(x,z) + \tfrac{1}{d}\bar{c}(x,z)^{\mathrm{t}}\bar{A}(x,z)^{-1}\bar{c}(x,z) \tag{3.2}$$

where z is nonnegative, d is a scalar,

$$\begin{array}{rcl} \bar{c}(x,z) & = & g(x) + Ze, \\ \bar{A}(x,z) & = & \nabla g(x)^{\mathrm{t}}\nabla g(x) + Z, \\ \bar{\lambda}(x,z) & = & -\bar{A}(x,z)^{-1}\nabla g(x)^{\mathrm{t}}\nabla f(x), \end{array}$$

and

$$Z = \operatorname{diag}\{z_i, \ldots, z_m\}.$$

Although this merit function is constrained, our algorithm ensures that the z_i always remain positive; thus the nonnegativity constraint presents no theoretical difficulty. Specifically, given a current iterate (x^k, z^k) and an approximate solution, δ^k, of (QP) we take the step direction in z to be

$$q^k = -[\nabla g(x^k)^{\rm t}\delta^k + g(x^k) + z^k] \tag{3.3}$$

and then set

$$\begin{aligned} x^{k+1} &= x^k + \alpha\delta^k \\ z^{k+1} &= z^k + \alpha q^k \end{aligned}$$

for some value of α. Observe that if $z^k \geq 0$ and δ^k is feasible for (QP), then

$$q^k + z^k = -[\nabla g(x^k)^{\rm t}\delta^k + g(x^k)] \geq 0 \tag{3.4}$$

and it follows that for $\alpha \in (0,1]$, $z^{k+1} = z^k + \alpha q^k \geq 0$. It is important to emphasize here that the δ^k are determined by (QP), the quadratic approximation to (NLP), and are not dependent on the choice of z^k. The z^k are generated solely for use with the merit function. That is, we are *not* solving the slack variable problem.

It was shown in [BTK91] that ψ_d has certain desirable properties when d is sufficiently small. They include: a solution of (NLP) corresponds to a minimum of (3.2); the direction (δ^k, q^k) is a descent direction for ψ_d when (x^k, z^k) is sufficiently close to feasibility for (3.1); a steplength of one is acceptable near the solution if the method is converging q-superlinearly; and the direction is always a descent direction for the function

$$r(x,z) = \|\bar{c}(x,z)\|^2$$

at (x^k, z^k). Despite these useful properties, ψ_d has two deficiencies that preclude its direct use in an algorithm. First, (δ^k, q^k) is only a descent direction of ψ_d near feasibility, and, second, the evaluation of gradients and additional nontrivial computational algebra are required to assess a prospective point. In order to overcome these difficulties, the *approximate merit function*

$$\psi_d^k(x,z) = f(x) + \bar{c}(x,z)^{\rm t}\bar{\lambda}^k + \tfrac{1}{d}\bar{c}(x,z)^{\rm t}\bar{A}^k\bar{c}(x,z)$$

where

$$\begin{aligned} \bar{A}^k &= \nabla g(x^k)^{\rm t}\nabla g(x^k) + Z^k \\ \bar{\lambda}^k &= -(\bar{A}^k)^{-1}\nabla g(x^k)^{\rm t}\nabla f(x^k) \end{aligned}$$

was developed in that paper as a "working" version of ψ_d at (x^k, z^k). As the values of $\bar{\lambda}^k$ and $\bar{A}^k$ are fixed, ψ_d^k can be easily evaluated in a line search algorithm to choose an appropriate value of α in a given direction of descent. It was shown in the aforementioned paper that (δ^k, q^k) is a descent direction for ψ_d^k everywhere, that ψ_d^k will not interfere with rapid local convergence, and that an implementable, globally convergent algorithm based on measuring progress by this function can be constructed. Moreover, it is not difficult to show that a sufficient reduction in ψ_d^k implies a sufficient reduction in ψ_d when $r(x^k, z^k)$ is sufficiently small. In Section 4 the way in which the functions ψ_d and ψ_d^k are handled in our algorithm will be clarified.

We do not attempt a complete convergence analysis of our algorithm here; however, in order to motivate and partially justify our trust region approach we will show that if δ^k is only an approximate solution to (QP), i.e., it is obtained by truncating the O3D algorithm after *any* finite number of iterations, then the resulting step (δ^k, q^k) is a descent direction for the approximate merit function ψ_d^k and also for r at (x^k, z^k) and that this direction is also a descent direction for the true merit function in the set

$$C(\eta) = \{(x, z) \,:\, \|\bar{c}(x, z)\| \leq \eta\}$$

provided η is sufficiently small. We note that to achieve local q-superlinear convergence the complete solution of (QP) is needed; this analysis is contained in [BTK91].

From (QP) and (2.2) we see that the program that is to be approximately solved at iteration k is

$$\begin{array}{l} \min\limits_{\delta,\theta} \; \nabla f(x^k)^{\mathrm{t}}\delta + \frac{1}{2}\delta^{\mathrm{t}} B^k \delta + M\theta \\ \text{subject to:} \;\; \nabla g(x^k)^{\mathrm{t}}\delta + g(x^k) - e\theta \leq 0. \end{array} \tag{3.5}$$

We use this formulation since we are not assuming a feasible start for solving (QP). The starting point at each value of k will be $(\delta, \theta) = (0, \theta^*)$, where θ^* is chosen large enough to guarantee that the point will be strictly feasible for (3.5). In accordance with the description of the O3D algorithm in Section 2, iteration proceeds on (3.5) until θ becomes negative, at which time θ is removed from the problem and the iteration continues only in the variable δ. Henceforth we suppress the dependence on k since it is fixed for the subproblem. We denote the iterates generated in the O3D algorithm by $(\tilde{\delta}^j, \theta^j)$ and set

$$j^* = \sup\{ j \,:\, \theta^j > 0\}.$$

Note that j^* is also the largest value of j for which $\tilde{\delta}^j$ is infeasible for (QP). If B is positive definite then j^* is well-defined, i.e., (3.5) has a solution. Now at some point in the iteration scheme we terminate the process and choose $\delta^k = \tilde{\delta}^j$; this may occur before or after feasibility for (QP) has been reached.

In the O3D algorithm, for $j \leq j^*$, we set

$$\tilde{\delta} = \tilde{\delta}^j + \xi$$

and

$$\theta = \theta^j + \gamma$$

where, in the notation of Section 2, we have

$$\begin{bmatrix} \tilde{\delta}^j \\ \theta^j \end{bmatrix} = \bar{\delta}^j$$

and

$$\begin{bmatrix} \xi \\ \gamma \end{bmatrix} = P^j \zeta.$$

The system

$$\begin{array}{l} \min_{\xi,\gamma} \nabla f^{\mathrm{t}}(\tilde{\delta}^j + \xi) + \frac{1}{2}(\tilde{\delta}^j + \xi)^{\mathrm{t}} B(\tilde{\delta}^j + \xi) + M(\theta^j + \gamma) \\ \text{subject to:} \quad \nabla g^{\mathrm{t}}(\tilde{\delta}^j + \xi) + g - e(\theta^j + \gamma) \leq 0 \end{array} \tag{3.6}$$

is now solved to obtain the optimal values (ξ^j, γ^j). When (3.6) has been solved, the new iterate is obtained by

$$\begin{aligned} \tilde{\delta}^{j+1} &= \tilde{\delta}^j + \rho\xi^j \\ \theta^{j+1} &= \theta^j + \rho\gamma^j \end{aligned}$$

for some steplength $\rho \in (0,1)$. For $j > j^*$ the variable γ and the value θ^j are deleted from this problem and only the optimal value for the variable ξ is obtained and only $\tilde{\delta}^j$ updated.

The first order necessary conditions for the solution to (3.6) are, for $j \leq j^*$,

$$\nabla f + B(\tilde{\delta}^j + \xi^j) + \nabla g \, \nu^{j+1} = 0 \tag{3.7}$$

$$M - e^{\mathrm{t}}\nu^{j+1} = 0 \tag{3.8}$$

$$(\nu^{j+1})^{\mathrm{t}} \left[\nabla g^{\mathrm{t}}(\tilde{\delta}^j + \xi^j) + g - e(\theta^j + \gamma^j)\right] = 0 \tag{3.9}$$

$$\nu^{j+1} \geq 0 \tag{3.10}$$

where ν^{j+1} is the Lagrange multiplier. For $j > j^*$, equation (3.8) is deleted and equation (3.9) no longer contains the terms involving θ^j and γ^j.

The following lemma will be needed for our main result.

Lemma 3.1. *Let* $(\tilde{\delta}^0, \theta^0) = (0, \theta^*)$ *and assume that* B *is positive definite. Then for* $i \geq 1$ *the following inequality holds.*

$$(\tilde{\delta}^i)^{\mathrm{t}} \nabla f \leq -\tfrac{1}{2}\rho(\tilde{\delta}^i)^{\mathrm{t}} B\,\tilde{\delta}^i - M(\hat{\theta} - \theta^0) \tag{3.11}$$

where

$$\hat{\theta} = \begin{cases} \theta^i & \text{if } \; i \leq j^* + 1, \\ \theta^{j^*+1} & \text{otherwise.} \end{cases}$$

Proof: The proof is by induction. We first prove the result for $0 < i \leq j^* + 1$. For $i = 1$, using (3.7)–(3.9), and the fact that $\tilde{\delta}^0 = 0$, we have

$$\begin{aligned} (\tilde{\delta}^1)^{\mathrm{t}} \nabla f &= \rho(\xi^0)^{\mathrm{t}} \nabla f \\ &= -\rho(\xi^0)^{\mathrm{t}} B \xi^0 - \rho(\xi^0)^{\mathrm{t}} \nabla g\, \nu^1 \\ &= -\rho(\xi^0)^{\mathrm{t}} B \xi^0 + \rho(\nu^1)^{\mathrm{t}}(g - e\,\theta^0) - \rho M \gamma^0. \end{aligned}$$

Now, by the choice of $\theta^0 = \theta^*$ we have $g - e\theta^0 \leq 0$ and so (3.10) implies

$$(\tilde{\delta}^1)^{\mathrm{t}} \nabla f \leq -\rho(\xi^0)^{\mathrm{t}} B \xi^0 - \rho M \gamma^0.$$

(3.11) follows for $i = 1$ from the positive definiteness of B and the definition of θ^1. Now assume that (3.11) holds for $i = j \leq j^*$. Then

$$(\tilde{\delta}^{j+1})^{\mathrm{t}} \nabla f = (\tilde{\delta}^j + \rho \xi^j)^{\mathrm{t}} \nabla f$$

which by the same arguments as above yields

$$\begin{aligned} (\tilde{\delta}^{j+1})^{\mathrm{t}} \nabla f &= (\tilde{\delta}^j)^{\mathrm{t}} \nabla f - \rho(\xi^j)^{\mathrm{t}} B \xi^j - \rho(\xi^j)^{\mathrm{t}} B \tilde{\delta}^j \\ &+ \rho(\nu^{j+1})^{\mathrm{t}}(\nabla g^{\mathrm{t}} \tilde{\delta}^j + g - e\,\theta^j) - \rho M \gamma^j. \end{aligned}$$

Since we are assuming that $j \leq j^*$ it follows that $\nabla g^{\mathrm{t}} \tilde{\delta}^j + g - e\,\theta^j \leq 0$. Thus, using the induction hypothesis, (3.10), and the positive definiteness of B we obtain

$$\begin{aligned} (\tilde{\delta}^{j+1})^{\mathrm{t}} \nabla f &\leq -\tfrac{1}{2}\rho(\tilde{\delta}^j)^{\mathrm{t}} B\, \tilde{\delta}^j - M(\theta^j - \theta^0) \\ &- \tfrac{1}{2}\rho(\xi^j)^{\mathrm{t}} B \xi^j - \rho(\xi^j)^{\mathrm{t}} B \tilde{\delta}^j - \rho M \gamma^j \\ &\leq -\tfrac{1}{2}(\tilde{\delta}^{j+1})^{\mathrm{t}} B \tilde{\delta}^{j+1} - M(\theta^{j+1} - \theta^0). \end{aligned}$$

Now suppose $j \geq j^* + 1$. Then

$$\begin{aligned} (\tilde{\delta}^{j+1})^{\mathrm{t}} \nabla f &= (\tilde{\delta}^j)^{\mathrm{t}} \nabla f - \rho(\xi^j)^{\mathrm{t}} B \xi^j - \rho(\xi^j)^{\mathrm{t}} B \tilde{\delta}^j \\ &+ \rho(\nu^{j+1})^{\mathrm{t}}(\nabla g^{\mathrm{t}} \tilde{\delta}^j + g). \end{aligned}$$

Since $j \geq j^* + 1$, $\tilde{\delta}^j$ is feasible for (QP). Thus

$$\nabla g^{\mathrm{t}} \tilde{\delta}^j + g \leq 0$$

and since the multiplier is nonnegative it follows that

$$(\tilde{\delta}^{j+1})^{\mathrm{t}} \nabla f \leq -\tfrac{1}{2}\rho (\tilde{\delta}^j)^{\mathrm{t}} B\, \tilde{\delta}^j - M(\hat{\theta} - \theta^0).$$

□

We now state the important theorems concerning the truncated step.

Theorem 3.2. *Assume that x^k is not a solution to (NLP), that B^k is positive definite and $\bar{A}^k$ is nonsingular (and hence positive definite). Suppose that the O3D algorithm for solving (QP) terminates at iteration i with approximate solution $\tilde{\delta}^i$ and assume that*

$$\hat{\theta} - \theta^0 = O\left(\left\| g(x^k) + z^k \right\|\right) \tag{3.12}$$

where $\hat{\theta}$ is defined as in Lemma 3.1. Then there exists a positive $\bar{d}^k$ such that if $d \leq \bar{d}^k$

$$\nabla \psi_d^k(x^k, z^k)^{\mathrm{t}} \begin{pmatrix} \tilde{\delta}^i \\ q^i \end{pmatrix} < 0$$

where $q^i = -[\nabla g(x^k)^{\mathrm{t}} \tilde{\delta}^i + g(x^k) + z^k]$.

Proof: We break ψ_d^k into two pieces:

$$\psi_d^k = \psi_1^k + \frac{1}{d}\psi_2^k \tag{3.13}$$

where

$$\psi_1^k(x, z) = f(x) + (g(x) + z)^{\mathrm{t}} \bar{\lambda}^k$$

and

$$\psi_2^k(x, z) = (g(x) + z)^{\mathrm{t}} \left(\bar{A}^k\right)^{-1} (g(x) + z).$$

Thus

$$\nabla \psi_1^k(x, z) = \begin{pmatrix} \nabla f(x) \\ 0 \end{pmatrix} + \begin{pmatrix} \nabla g(x) \\ I \end{pmatrix} \bar{\lambda}^k \tag{3.14}$$

and

$$\nabla \psi_2^k(x, z) = \begin{pmatrix} \nabla g(x) \\ I \end{pmatrix} \left(\bar{A}^k\right)^{-1} (g(x) + z). \tag{3.15}$$

Let

$$\omega_1 = \nabla\psi_1^k(x^k, z^k)^{\mathrm{t}} \begin{pmatrix} \tilde{\delta}^i \\ q^i \end{pmatrix}$$

and

$$\omega_2 = \nabla\psi_2^k(x^k, z^k)^{\mathrm{t}} \begin{pmatrix} \tilde{\delta}^i \\ q^i \end{pmatrix}.$$

Again suppressing the index k, we have from (3.3), the lemma, and the hypotheses

$$\begin{aligned} \omega_1 &= (\tilde{\delta}^i)^{\mathrm{t}}\nabla f + (\nabla g^{\mathrm{t}}\tilde{\delta}^i + q^i)^{\mathrm{t}}\bar{\lambda} \\ &= (\tilde{\delta}^i)^{\mathrm{t}}\nabla f - (g+z)^{\mathrm{t}}\bar{\lambda} \\ &\leq -\tfrac{1}{2}\rho^2(\tilde{\delta}^i)^{\mathrm{t}}B\,\tilde{\delta}^i - M(\hat{\theta} - \theta^0) - (g+z)^{\mathrm{t}}\bar{\lambda} \\ &\leq -\tfrac{1}{2}\mu\left\|\tilde{\delta}^i\right\|^2 + K_1\left\|g+z\right\| \end{aligned}$$

for positive constants μ and K_1; and from (3.15) and the hypothesis

$$\begin{aligned} \omega_2 &= -(g+z)^{\mathrm{t}}\bar{A}^{-1}(g+z) \\ &\leq -\gamma\left\|g+z\right\|^2 \end{aligned}$$

for some positive constant γ. Combining the above inequalities we have

$$\nabla\psi_d^k(x^k, z^k)^{\mathrm{t}} \begin{pmatrix} \tilde{\delta}^i \\ q^i \end{pmatrix} \leq -\mu\left\|\tilde{\delta}^i\right\|^2 + K_1\left\|g(x^k)+z^k\right\| - \frac{1}{d}\gamma\left\|g(x^k)+z^k\right\|^2 \tag{3.16}$$

and the result follows.

$\square$

Theorem 3.3. *Let the assumptions of the preceding theorem hold. Then there exists a constant $\bar{\eta}^k > 0$ such that if $(x^k, z^k) \in C(\bar{\eta}^k)$ then there is a positive $\bar{d}^k$ such that for $d \leq \bar{d}^k$*

$$\nabla\psi_d(x^k, z^k)^{\mathrm{t}} \begin{pmatrix} \tilde{\delta}^i \\ q^i \end{pmatrix} < 0.$$

Proof: Since for $\left\|\bar{c}(x^k, z^k)\right\|$ sufficiently small we have

$$\nabla\psi_d(x^k, z^k) = \nabla\psi_d^k(x^k, z^k) + O\left(\left\|g(x^k)+z^k\right\|\right), \tag{3.17}$$

the same argument used in the previous theorem leads to an inequality similar to (3.16) for ψ_d.

□

Theorem 3.4. *Assume that $\bar{c}(x^k, z^k) \neq 0$. Then the direction $(\tilde{\delta}^i, q^i)$ is a descent direction for the function $r = \|\bar{c}(x, z)\|^2$.*

Proof: This result follows from the fact that

$$\nabla r(x^k, z^k)^{\mathrm{t}} \begin{pmatrix} \tilde{\delta}^i \\ q^i \end{pmatrix} = -(g(x^k) + z^k) \begin{pmatrix} \nabla g(x^k) \\ I \end{pmatrix}^{\mathrm{t}} \begin{pmatrix} \tilde{\delta}^i \\ q^i \end{pmatrix} = -\left\| g(x^k) + z^k \right\|^2.$$

□

The value of $\bar{d}^k$ in the first two theorems will depend on k; whereas, ideally, we should like the result to be independent of k. In theory, this can be achieved for (x^k, z^k) in a compact set not containing a solution by assuming that the B^k and the $\bar{A}^k$ are uniformly positive definite, that ρ is bounded away from zero, and that (3.12) holds for each k. Because of the addition of nonnegative Z to the positive semi-definite $\nabla g(x^k)^{\mathrm{t}} \nabla g(x^k)$, $\bar{A}^k$ is almost always positive definite. ρ will be bounded away from zero in cases where the solution of (QP) is on the boundary; for interior points the (QP) will be almost exactly solved by the O3D algorithm in a few steps; in which case the step is a descent direction for ψ_d^k. Finally, careful choices of θ^* and M can make the last condition likely to be satisfied in practice. Notwithstanding these comments, the purpose of the theorems is to justify our approach, not to claim that the procedure is foolproof. In solving any large problem one is apt to encounter situations not covered by the theory; the adjustments necessary to make our algorithm effective are discussed in the next section.

4. The Truncated SQP Algorithm

In this section we give a somewhat detailed description of our algorithm. Initially we assume that some fairly stringent restrictions on the problem are satisfied. In particular we assume the Hessian approximations, B^k, are positive definite, the matrices $\bar{A}^k$, are nonsingular, the active constraint gradients are linearly independent, and the linearized constraints in (QP) are consistent. In real-world applications these assumptions are not always valid so we have tried to make our algorithm flexible enough to perform well in situations where one or more of these assumptions fail to hold. We describe some of these adaptations at the end of this section.

The implementation of the algorithm depends upon four important parameters that need to be either computed or modified throughout the course of the algorithm. η, the *globalization parameter*, was introduced in Theorem 3.3. It is a measure of the size of the domain about the feasible region (i.e., the set of (x, z) such that $\bar{c}(x, z) = 0$), in which the direction (δ^k, q^k) is a descent direction for the true merit function ψ_d. A current estimate of η is maintained in the algorithm. τ, the *trust region parameter*, is the maximum value of the norm of the step allowed in approximately solving (QP). It is updated at every iteration. α, the *steplength parameter* determines the length of the step in the variables (x, z) in the direction (δ^k, q^k). It is chosen to guarantee progress towards the solution, either in decreasing the merit function or infeasibility. Finally, d, the *merit function parameter*, is critical to the success of the algorithm. It must be small enough to guarantee that the theoretical properties described in the preceding section are valid. Although the theory allows arbitrarily small values of d, the algorithm becomes very slow if d is too small, thus it is monitored throughout the algorithm and either increased or decreased as necessary.

The outline of the algorithm is followed by specific comments on the procedures and their justifications.

Basic Truncated SQP Algorithm

1. Initialization:

   a. Given x^0, τ, η, and d, set $k := 0$.

2. Calculation of the basic trust region step:

   a. While $\|\delta\| < \tau$, iterate (using O3D) on

   $$\begin{aligned} &\min_{\delta} \nabla f(x^k)^{\mathrm{t}} \delta + \tfrac{1}{2} \delta^{\mathrm{t}} B^k \delta \\ &\text{subject to:} \quad \nabla g(x^k)^{\mathrm{t}} \delta + g(x^k) \le 0 \end{aligned}$$

   to obtain δ^k.

   b. Set $q^k = -[\nabla g(x^k)^{\mathrm{t}} \delta^k + g(x^k) + z^k]$.

   c. Decrease d if necessary.

3. Computation of the steplength parameter:

   a. Choose α such that ψ_d^k is reduced.

   b. If both

$$\left\|\bar{c}(x^k + \alpha\delta^k, z^k + \alpha q^k)\right\| \geq \left\|\bar{c}(x^k, z^k)\right\|$$

   and

$$\left\|\bar{c}(x^k, z^k)\right\| > \eta,$$

   reduce α until

$$\left\|\bar{c}(x^k + \alpha\delta^k, z^k + \alpha q^k)\right\| < \left\|\bar{c}(x^k, z^k)\right\|.$$

4. Update of the estimate of the globalization parameter:

   a. If

$$\psi_d(x^k + \alpha\delta^k, z^k + \alpha q^k) > \psi_d(x^k, z^k),$$

   set $\eta = \frac{1}{2}\left\|\bar{c}(x^k, z^k)\right\|$.

5. Update of the variables and check for termination:

   a. Set

$$\begin{aligned} x^{k+1} &:= x^k + \alpha\delta^k \\ z^{k+1} &:= z^k + \alpha q^k. \end{aligned}$$

   b. If convergence criteria are met, quit.

6. Adjustment of the merit function and trust region parameters:

   a. Update d if necessary.

   b. Adjust the trust region radius τ.

7. Return:

   a. Set $k := k + 1$.

   b. Go to Step 2.

The Globalization Parameter: The globalization step is based on work in [BogT89] and [BTK91]. In Step 3a we require that the approximate merit function be reduced and, in addition, if the current iterate lies outside the set $C(\eta)$ we require that the constraint infeasibilities also be reduced. This is possible as a result of Theorems 3.2 and 3.4. If we have a good estimate of η and $(x^k, z^k) \in C(\eta)$ then Theorem 3.3 implies that the true merit function can also be reduced; if this is not the case, then our estimate of η is too large and we reduce its value in Step 4. This procedure will eventually lead to a sufficiently small value of η. Note that this arrangement allows steps that may increase the merit function, but only in a controlled way. It also allows steps that may increase the constraint infeasibilities, but only when inside of $C(\eta)$.

Updating τ: Our procedure for updating τ, the trust region radius, in Step 6b is similar to the standard strategy used in trust region algorithms (see [DS83] or [MS83]) in that we base the decision on how to change τ on a comparison of a predicted relative reduction, $pred_k$, and an actual relative reduction, $ared_k$, in a function used to measure the progress toward the solution. (Various formulas for the predicted relative reduction, $pred_k$, have been suggested for different merit functions, especially for equality constrained programming problems; see, for example, [EA92]). What is distinctive about our procedure is that we use different merit functions for computing $pred_k$ and $ared_k$ depending on the current status of the algorithm.

The determination of predicted and actual reductions is done using the approximate merit function when the current iterate lies outside of $C(\eta)$ since there the approximate merit function is used to determine the steplength. On the other hand, when the current iterate is in $C(\eta)$ then the actual merit function is used to compute the predicted and actual reductions. To compute the predicted reduction when we are using either the actual or approximate merit function we employ the representation of (3.13), writing

$$\psi_d(x, z) = \psi_1(x, z) + \frac{1}{d}\psi_2(x, z)$$

or

$$\psi_d^k(x, z) = \psi_1^k(x, z) + \frac{1}{d}\psi_2^k(x, z).$$

We observe that (QP) can be thought of as using a quadratic approximation to ψ_1 as its objective function and a type of linear approximation to ψ_2 for the constraints. Based on this observation, when $(x^k, z^k) \in C(\eta)$, we define

$$\begin{aligned} pred_k &= \left\{ -\alpha^k \nabla\psi_1(x^k, z^k)^{\mathsf{t}} \begin{pmatrix} \delta^k \\ q^k \end{pmatrix} - \frac{(\alpha^k)^2}{2} \begin{pmatrix} \delta^k \\ q^k \end{pmatrix}^{\mathsf{t}} \nabla_k^2 \psi_1(x^k, z^k) \begin{pmatrix} \delta^k \\ q^k \end{pmatrix} \right. \\ &\quad \left. - \frac{\alpha^k}{d} \nabla\psi_2(x^k, z^k)^{\mathsf{t}} \begin{pmatrix} \delta^k \\ q^k \end{pmatrix} \right\} / \psi_d(x^k, z^k) \end{aligned} \tag{4.1}$$

where the derivatives are with respect to x and z. For (x^k, z^k) outside $C(\eta)$ we define an analogous formula using ψ_1^k and ψ_2^k. In both cases the value of $ared_k$ is the difference in the values of ψ_d (or ψ_d^k) at the points (x^{k+1}, z^{k+1}) and (x^k, z^k) divided by the value of $\psi_d(x^k, z^k)$ (or $\psi_d^k(x^k, z^k)$).

When the actual merit function is being used a valid criticism of the formula for $pred_k$ is its dependence on higher order derivatives. For instance, in computing the derivative of ψ_1, the derivatives of the Lagrange multiplier vector $\bar{\lambda}(x, z)$ and the matrix $\bar{A}(x, z)$ must be computed. These computations require the evaluation of the second derivatives of the constraint functions. We partially mitigate this difficulty by always treating $\bar{A}(x, z)$ as a constant. Because we use finite difference approximations to the Hessian of the Lagrangian, the second order derivatives involved in the computation of $\bar{\lambda}(x, z)$ are available. (This situation is discussed further in Section 6.) If (4.1) is used with the approximate merit function then the $\nabla\bar{\lambda}$ term does not occur and the computation is simpler.

The above choices for $pred_k$ are not used when the trust region constraint causes O3D to terminate in phase I, i.e., if the linearized constraints are inconsistent or are not satisfied. In these situations the resulting step is dominated by a feasibility improving component and it makes little sense for the adjustment to τ to be determined by either of the merit functions; rather, a comparison of the predicted and actual improvement in constraint infeasibility seems more appropriate. Therefore, in this case the function $\bar{c}(x, z)$ is used for comparison puposes. The values of $pred_k$ and $ared_k$ are given as follows for the case when the O3D algorithm terminates in phase I:

$$pred_k = \left\{ \left\| c(x^k, z^k) \right\|^2 - \left\| \alpha^k \nabla g(x^k)^{\mathsf{t}} \delta^k + g(x^k) + z^{k+1} \right\|^2 \right\} / \left\| c(x^k, z^k) \right\|^2$$

and

$$ared_k = \left\{ \left\| c(x^k, z^k) \right\|^2 - \left\| c(x^{k+1}, z^{k+1}) \right\|^2 \right\} / \left\| c(x^k, z^k) \right\|^2 .$$

These heuristics for choosing $pred_k$ and $ared_k$ appear to work well. Specifically, they allow the trust region radius, τ, to be increased even in the event that the step returned by O3D does not satisfy linearized constraints or it results in an increase in one of the merit functions. In our experience, the alternative formulas based solely on constraint violations never are employed close to the solution. Indeed, the iterates preceding convergence have always been observed to be well inside $C(\eta)$ where satisfying the linearized constraints and decreasing the merit functions poses no problem.

The steplength α: Since we expect our trust region procedure to allow a steplength of one much of the time, we do not include a sophisticated line search. Instead we perform a simple backtracking procedure, using a factor of one half, until an Armijo condition is satisfied.

Adjusting d: Choosing a good value for the merit function parameter d is essential in our algorithm. While it is clear that (in a compact set) a sufficiently small value of d will assure that the results given in Section 3 will hold, there are three very important practical reasons why the parameter must be adjusted rather than fixed. First, if the angle between the directions generated by O3D and the gradients of the merit function becomes nearly orthogonal the steps might become too small. We adjust d to avoid this possibility. Second, the *approximate* merit function, ψ_d^k, is changing at each iteration and it is possible a previous iterate might be acceptable to the current ψ_d^k, i.e., cycling might occur. This worry can also be alleviated by adjusting d. A third reason for changing d is to allow for larger steps. It is seen from the theory and has been verified by numerical experience that if d is too small then the form of the merit function forces the path of the iterates to follow the "nearly active" constraints closely. This causes the algorithm to take very small steps and, in particular, to be slow in moving away from a nonoptimal active set. By making it possible to increase d we can significantly improve the algorithm's performance.

In the implementation of our algorithm there are two opportunities to adjust d: in Step 2, after solving the quadratic subproblem, and in Step 6, after the step has been taken. In the first of these adjustments d can only be decreased; in the second, the parameter may be increased or decreased.

In Step 2, the angle between the gradient of the approximate merit function $(\nabla\psi_d^k)^{\mathrm{t}}$ and the step direction $(\delta^k, q^k)^{\mathrm{t}}$ is computed. If this angle is too large, we conclude that d is not small enough to ensure a good decrease in ψ_d^k, and we decrease the parameter. To be more specific, we compute

$$w(d) = \frac{(\nabla \psi_d^k)^{\mathrm{t}} \begin{pmatrix} \delta^k \\ q^k \end{pmatrix}}{\left\|\nabla \psi_d^k\right\| \cdot \|(\delta^k, q^k)\|}.$$

If $w(d) \geq -.1$ we calculate a value $\hat{d}$ so that $w(\hat{d}) \approx -.5$. This can be easily accomplished by fixing the denominator. We safeguard the procedure by not allowing more than a certain percentage decrease, currently 50%.

If d was not decreased in Step 2 we consider modifying it after a step has been taken (Step 6). Here the primary concern is to avoid cycling. To do so we compute an interval of values for the penalty parameter as follows. For a fixed integer κ we seek a value of the parameter, $\bar{d}$, such that

$$\psi_{\bar{d}}^k(x^k, z^k) < \psi_{\bar{d}}^k(x^{k-i}, z^{k-i}), \qquad i = 1, \ldots, \kappa. \tag{4.2}$$

Inequality (4.2) implies that none of the past κ iterates will be acceptable to the approximate merit function with the new value of $\bar{d}$. (Thus if $\kappa = k$ no cycling would be possible). To accomplish this, we use the decomposition of (3.13)

$$\psi_d^k = \psi_1^k + \frac{1}{d}\psi_2^k. \tag{4.3}$$

We then compute the values of $\psi_1^k(x^{k-i}, z^{k-i})$ and $\psi_2^k(x^{k-i}, z^{k-i})$, $i = 1, \ldots, \kappa$, and consider the inequalities

$$\psi_1^k(x^k, z^k) + \frac{1}{d}\psi_2^k(x^k, z^k) < \psi_1^k(x^{k-i}, z^{k-i}) + \frac{1}{d}\psi_2^k(x^{k-i}, z^{k-i}). \tag{4.4}$$

We define d_i^u and d_i^l to be the upper and lower values of d that ensure the inequality is satisfied. Then letting

$$d^u = \min\{d_i^u : i = 1, \ldots, \kappa\} \tag{4.5}$$

and

$$d^l = \max\{d_i^l : i = 1, \ldots, \kappa\} \tag{4.6}$$

we obtain an interval (d^l, d^u). Assuming that this interval exists it is the case that if the value of d for the next step is chosen in this interval, the next iterate will not return to one of the previous κ iterates. In practice a value of $\kappa \approx 5$ is usually more than sufficient to prevent cycling. If the interval doesn't exist, then we make no change.

Given that we can choose d to avoid cycling, our second objective at this juncture is to increase d to allow bigger steps. If the d^u is larger than the current d then we can safely increase d without worrying about possible cycling. However, we safeguard this increase in two ways. First, we require that the predicted reduction based on the approximate merit function must be greater than the predicted reduction in the linearized constraints. This restriction prevents d from being increased prematurely due primarily to a large decrease in constraint infeasibilites. Specifically, writing the predicted reduction in ψ_d^k (as given in (4.1)) as

$$\mathcal{P}_{\mathcal{Q}} + \frac{1}{d}\mathcal{P}_{\mathcal{L}},$$

we insist that for a new value of d

$$\mathcal{P}_{\mathcal{Q}} + \frac{1}{d}\mathcal{P}_{\mathcal{L}} > \mathcal{P}_{\mathcal{L}}. \tag{4.7}$$

Second, we use a maximum allowable change much as in Step 2 to limit the growth of d. Computationally, these simple procedures for updating d appear to be effective, especially in the presence of highly nonlinear constraints and poorly scaled problems.

Convergence Criteria: The convergence criteria used are standard, and similar to those in [BogTK91]. We first insist that the constraints be satisfied to a close tolerance; specifically we require

$$\|\max(g(x^k), 0)\|_\infty \leq 10^{-6}. \tag{4.8}$$

We also require that either

$$\frac{\|\nabla f(x^k) + \nabla g(x^k)\lambda^k\|}{|f(x^k)|} \leq 10^{-7} \tag{4.9}$$

or

$$\|x^k - x^{k-1}\|_\infty \leq 10^{-8}(1 + \left\|x^k\right\|). \tag{4.10}$$

The criterion (4.9) is usually preferable to criterion (4.10). The Lagrange multipliers returned by the quadratic program are used in the convergence tests above unless the trust region constraint determines the approximate solution of the (QP). In that case, we use the least squares approximation to the multipliers, replacing all negative multipliers with machine zeros. In all of the problems solved to date, the trust region never comes into play when the iterates get close to the solution; therefore the (QP) multipliers are used for the convergence test at the solution.

As mentioned at the beginning of this section, in any practical algorithm provisions must be made to handle the "pathological" situations that can arise in a given problem. In the following we comment on our attempts to handle some of these potential difficulties.

Inconsistent $(\boldsymbol{QP})$: One difficulty that occurs frequently when making linear approximations to nonlinear constraints is the feasibility or consistency of the (QP) . In the case of inconsistency, O3D will, even if it runs to completion, not exit Phase I and will return a positive value of the artificial variable. Nevertheless, by Theorem 3.2, the direction is always a descent direction for ψ_d^k and for $\|\bar{c}\|^2$. As a result, the step taken in this direction will almost always decrease infeasibility, making it less likely that an inconsistent set of linearized constraints will be encountered during subsequent iterations.

The major difficulty for our algorithm when the (QP) is inconsistent involves the non-negative slack variables, z^k. Our slack variable updating scheme ensures that non-negative slacks remain non-negative, *provided* the linearized constraints are satisfied. When an inconsistent (QP) is encountered we replace any negative slacks using the following rule:

If $z_i^{k+1} < 0$ then set

$$z_i^{k+1} = \begin{cases} \epsilon_{Mach} & g_i(x^{k+1}) \geq 0 \\ -g_i(x^{k+1}) & g_i(x^{k+1}) < 0 \end{cases}$$

where ϵ_{Mach} is machine epsilon. This is sometimes referred to as 'closing' the constraints (see for example [Tap80]).

The Hessian Matrix: While the Hessian of the Lagrangian at the solution is known to be positive definite on the appropriate subspace, it may be indefinite in general. We experimented with two different approaches to handle this possibility. First, we simply modified the Hessian matrix by adding non-negative elements to the diagonal of the Hessian to ensure that the Choleski factorization of the matrix had positive elements along its diagonal (see [GilMW81]). This modification was easy to implement, but it was observed to slow convergence on some problems. While it guarantees that a positive definite matrix will be delivered to the (QP) solver, if this modification takes place when iterates get close to the solution, it generally precludes local q-superlinear convergence.

An alternative to modifying the Hessian of the Lagrangian is simply to allow O3D to iterate on the indefinite QP subproblem, halting the iterations when the solution exceeds the trust region radius. We implemented this approach and it seemed to yield superior results to those obtained by making the Hessian positive definite (especially when the iterates are close were close to a solution) even though, theoretically, we can only prove that we obtain a descent direction when the Hessian is positive definite.

Although in many of the applications some analytic derivatives were available, no use of analytic derivative information was used in these numerical experiments. When possible, first and second derivatives were computed using forward and central finite differences respectively. A costly one-time calculation provided a zero/non-zero stencil of the Hessian of the Lagrangian and the Jacobian matrix of the constraint function. These stencils were then used for the duration of the solution process. For some problems, these finite difference approximations are not convenient to use. This can be the case with control problems governed by partial differential equations (see [Lio71] or [Lio88]). If the partial differential equation is solved using a finite element method, with piecewise linear elements, then evaluating the derivative of the objective function with respect to the control variables can be quite cumbersome. In these cases, which occured in the control problems in our test suite, one can approximate the first derivatives of the objective function by solving an adjoint problem with a computational cost comparable to one function evaluation. (For examples, see [Glo84]). The objective function portion of the Hessian of the Lagrangian can then be approximated with forward finite differences.

Linearly Dependent Constraint Gradients: Linearly dependent constraint gradients cause many theoretical and computational difficulties in constrained optimization. In our algorithm, even though O3D has no difficulty in dealing with these problems, evaluating the merit function and computing the least squares approximation to the Lagrange multipliers become problematical. While there is no theory to support this algorithm's performance on problems with linearly dependent constraint gradients, it has been successful on problems with *some* degeneracy in the constraints. We simply make the slack variables sufficiently positive (which allows us to factor the crucial matrices) and continue. However, the algorithm failed to solve some problems that had a large amount of degeneracy in the linearized constraint matrix. This was, of course, problem dependent but it was observed that the current implementation can usually solve problems with up to 10 percent of the constraint gradients being linearly dependent. This degeneracy causes the performance of the merit functions to deteriorate. In particular, the least squares approximation to Lagrange multipliers seems to be especially poor, resulting in only very small steps being allowed, even close to the solution.

5. Numerical Results

The modified algorithm was coded in Fortran and is installed on a SPARCstation 10 using *IEEE floating point arithmetic (64 bit)*. The current implementation has being used to solve a wide variety of medium to large scale problems. In this section we report the results of a set of performance tests designed specifically to answer questions about the trust region strategy and the procedure to update the penalty parameter, d.

A set of eight problems was chosen as the test suite. These problems ranged in size from 500–1000 variables and from 1000–2000 constraints. The first four are relatively straightforward nonlinear programming test examples, while the last four are from actual applications: two discretized control problems, a density estimation problem from statistics, and a "molecular distance" problem. A more complete description of these problems is found in the Appendix. The problems all have nonlinear inequality constraints and exploitable sparsity. Problem 4 (NLP4) was designed to have a controllable percentage of linear dependency in the constraint gradients to demonstrate the weaknesses in the algorithm associated with this difficulty. We ran three versions of our algorithm on each problem; using a positive definite modification of the Hessian matrix, as discussed in Section 4, with and without the trust region strategy and using the unmodified Hessian with the trust region. (Using the unmodified Hessian results in failure in most cases if no trust region strategy is employed.) In addition, each problem was run from two starting points; one, labeled "c", which was close to the solution in the sense that each of the variables was of the same order of magnitude as in the solution and a distant start, labeled "f".

The results of the numerical tests on these problems are summarized in Tables 1–3. The first two columns of each table gives the number of SQP iterations ("nl-i") and the total number of O3D iterations ("qp-i"). The next two columns contain the stopping criterion that was met and the value of the gradient of the Lagrangian at the solution. Unless the algorithm failed, (which is denoted by "Failure" in the tables) feasibility condition (4.8) was satisfied for all solutions. The stopping criterion is denoted by either a "1" or a "2" depending on whether

(4.9) or (4.10) was satisfied. If both were conditions were satisfied, a "3" appears in the column. The remaining columns give information about the values of the parameter d for each run; columns five through eight giving the initial, maximum, minimum, and final values of this parameter and the final column giving the last iteration at which d was changed.

The results of the tests illustrate that using the unmodified Hessian with the trust region was most effective in reducing the number of O3D iterations *and* the number of SQP iterations. The trust region strategy prevented long, unprofitable steps from being generated when far from the solution and the use of the unmodified Hessian allowed the trust region to become inactive near the solution thus allowing rapid local convergence. Requiring the Hessian to be positive definite often precluded rapid local (q-superlinear) convergence and, when used in conjunction with the trust region strategy, resulted in the trust region's being active close to the solution.

The results also show that the value of the parameter d varied over several orders of magnitude. The procedures discussed in Section 4 that allowed the value of d to increase or decrease greatly enhanced the algorithm; earlier tests using either a fixed value of d or only allowing a reduction in d yielded inferior results.

Another modification in our algorithm, not reflected in the table or included in the description in the preceding section, was made to force the O3D algorithm to take a minimum number of steps. We found that when the trust region radius τ became small the algorithm would sometimes exit O3D after only one iteration, resulting in a poor step direction. This poor step would result in a further decrease in τ, and eventually the algorithm would fail. When we required a minimum number of steps to be taken in O3D (our choice was 7) this problem disappeared.

Problem	nl-i	qp-i	conv	$\|\nabla_x l\|_\infty$	d_0	$\max d$	$\min d$	final d	last d-cha
NLP1 - c	37	1435	2	1.2e-7	1.00e00	2.08e00	4.45e-2	9.35e-1	34
NLP1 - f	49	1656	1	7.3e-8	1.00e00	1.07e00	5.89e-2	9.20e-1	46
NLP2- c	66	2211	1	1.2e-7	1.00e00	2.83e00	6.89e-2	1.72e00	61
NLP2- f	71	2369	1	3.1e-8	6.98e-1	3.00e00	8.31e-2	1.51e00	64
NLP3-c	29	983	1	6.7e-8	1.00e00	1.12e00	8.13e-1	1.03e00	22
NLP3-f	39	1314	1	4.2e-8	1.00e00	1.05e00	9.84e-1	1.03e00	31
NLP4- c	–	6		Failure	1.00e00	-	-	-	failure
NLP4 -f	–	6		Failure	1.00e00	-	-	-	failure
Truss - c	103	3561	1	4.4e-8	9.87e-1	1.01e00	9.57e-2	9.57e-1	100
Truss - f	110	3799	2	1.9e-7	1.00e00	1.08e00	8.93e-2	8.93e-1	106
Stat - c	135	4561	3	1.1e-8	1.00e00	2.33e00	8.25e-2	9.70e-1	129
Stat - f	144	4805	1	3.3e-8	1.00e00	2.16e00	7.77e-2	8.49e-1	140
BCHeat-c	257	5398	1	7.8e-8	9.18e-1	1.98e00	5.23e-2	1.24e00	254
BCHeat-f	289	5971	1	1.9e-7	1.00e00	4.21e00	4.92e-2	1.37e00	281
Molec-c	37	1376	1	9.8e-9	9.88e-1	1.21e1	1.17e-2	9.81e-1	34
Molec-f	41	1437	2	6.5e-6	1.00e00	2.38e0	1.49e-1	5.52e-1	39

Table 1: **Modified Hessians with no trust region**

Problem	nl-i	qp-i	conv	$\|\nabla_x l\|_\infty$	d_0	$\max d$	$\min d$	final d	last d-cha
NLP1- c	94	1412	2	3.2e-7	1.00e00	8.58e00	3.13e-3	6.55e-2	88
NLP1- f	108	2947	2	6.8e-8	1.00e00	7.50e00	1.23e-3	2.60e-2	99
NLP2 - c	213	2744	1	1.6e-7	1.00e00	3.85e00	1.28e-3	6.48e-1	209
NLP2-f	231	2963	1	5.4e-8	6.98e-1	1.10e01	5.67e-3	8.57e-1	221
NLP3-c	42	932	1	3.7e-8	1.00e00	1.64e00	4.27e-1	9.83e-1	39
NLP3-f	44	946	1	9.1e-8	1.00e00	1.53e00	2.71e-1	9.22e-1	38
NLP4- c	199	2582	2	9.2e-6	1.00e00	1.41e00	8.92e-1	1.18e00	49
NLP4-f	201	2599	2	5.2e-7	1.00e00	2.01e00	9.94e-1	1.21e00	55
Truss - c	195	3528	1	6.1e-8	9.87e-1	1.13e00	9.87e-1	1.11e00	189
Truss - f	195	3544	2	2.9e-7	1.00e00	1.47e00	1.00e00	1.47e00	188
Stat - c	144	4519	3	2.3e-8	1.00e00	2.39e00	1.00e00	1.53e00	140
Stat-f	150	4581	1	4.2e-8	1.00e00	2.48e00	1.00e00	1.89e00	144
BCHeat -c	257	2898	1	8.1e-8	9.18e-1	4.15e00	1.38e-1	4.10e00	249
BCHeat -f	289	3071	1	9.9e-8	1.00e00	3.74e00	2.44e-1	3.89e00	281
Molec-c	39	546	1	1.3e-7	9.88e-1	1.71e01	3.74e-2	2.22e00	36
Molec-f	44	621	1	6.6e-8	1.00e00	1.48e00	7.39e-2	9.52e-2	38

Table 2: **Modified Hessians with trust region**

Problem	nl-i	qp-i	conv	$\|\nabla_x l\|_\infty$	d_0	$\max d$	$\min d$	final d	last d-cha
NLP1-c	43	820	2	2.1e-7	1.00e00	3.69e00	5.13e-2	7.13e-1	38
NLP1-f	46	913	3	1.7e-8	1.00e00	6.58e00	6.27e-2	6.14e-1	39
NLP2-c	51	1330	1	9.8e-8	1.00e00	4.11e00	9.65e-2	5.95e-1	44
NLP2-f	53	1351	1	1.1e-7	6.98e-1	2.94e00	1.20e-1	2.47e-1	48
NLP3-c	35	832	1	4.5e-8	1.00e00	1.89e00	2.46e-2	3.79e-1	29
NLP3-f	39	867	1	7.3e-8	1.00e00	1.57e00	2.22e-2	4.52e-1	28
NLP4-c	–	–		Failure	-	-	-	-	failure
NLP4-f	–	–		Failure	-	-	-	-	failure
Truss-c	94	2242	1	3.9e-8	9.87e-1	1.03e00	1.26e-1	9.11e-1	87
Truss-f	96	2261	1	6.6e-8	1.00e00	1.33e00	5.67e-2	7.84e-1	85
Stat-c	121	1577	3	1.1e-8	1.00e00	2.58e00	1.57e-1	9.34e00	114
Stat-f	121	1585	1	4.7e-8	1.00e00	2.19e00	1.65e-1	7.03e00	111
BCHeat-c	231	2498	1	1.2e-7	9.18e-1	3.24e00	6.04e-2	8.83e-1	226
BCHeat-f	239	2871	3	2.4e-8	1.00e00	1.61e01	3.89e-2	4.98e-1	222
Molec-c	39	550	1	8.76e-8	9.88e-1	1.02e00	4.34e-2	6.04e-1	36
Molec-f	45	658	2	2.3e-7	1.00e00	8.78e00	1.35e-2	6.53e-1	42

Table 3: **Un-modified Hessians with trust region**

6. Future Directions

Here we briefly mention areas where improvements in the theory and the implementation of our algorithm can be made.

First, the theory is incomplete. We need to develop a global convergence theory under relatively weak conditions. Specifically, many of the problems that we have seen have been degenerate and this significantly slows the convergence of the method. The primary culprit is the extremely poor multiplier estimates provided by the least squares procedure.

Algorithmically, we observe that the current implementation requires the factorization of both $(\nabla g^t \nabla g + Z)$ and $(\nabla g \nabla g^t)$, the latter in O3D. While the sparse matrix package makes this reasonable for the problems that we have currently considered, it is clearly expensive to maintain both.

Finally, our current implementation uses analytic or finite difference Hessian approximations. An examination of the details of O3D reveals that a limited memory BFGS or limited memory SR1 could be readily incorporated into the code. Alternatives include using some automatic differentiation techniques for estimating these derivatives.

References

[BDRW91] Paul T. Boggs, Paul D. Domich, Janet E. Rogers, and Christoph Witzgall. An interior point method for linear and quadratic programming problems. *Mathematical Programming Society Committee on Algorithms Newsletter*, (19):32–40, 1991.

[BDRW94] Paul T. Boggs, Paul D. Domich, Janet E. Rogers, and Christoph Witzgall. An interior-point method for general large scale quadratic programming problems. Internal Report (in process), National Institute of Standards and Technology, 1994.

[BP91] J. Burger and M. Pogu. Functional and numerical solution of a control problem originating from heat transfer. *Journal of Optimization Theory and Applications*, 68:49–73, 1991.

[BT84] Paul T. Boggs and Jon W. Tolle. A family of descent functions for constrained optimization. *SIAM Journal on Numerical Analysis*, 21:1146–1161, 1984.

[BT89] Paul T. Boggs and Jon W. Tolle. A strategy for global convergence in a sequential quadratic programming algorithm. *SIAM Journal on Numerical Analysis*, 21:600–623, 1989.

[BTK] Paul T. Boggs, Jon W. Tolle, and Anthony J. Kearsley. A truncated SQP algorithm for large scale nonlinear programming problems. In *Proceedings of the Sixth Conference on Numerical Analysis and Optimization (To Appear).*

[BTK91] Paul T. Boggs, Jon W. Tolle, and Anthony J. Kearsley. A merit function for inequality constrained nonlinear programming problems. Internal Report 4702, National Institute of Standards and Technology, 1991.

[CGT92a] A. R. Conn, N. I. M. Gould, and Ph. T. Toint. *Lancelot: A Fortran Package for Large-Scale Nonlinear Optimization*, volume 17 of *Series in Computational Mathematics*. Springer-Verlag, Heidelberg and New York, 1992.

[CGT92b] A. R. Conn, N. I. M. Gould, and Ph. T. Toint. Large-scale nonlinear constrained optimization. Technical Report 92-02, Facultés Universitaires de Namur, Département de Mathématique, 1992.

[Col] Thomas F. Coleman. Large scale numerical optimization: Introduction and overview. In *Encyclopedia of Computer Science and Technology*. Marcel Dekker (to appear), New York.

[DBRW91] Paul D. Domich, Paul T. Boggs, Janet E. Rogers, and Christoph Witzgall. Optimizing over three-dimensional subspaces in an interior-point method for linear programming. *Linear Algebra and its Applications*, 152:315–342, July 1991.

[DES82] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19:400–408, 1982.

[DS83] John E. Dennis Jr. and Robert. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[EA92] M. M. El-Alem. A robust trust region algorithm with nonmonotonic penalty parameter scheme for constrained optimization. Department of Mathematical Sciences 92-30, Rice University, 1992.

[EW94] Stanley C. Eisenstat and Homer Walker. Globally convergent inexact Newton methods. *SIAM Journal on Optimization*, 4:(to appear), 1994.

[GHR93] W. Glunt, T. L. Hayden, and M. Raydan. Molecular conformations from distance matrices. *Journal of Computational Chemistry*, 14:114–120, 1993.

[Glo84] Roland Glowinski. *Numerical Methods for Nonlinear Variational Problems*. Springer-Verlag, Berlin, 1984.

[GMW81] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, Inc., 1981.

[KS92] F.-S. Kupfer and E. W. Sachs. Numerical solution of a nonlinear parabolic control problem by a reduced sqp method. *Computational Optimization and Applications*, 1:113–135, 1992.

[Lio71] J. L. Lions. *Optimal Control of Systems Governed by Partial Differential Equations.* Springer-Verlag, Berlin, 1971.

[Lio88] J. L. Lions. Exact controllability, stabilization and perturbations for distributed systems. *SIAM Review*, 30:1–68, 1988.

[MS83] Jorge Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal of Scientific and Statistical Computing*, 4:553–572, 1983.

[Tap80] Richard A. Tapia. On the role of slack variables in quasi-Newton methods for constrained optimization. In L. C. W. Dixon and G. P. Szegö, editors, *Numerical Optimization of Dynamical Systems*, pages 235–246, Amsterdam, 1980. North-Holland.

[TT78] Richard A. Tapia and James R. Thompson. *Nonparametric Probability Density Estimation.* Johns Hopkins, Baltimore, 1978.

[TT90] Richard A. Tapia and James R. Thompson. *Nonparametric Function Estimation, Modeling, and Simulation.* SIAM, Philadelphia, 1990.

7. Appendix

Nonlinear Program # 1 (NLP1)

$$
\begin{aligned}
&\min f(x) = \frac{1}{2}((x_1 - x_{100})x_2 + x_{101})^2 \\
&\text{subject to:} \\
&\quad x_1 x_{i+1} + (1 + \frac{2}{i})x_i x_{100} + x_{101} \le 0, \qquad i = 1, \ldots, 99 \\
&\quad (\sin(x_j))^2 - \frac{1}{2} \le 0, \qquad j = 1, \ldots, 100 \\
&\quad \sin((x_j)^2) \ge 0, \qquad j = 1, \ldots, 100 \\
&\quad x_j \le j, \qquad j = 1, \ldots, 100 \\
&\quad -x_j \le 1, \qquad j = 1, \ldots, 100 \\
&\quad (x_1 + x_{100})^2 = 1
\end{aligned}
$$

Explanation: This problem with 101 variables and 500 constraints is taken from [CGT92a] where it was used to illustrate separability in nonlinear programming.

Nonlinear Program # 2 (NLP2)

$$
\begin{aligned}
&\min f(x) = 1000 \left[\left(\sum_{i=1}^{n} x_i^3 \right)^2 - \left(\sum_{i=1}^{n} x_i^2 \right) \left(\sum_{i=1}^{n} x_i^4 \right) \right] \\
&\text{subject to:} \\
&\quad x_1 \ge 0 \quad \text{and} \quad x_n \le 1 \\
&\quad x_i - x_{i+1} \le 0, \qquad i = 1, \ldots, (n-1) \\
&\quad x_i^2 - x_i x_{i+1}^2 \le 0, \qquad i = 1, \ldots, (n-1)
\end{aligned}
$$

Explantion: There are many local extrema for this problem; we made no special effort to locate global minima. The objective function is highly nonlinear and has a dense Hessian, but the constraints have sparse banded first derivatives. Our example uses $n = 250$.

Nonlinear Program # 3 (NLP3)

$$
\begin{aligned}
&\min f(x) = \sum_{i=1}^{n} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right] \\
&\text{subject to:} \\
&\qquad x_1 \geq 0 \quad \text{and} \quad x_n \geq 0 \\
&\qquad x_i - x_{i+1} \leq 0, \qquad i = 1, 3 \ldots, (n-1) \\
&\qquad 4x_{i+1} - x_i^2 - 4 \leq 0 \qquad i = 1, 3 \ldots, (n-1) \\
&\qquad 2x_{i+1} + x_i - 1 \leq 0 \qquad i = 1, 3 \ldots, (n-1)
\end{aligned}
$$

Explantion: The objective function here is Rosenbrock's function. The objective function is highly nonlinear and has a dense Hessian. The constraints have sparse banded first derivatives. We solved the problem with $n = 250$.

Nonlinear Program # 4 (NLP4)

$$
\begin{aligned}
&\min f(x) = x^{\mathsf{t}} L^2 x \\
&\text{subject to:} \\
&\qquad i - (x_i^2 + x_{2i}^2) \leq 0, \qquad i = 1, \ldots, (n/2) \\
&\qquad \sqrt{2i} - (x_i + x_{2i}) \leq 0, \qquad i = 1, \ldots, (n/2) \\
&\qquad \log(x_i + x_{i+1} + x_{i+2}) - x_i + x_{i+1} + x_{i+2} \leq 0 \\
&\qquad i = 1, \ldots, n-2
\end{aligned}
$$

Explantion: The matrix L is the discretized tridiagonal Laplacian operator so the objective function is convex and quadratic. The constraints are nonlinear and the gradients of the active constraints at the solution are linearly dependent. The problem on which we reported results has $n = 1000$.

Truss Problem (Truss)

$$
\begin{aligned}
&\min_x \rho(c^{\mathsf{t}} x) \\
&\text{subject to:} \\
&\qquad S(x)^{-1} F - b \leq 0 \\
&\qquad X(x) G S(x)^{-1} F - \beta x \leq 0
\end{aligned}
$$

Explantion: This problem chooses the state variables $x \in \mathcal{R}^n$ to minimize the weight of an optimal n-bar truss design, subject to constraints on the deflection and stress of the truss. The function ρ is the density of the material and in our problem was a non-convex polynomial, $\rho(\zeta) = \zeta^4 - \zeta^2 + 1$. c is a vector containing the lengths of the bars in the truss. The matrix S is the positive definite stiffness matrix, G is a matrix that represents the geometry of the truss and design and F is the vector of applied forces. The vector b and scalar β form bounds on the maximum allowable deflections in the state variables, and the maximum allowable stress in the truss. We solved a problem with $n = 500$ and 1500 constraints.

Maximum Penalized Likelihood Estimate (Stat)

$$
\begin{aligned}
&\min_x f(x(t)) = -\prod_{i=1}^{n} x(t) e^{\phi_\mu(x(t))} \\
&\text{subject to:} \\
&\qquad x \in H^2(-\infty, \infty) \\
&\qquad \int_{-\infty}^{\infty} x(t)^2 dt = 1 \\
&\qquad -x(t) \leq 0 \quad \text{for all} \quad t
\end{aligned}
$$

Explantion: This particular maximum penalized likelihood estimator is sometimes referred to as 'the second estimate of Gaskins and Good' (see, e.g., [TT78] or [TT90]). We discretize this problem by taking a finite random sample of t_i's say, $t_i \in [\alpha, \beta]$. $\phi(x)$ is defined by

$$\phi(x) = \alpha \int_{-\infty}^{\infty} x'(t)^2 dt + \beta \int_{-\infty}^{\infty} x''(t)^2 dt \tag{7.1}$$

and given $\mu > 0$ the regularized function $\phi_\mu(x)$ is defined by

$$\phi_\mu(x) = \phi(x)\mu \int_{-\infty}^{\infty} x(t)^2 dt. \tag{7.2}$$

The discrete approximate of $x(t)$ was taken to be a cubic spline. The resulting problem had 500 variables and 1000 constraints.

Boundary Control of Heat Equation (BCHeat)

$$\min f(x,y) = \int_0^T (x(1,t) - x_d(t))^2 + ay(t)$$

subject to:

$$C(x(z,t))x_t(z,t) - \nabla(\lambda(x(z,t))\nabla x(z,t)) = f(z,t) \quad \text{on} \quad \Omega \times [0,T]$$
$$\lambda(x(z,t))\nabla x(z,t) = b(z,t) \quad \text{on} \quad \partial\Omega \times [0,T]$$
$$x(z,0) = x_0 \quad \text{on} \quad \Omega$$
$$x \in L^2(0,T;H^1(\Omega))$$
$$y \in L^2(0,T)$$

Explantion: This problem has been solved by Newton's method (see [BP91]) and by reduced methods (see [KS92]). $x_d(t)$ is the desired profile of $x(t)$ and Ω is a square in $\Re^2$. The inequality constraints are quadratic and linear, and arise from enforcing the space conditions $x \in L^2(0,T) \times H^1(\Omega))$ and $y \in L^2(0,T)$. Our discretization results in 500 variables and 1200 constraints.

Molecule Distance Problem (Molec)

$$\min_{x \in \mathcal{R}^{3d}} \|\Delta - D(X)\|_F$$

subject to:

$$a_{ij} \leq D_{ij} \leq b_{ij}$$

Explanation: Here $\Delta, D(X), a, b \in \mathcal{R}^{m \times m}$ and $X \in \mathcal{R}^{n \times 3}$ where n is the number of atoms and m is the number of interatomic distances ($2m = n^2 - n$). Δ is a set of observed data, X is a configuration of atoms (their locations in $\Re^3$, and D is a transformation into the space of "distance matrices". The bound matrices a, b are upper and lower bounds based on estimating errors in measurements. This problem arises in the processing of NMR data for visualization of large proteins and organic molecules (see, e.g., [GHR93]). The results in the tables correspond to a problem we solved with 100 variables and 5000 constraints.