SNOPT: AN SQP ALGORITHM FOR LARGE-SCALE CONSTRAINED OPTIMIZATION*

PHILIP E. GILL†, WALTER MURRAY‡, AND MICHAEL A. SAUNDERS‡

Abstract. Sequential quadratic programming (SQP) methods have proved highly effective for solving constrained optimization problems with smooth nonlinear functions in the objective and constraints. Here we consider problems with general inequality constraints (linear and nonlinear). We assume that first derivatives are available, and that the constraint gradients are sparse.

We discuss an SQP algorithm that uses a smooth augmented Lagrangian merit function and makes explicit provision for infeasibility in the original problem and the QP subproblems. SNOPT is a particular implementation that makes use of a semidefinite QP solver. It is based on a limited-memory quasi-Newton approximation to the Hessian of the Lagrangian, and uses a reduced-Hessian algorithm (SQOPT) for solving the QP subproblems. It is designed for problems with many thousands of constraints and variables but a moderate number of degrees of freedom (say, up to 2000). Numerical results are given for most problems in the CUTE test collection (about 800 smooth problems) and for a number of other applications, including trajectory optimization in the aerospace industry.

As much as possible, we isolate the SQP algorithm from the method used to solve the QP subproblems. In this way SNOPT may be extended to allow for an arbitrary number of active constraints or degrees of freedom, given an appropriate QP solver.

Key words. large-scale optimization, nonlinear programming, nonlinear inequality constraints, sequential quadratic programming, quasi-Newton methods, limited-memory methods

AMS subject classifications. 49J20, 49J15, 49M37, 49D37, 65F05, 65K05, 90C30

1. Introduction.

We present a sequential quadratic programming method for solving large-scale optimization problems involving linear and nonlinear inequality constraints. SQP methods have proved reliable and efficient for many such problems. For example, under mild conditions the general-purpose solvers NLPQL [48] and NPSOL [25, 28] typically find a (local) optimum from an arbitrary starting point, and they require relatively few evaluations of the problem functions and gradients compared to other solvers.

1.1. The optimization problem.

The algorithm we describe applies to constrained optimization problems of the form

$$\text{NP} \qquad \underset{x}{\text{minimize}} \quad f(x)$$
$$\text{subject to} \quad l \le \begin{pmatrix} x \\ c(x) \\ Ax \end{pmatrix} \le u,$$

where $x \in \mathbb{R}^n$, $f(x)$ is a linear or nonlinear objective function, $c(x)$ is a vector of nonlinear constraint functions $c_i(x)$ with sparse derivatives, A is a sparse matrix, and l and u are given bounds.

*This paper may be referenced as Report NA 97-2, Dept of Mathematics, University of California, San Diego, and Report SOL 97-3, Dept of EESOR, Stanford University. This research was partially supported by National Science Foundation grants DMI-9204208, DMI-9204547 and DMI-9424639, and Office of Naval Research grants N00014-90-J-1242 and N00014-96-1-0274.

†Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112 (pgill@ucsd.edu).

‡Department of EESOR, Stanford University, Stanford, CA 94305-4023 (walter@SOL-walter.stanford.edu, mike@SOL-michael.stanford.edu).

We assume that the nonlinear functions are smooth and that their first derivatives are available (and possibly expensive to evaluate). In the present implementation we assume that the number of active constraints at a solution is reasonably close to n. In other words, the number of degrees of freedom is not too large (say, less than 1000).

Important examples are control problems such as those arising in optimal trajectory calculations. For several years, the optimal trajectory system OTIS (Hargraves and Paris [32]) has been applied successfully within the aerospace industry, using NPSOL to solve the associated optimization problems. Although NPSOL has solved examples with over a thousand variables and constraints, it was not designed for large problems with sparse constraint derivatives. (The Jacobian of $c(x)$ is treated as a dense matrix.) Our aim here is to describe an SQP method that has the favorable theoretical properties of the NPSOL algorithm, but is suitable for large sparse problems such as those arising in trajectory calculations. The implementation is called SNOPT (Sparse Nonlinear Optimizer).

1.2. Infeasible constraints. SNOPT makes explicit allowance for infeasible constraints. Infeasible linear constraints are detected first by solving a problem of the form

$$\boxed{\begin{array}{ll} \text{FLP} & \underset{x,v,w}{\text{minimize}} \quad e^T(v+w) \\ & \text{subject to} \quad l \le \begin{pmatrix} x \\ Ax - v + w \end{pmatrix} \le u, \quad v \ge 0, \quad w \ge 0, \end{array}}$$

where e is a vector of ones. This is equivalent to minimizing the one-norm of the general linear constraint violations subject to the simple bounds. (In the linear programming literature, the approach is often called *elastic programming*. Other algorithms based on minimizing one-norms of infeasibilities are given by Conn [13] and Bartels [1].)

If the linear constraints are infeasible ($v \neq 0$ or $w \neq 0$), SNOPT terminates without computing the nonlinear functions. Otherwise, all subsequent iterates satisfy the linear constraints. (As with NPSOL, such a strategy allows linear constraints to be used to define a region in which f and c can be safely evaluated.)

SNOPT then proceeds to solve NP as given, using QP subproblems based on linearizations of the nonlinear constraints. If a QP subproblem proves to be infeasible or unbounded (or if the Lagrange multiplier estimates for the nonlinear constraints become large), SNOPT enters "nonlinear elastic" mode and solves the problem

$$\boxed{\begin{array}{ll} \text{NP}(\gamma) & \underset{x,v,w}{\text{minimize}} \quad f(x) + \gamma e^T(v+w) \\ & \text{subject to} \quad l \le \begin{pmatrix} x \\ c(x) - v + w \\ Ax \end{pmatrix} \le u, \quad v \ge 0, \quad w \ge 0, \end{array}}$$

where γ is a nonnegative penalty parameter and $f(x)+\gamma e^T(v+w)$ is called a *composite objective*. If NP has a feasible solution and γ is sufficiently large, the solutions to NP and NP(γ) are identical. If NP has no feasible solution, NP(γ) will tend to determine a "good" infeasible point if γ is again sufficiently large. (If γ were infinite, the nonlinear constraint violations would be minimized subject to the linear constraints and bounds.) A similar ℓ_1 formulation of NP is fundamental to the Sℓ_1QP algorithm of Fletcher [18]. See also Conn [12].

1.3. Other work on large-scale SQP. There has been considerable interest elsewhere in extending SQP methods to the large-scale case. Some of this work has focused on problems with nonlinear *equality* constraints. The method of Lalee, Nocedal and Plantenga [35], somewhat related to the trust-region method of Byrd and Omojokun [42], uses either the exact Lagrangian Hessian or a limited-memory quasi-Newton approximation defined by the method of Zhu *et al.* [52]. The method of Biegler, Nocedal and Schmidt [3] is in the class of *reduced-Hessian methods*, which maintain a dense approximation to the reduced Hessian, using quasi-Newton updates.

For large problems with general inequality constraints as in Problem NP, SQP methods have been proposed by Eldersveld [17], Tjoa and Biegler [50], and Betts and Frank [2]. The first two approaches are also reduced-Hessian methods. In [17], a full but structured Hessian approximation is formed from the reduced Hessian. The implementation LSSQP solves the same class of problems as SNOPT. In [50], the QP subproblems are solved by eliminating variables using the (linearized) equality constraints. The remaining variables are optimized using a dense QP algorithm. Bounds on the eliminated variables become dense constraints in the reduced QP. The method is efficient for problems whose constraints are mainly nonlinear equalities, with few bounds on the variables. In contrast, the method of Betts and Frank uses the exact Lagrangian Hessian or a finite-difference approximation, and since the QP solver works with sparse KKT factorizations (see §7), the method is not restricted to problems with few degrees of freedom.

1.4. Other large-scale methods. Two existing packages MINOS [39, 40, 41] and CONOPT [16] are designed for large problems with a modest number of degrees of freedom. MINOS uses a *projected Lagrangian* or *sequential linearly constrained* (SLC) method, whose subproblems require frequent evaluation of the problem functions. CONOPT uses a *generalized reduced gradient* (GRG) method, which maintains near-feasibility with respect to the nonlinear constraints, again at the expense of many function evaluations. SNOPT is likely to outperform MINOS and CONOPT when the functions (and their derivatives) are expensive to evaluate. Relative to MINOS, an added advantage is the existence of a merit function to ensure global convergence. This is especially important when the constraints are highly nonlinear.

LANCELOT Release A [14] is another widely used package in the area of large-scale constrained optimization. It uses a *sequential augmented Lagrangian* (SAL) method. All constraints other than simple bounds are included in an augmented Lagrangian function, which is minimized subject to the bounds. In general, LANCELOT is recommended for large problems with many degrees of freedom. It complements SNOPT and the other methods discussed above. A comparison between LANCELOT and MINOS has been made in [6, 7].

2. The SQP iteration. Here we discuss the main features of an SQP method for solving a generic nonlinear program. All features are readily specialized to the more general constraints in Problem NP.

2.1. The generic problem. In this section we take the problem to be

GNP	$\underset{x}{\text{minimize}}$	$f(x)$
	subject to	$c(x) \geq 0,$

where $x \in \mathbb{R}^n$, $c \in \mathbb{R}^m$, and the functions $f(x)$ and $c_i(x)$ have continuous second derivatives. The gradient of f is denoted by the vector $g(x)$, and the gradients of each

element of c form the rows of the Jacobian matrix $J(x)$.

We assume that a Karush-Kuhn-Tucker (KKT) point (x^*, π^*) exists for GNP, satisfying the first-order optimality conditions:

$$c(x^*) \geq 0, \qquad \pi^* \geq 0, \qquad c(x^*)^T\pi^* = 0, \qquad J(x^*)^T\pi^* = g(x^*). \tag{2.1}$$

2.2. Structure of the SQP method. An SQP method obtains search directions from a sequence of quadratic programming subproblems. Each QP subproblem minimizes a quadratic model of a certain Lagrangian function subject to linearized constraints. Some merit function is reduced along each search direction to ensure convergence from any starting point.

The basic structure of an SQP method involves *major* and *minor* iterations. The major iterations generate a sequence of iterates (x_k, π_k) that converge to (x^*, π^*). At each iterate a QP subproblem is used to generate a search direction towards the next iterate (x_{k+1}, π_{k+1}). Solving such a subproblem is itself an iterative procedure, with the *minor* iterations of an SQP method being the iterations of the QP method.

For an overview of SQP methods, see, for example, Fletcher [19], Gill, Murray and Wright [29], Murray [36], and Powell [46].

2.3. The modified Lagrangian. Let x_k and π_k be estimates of x^* and π^*. For several reasons, our SQP algorithm is based on the *modified Lagrangian* associated with GNP, namely

$$\mathcal{L}(x, x_k, \pi_k) = f(x) - \pi_k^T d_L(x, x_k), \tag{2.2}$$

which is defined in terms of the *constraint linearization* and the *departure from linearity*:

$$\begin{aligned} c_L(x, x_k) &= c(x_k) + J(x_k)(x - x_k), \\ d_L(x, x_k) &= c(x) - c_L(x, x_k); \end{aligned}$$

see Robinson [47] and Van der Hoek [51]. The first and second derivatives of the modified Lagrangian with respect to x are

$$\begin{aligned} \nabla\mathcal{L}(x, x_k, \pi_k) &= g(x) - (J(x) - J(x_k))^T\pi_k, \\ \nabla^2\mathcal{L}(x, x_k, \pi_k) &= \nabla^2 f(x) - \sum_i (\pi_k)_i \nabla^2 c_i(x). \end{aligned}$$

Observe that $\nabla^2\mathcal{L}$ is independent of x_k (and is the same as the Hessian of the conventional Lagrangian). At $x = x_k$, the modified Lagrangian has the same function and gradient values as the objective:

$$\mathcal{L}(x_k, x_k, \pi_k) = f(x_k), \qquad \nabla\mathcal{L}(x_k, x_k, \pi_k) = g(x_k).$$

2.4. The QP subproblem. Let $\mathcal{L}_Q$ be the quadratic approximation to $\mathcal{L}$ at $x = x_k$:

$$\mathcal{L}_Q(x, x_k, \pi_k) = f(x_k) + g(x_k)^T(x - x_k) + \tfrac{1}{2}(x - x_k)^T\nabla^2\mathcal{L}(x_k, x_k, \pi_k)(x - x_k).$$

If $(x_k, \pi_k) = (x^*, \pi^*)$, optimality conditions for the quadratic program

GQP*	$\underset{x}{\text{minimize}}$	$\mathcal{L}_Q(x, x_k, \pi_k)$	
	subject to	linearized constraints	$c_L(x, x_k) \geq 0$

are identical to those for the original problem GNP. This suggests that if H_k is an approximation to $\nabla^2\mathcal{L}$ at the point (x_k, π_k), an improved estimate of the solution may be found from $(\widehat{x}_k, \widehat{\pi}_k)$, the solution of the following QP subproblem:

$$\begin{array}{lll} \text{GQP}_k & \underset{x}{\text{minimize}} & f(x_k) + g(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T H_k (x - x_k) \\ & \text{subject to} & c(x_k) + J(x_k)(x - x_k) \geq 0. \end{array}$$

Optimality conditions for GQP_k may be written as

$$\begin{aligned} g(x_k) + H_k(\widehat{x}_k - x_k) &= J(x_k)^T\widehat{\pi}_k, & \widehat{\pi}_k &\geq 0, \quad \widehat{s}_k \geq 0, \\ c(x_k) + J(x_k)(\widehat{x}_k - x_k) &= \widehat{s}_k, & \widehat{\pi}_k^T\widehat{s}_k &= 0, \end{aligned}$$

where $\widehat{s}_k$ is a vector of slack variables for the linearized constraints. In this form, $(\widehat{x}_k, \widehat{\pi}_k, \widehat{s}_k)$ can be regarded as estimates of (x^*, π^*, s^*), where the nonnegative variables s^* satisfy $c(x^*) - s^* = 0$. The vector $\widehat{s}_k$ is needed explicitly for the line search (see §2.7).

2.5. The working-set matrix W_k. The *working set* is an important quantity for both the major and the minor iterations. It is the current estimate of the set of constraints that are binding at a solution. More precisely, suppose that GQP_k has just been solved. Although we try to regard the QP solver as a "black box", we normally expect it to return an independent set of constraints that are active at the QP solution. This is an optimal working set for subproblem GQP_k.

The same constraint indices define a working set for GNP (and for subproblem GQP_{k+1}). The corresponding gradients form the rows of the *working-set matrix* W_k, an $n_Y \times n$ full-rank submatrix of the Jacobian $J(x_k)$.

2.6. The null-space matrix Z_k. Let Z_k be an $n \times n_Z$ full-rank matrix that spans the null space of W_k. (Thus, $n_Z = n - n_Y$ and $W_k Z_k = 0$.) The QP solver will often return Z_k as part of some matrix factorization. For example, in NPSOL it is part of an orthogonal factorization of W_k, while in LSSQP [17] (and in the current SNOPT) it is defined from a sparse LU factorization of part of W_k. In any event, Z_k is useful for theoretical discussions, and its column dimension has strong practical implications. Important quantities are the *reduced Hessian* $Z_k^T H_k Z_k$ and the *reduced gradient* $Z_k^T g$.

2.7. The merit function. Once the QP solution $(\widehat{x}_k, \widehat{\pi}_k, \widehat{s}_k)$ has been determined, new estimates of the GNP solution are computed using a line search on the augmented Lagrangian merit function

$$\mathcal{M}(x, \pi, s) = f(x) - \pi^T\big(c(x) - s\big) + \tfrac{1}{2}\big(c(x) - s\big)^T D\big(c(x) - s\big), \tag{2.3}$$

where D is a diagonal matrix of penalty parameters. If (x_k, π_k, s_k) are the current estimates of (x^*, π^*, s^*), the line search determines a step length α_k $(0 < \alpha_k \leq 1)$ such that the new point

$$\begin{pmatrix} x_{k+1} \\ \pi_{k+1} \\ s_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \pi_k \\ s_k \end{pmatrix} + \alpha_k \begin{pmatrix} \widehat{x}_k - x_k \\ \widehat{\pi}_k - \pi_k \\ \widehat{s}_k - s_k \end{pmatrix} \tag{2.4}$$

gives a *sufficient decrease* in the merit function (2.3). Let $\varphi_k(\alpha)$ denote the merit function computed at the point $(x_k + \alpha(\widehat{x}_k - x_k), \pi_k + \alpha(\widehat{\pi}_k - \pi_k), s_k + \alpha(\widehat{s}_k - s_k))$,

i.e., $\varphi_k(\alpha)$ defines $\mathcal{M}$ as a univariate function of the step length. Initially D is zero (for $k = 0$). When necessary, the penalties in D are increased by the minimum-norm perturbation that ensures *sufficient descent* for $\varphi_k(\alpha)$ [28]. (Note: As in NPSOL, s_{k+1} in (2.4) is redefined to minimize the merit function as a function of s, prior to the solution of GQP_{k+1}. For more details, see [25, 17].)

In the line search, for some $\beta > 0$ the condition

$$c(x_k + \alpha_k p_k) \geq -\beta, \tag{2.5}$$

is enforced. (We use $\beta_i = \tau \max\{1, -c_i(x_0)\}$, where τ is a specified constant, e.g., $\tau = 10$.) This defines a region in which the objective is expected to be defined and bounded below. Murray and Prieto [38] show that under certain conditions, convergence can be assured if the line search enforces (2.5). If the objective is bounded below in $\mathbb{R}^n$ then β may be any positive vector.

If α_k is essentially zero (because $\|p_k\|$ is very large), the objective is considered "unbounded" in the expanded region. Elastic mode is entered (or continued) as described in §4.3.

2.8. The approximate Hessian. As suggested by Powell [44], we maintain a positive-definite approximate Hessian H_k. On completion of the line search, let the change in x and the gradient of the modified Lagrangian be

$$\delta_k = x_{k+1} - x_k \quad \text{and} \quad y_k = \nabla\mathcal{L}(x_{k+1}, x_k, \pi) - \nabla\mathcal{L}(x_k, x_k, \pi),$$

for some vector π. An estimate of the curvature of the modified Lagrangian along δ_k is incorporated using the BFGS quasi-Newton update,

$$H_{k+1} = H_k + \theta_k y_k y_k^T - \phi_k q_k q_k^T,$$

where $q_k = H_k\delta_k$, $\theta_k = 1/y_k^T\delta_k$ and $\phi_k = 1/q_k^T\delta_k$. When H_k is positive definite, H_{k+1} is positive definite if and only if the approximate curvature $y_k^T\delta_k$ is positive. The consequences of a negative or small value of $y_k^T\delta_k$ are discussed in the next section.

There are several choices for π, including the QP multipliers $\widehat{\pi}_{k+1}$ and least-squares multipliers λ_k (see, e.g., [22]). Here we use the updated multipliers π_{k+1} from the line search, because they are responsive to short steps in the search and they are available at no cost. The definition of $\mathcal{L}$ (2.2) yields

$$\begin{aligned} y_k &= \nabla\mathcal{L}(x_{k+1}, x_k, \pi_{k+1}) - \nabla\mathcal{L}(x_k, x_k, \pi_{k+1}) \\ &= g(x_{k+1}) - (J(x_{k+1}) - J(x_k))^T\pi_{k+1} - g(x_k). \end{aligned}$$

2.9. Maintaining positive-definiteness. Since the Hessian of the modified Lagrangian need not be positive definite at a local minimizer, the approximate curvature $y_k^T\delta_k$ can be negative or very small at points arbitrarily close to (x^*, π^*). The curvature is considered not sufficiently positive if

$$y_k^T\delta_k < \sigma_k, \qquad \sigma_k = \alpha_k(1-\eta)p_k^T H_k p_k, \tag{2.6}$$

where η is a preassigned constant ($0 < \eta < 1$) and p_k is the search direction $\widehat{x}_k - x_k$ defined by the QP subproblem. In such cases, if there are nonlinear constraints, two attempts are made to modify the update: the first modifying δ_k and y_k, the second modifying only y_k. If neither modification provides sufficiently positive approximate curvature, no update is made.

First modification. First, we define a new point z_k and evaluate the nonlinear functions there to obtain new values for δ_k and y_k:

$$\delta_k = x_{k+1} - z_k, \qquad y_k = \nabla\mathcal{L}(x_{k+1}, x_k, \pi_{k+1}) - \nabla\mathcal{L}(z_k, x_k, \pi_{k+1}).$$

We choose $z_k = x_k + \alpha_k(\bar{x}_k - x_k)$, where $\bar{x}_k$ is the first *feasible* iterate found for problem GQP_k (see §4).

The purpose of this modification is to exploit the properties of the reduced Hessian at a local minimizer of GNP. With this choice of z_k, $\delta_k = x_{k+1} - z_k = \alpha_k p_N$, where p_N is the vector $\widehat{x}_k - \bar{x}_k$. Then,

$$y_k^T\delta_k = \alpha_k y_k^T p_N \approx \alpha_k^2 p_N^T \nabla^2\mathcal{L}(x_k, x_k, \pi_k) p_N,$$

and $y_k^T\delta_k$ approximates the curvature along p_N. If W_k, the final working set of problem GQP_k, is also the working set at $\bar{x}_k$, then $W_k p_N = 0$ and it follows that $y_k^T\delta_k$ approximates the curvature for the reduced Hessian, which must be positive semi-definite at a minimizer of GNP.

The assumption that the QP working set does not change once z_k is known is always justified for problems with equality constraints (see Byrd and Nocedal [11] for a similar scheme in this context). With inequality constraints, we observe that $W_k p_N \approx 0$, particularly during later major iterations, when the working set has settled down.

This modification exploits the fact that SNOPT maintains feasibility with respect to any linear constraints in GNP. (Such a strategy allows linear constraints to be used to define a region in which f and c can be safely evaluated.) Although an additional function evaluation is required at z_k, we have observed that even when the Hessian of the Lagrangian has negative eigenvalues at a solution, the modification is rarely needed more than a few times if used in conjunction with the augmented Lagrangian modification discussed next.

Second modification. If (x_k, π_k) is not close to (x^*, π^*), the modified approximate curvature $y_k^T\delta_k$ may not be sufficiently positive and a second modification may be necessary. We choose Δy_k so that $(y_k + \Delta y_k)^T\delta_k = \sigma_k$ (if possible), and redefine y_k as $y_k + \Delta y_k$. This approach was first suggested by Powell [45], who proposed redefining y_k as a linear combination of y_k and $H_k\delta_k$.

To obtain Δy_k, we consider the *augmented* modified Lagrangian [40]:

$$\mathcal{L}_A(x, x_k, \pi_k) = f(x) - \pi_k^T d_L(x, x_k) + \tfrac{1}{2} d_L(x, x_k)^T \Omega d_L(x, x_k), \tag{2.7}$$

where Ω is a matrix of parameters to be determined: $\Omega = \mathrm{diag}(\omega_i)$, $\omega_i \geq 0$, $i = 1, \ldots, m$. The perturbation

$$\Delta y_k = (J(x_{k+1}) - J(x_k))^T \Omega d_L(x_{k+1}, x_k)$$

is equivalent to redefining the gradient difference as

$$y_k = \nabla\mathcal{L}_A(x_{k+1}, x_k, \pi_{k+1}) - \nabla\mathcal{L}_A(x_k, x_k, \pi_{k+1}). \tag{2.8}$$

We choose the smallest (minimum two-norm) ω_i's that increase $y_k^T\delta_k$ to σ_k (2.6). They are determined by the linearly constrained least-squares problem

LSP	$\underset{\omega}{\text{minimize}} \quad \|\omega\|^2$
	subject to $\; a^T\omega = \beta, \;\; \omega \geq 0,$

where $\beta = \sigma_k - y_k^T\delta_k$ and $a_i = v_i w_i$ $(i = 1, \ldots, m)$, with $v = (J(x_{k+1}) - J(x_k))\delta_k$ and $w = d_L(x_{k+1}, x_k)$. The optimal ω can be computed analytically [25, 17]. If no solution exists, or if $\|\omega\|$ is very large, no update is made.

The approach just described is related to the idea of updating an approximation of the Hessian of the augmented Lagrangian, as suggested by Han [31] and Tapia [49]. However, we emphasize that the second modification is not required in the neighborhood of a solution because as $x \to x^*$, $\nabla^2\mathcal{L}_A$ converges to $\nabla^2\mathcal{L}$ and the first modification will already have been successful.

2.10. Convergence tests. A point (x, π) is regarded as a satisfactory solution if it satisfies the first-order optimality conditions (2.1) to within certain tolerances. Let δ_P and δ_D be specified small positive constants, and define $\delta_x = \delta_P(1 + \|x\|)$, $\delta_\pi = \delta_D(1 + \|\pi\|)$. The SQP algorithm terminates if

$$c_i(x) \geq -\delta_x, \quad \pi_i \geq -\delta_\pi, \quad c_i(x)\pi_i \leq \delta_\pi, \quad |d_j| \leq \delta_\pi, \tag{2.9}$$

where $d = g(x) - J(x)^T\pi$. These conditions cannot be satisfied if GNP is infeasible, but in that case the SQP algorithm will eventually enter elastic mode and satisfy analogous tests for the problem

GNP(γ)	$\underset{x,v}{\text{minimize}}$	$f(x) + \gamma e^T v$
	subject to	$c(x) + v \geq 0, \quad v \geq 0,$

whose optimality conditions include

$$0 \leq \pi_i \leq \gamma, \quad (c_i(x) + v_i)\pi_i = 0, \quad v_i(\gamma - \pi_i) = 0.$$

The fact that $\|\pi^*\|_\infty \leq \gamma$ at a solution of GNP(γ) leads us to initiate elastic mode if $\|\pi_k\|$ exceeds γ.

3. Large-scale Hessians. In the large-scale case, we cannot treat H_k as an $n \times n$ dense matrix. Nor can we maintain dense triangular factors of a transformed Hessian $Q^T H_k Q = R^T R$ as in NPSOL. We discuss the alternatives implemented in SNOPT.

3.1. Linear variables. If only some of the variables occur nonlinearly in the objective and constraint functions, the Hessian of the Lagrangian has structure that can be exploited during the optimization. We assume that the nonlinear variables are the first $\bar{n}$ components of x. By induction, if H_0 is zero in its last $n - \bar{n}$ rows and columns, the last $n - \bar{n}$ components of the BFGS update vectors y_k and $H_k\delta_k$ are zero for all k, and every H_k has the form

$$H_k = \begin{pmatrix} \bar{H}_k & 0 \\ 0 & 0 \end{pmatrix}, \tag{3.1}$$

where $\bar{H}_k$ is $\bar{n} \times \bar{n}$. Simple modifications of the methods of §2.9 can be used to keep $\bar{H}_k$ positive definite. A QP subproblem with Hessian of this form is either unbounded, or has at least $n - \bar{n}$ constraints in the final working set. This implies that the reduced Hessian need never have dimension greater than $\bar{n}$.

In order to treat semidefinite Hessians such as (3.1), the QP solver must include an *inertia controlling* working-set strategy, which ensures that the reduced Hessian has at most one zero eigenvalue. See §4.2.

3.2. Dense Hessians. The Hessian approximations H_k are matrices of order $\bar{n}$, the number of nonlinear variables. If $\bar{n}$ is not too large, it is efficient to treat each H_k as a dense matrix and apply the BFGS updates explicitly. The storage requirement is fixed, and the number of major iterations should prove to be moderate. (We can expect 1-step Q-superlinear convergence.)

3.3. Limited-memory Hessians. To treat problems where the number of nonlinear variables $\bar{n}$ is very large, we use a limited-memory procedure to update an initial Hessian approximation H_r a limited number of times. The present implementation is quite simple and has benefits in the SQP context when the constraints are linear.

Initially, suppose $\bar{n} = n$. Let ℓ be preassigned (say $\ell = 20$), and let r and k denote two major iterations such that $r \le k \le r + \ell$. Up to ℓ updates to a positive-definite H_r are accumulated to represent the Hessian as

$$H_k = H_r + \sum_{j=r}^{k-1} \theta_j y_j y_j^T - \phi_j q_j q_j^T, \tag{3.2}$$

where $q_j = H_j \delta_j$, $\theta_j = 1/y_j^T \delta_j$ and $\phi_j = 1/q_j^T \delta_j$. The quantities $(y_j, q_j, \theta_j, \phi_j)$ are stored for each j. During major iteration k, the QP solver accesses H_k by requesting products of the form $H_k v$. These are computed with work proportional to $k - r$:

$$H_k v = H_r v + \sum_{j=r}^{k-1} \theta_j (y_j^T v) y_j - \phi_j (q_j^T v) q_j.$$

On completion of iteration $k = r + \ell$, the diagonals of H_k are computed from (3.2) and saved to form the next positive-definite H_r (with $r = k + 1$). Storage is then "reset" by discarding the previous updates. (Similar schemes are suggested by Buckley and LeNir [9, 10] and Gilbert and Lemaréchal [20].)

If $\bar{n} < n$, H_k has the form (3.1) and the same procedure is applied to $\bar{H}_k$. Note that the vectors y_j and q_j have length $\bar{n}$—a benefit when $\bar{n} \ll n$. The modified Lagrangian $\mathcal{L}_A$ (2.7) retains this property for the modified y_k in (2.8).

4. The QP solver SQOPT. Since SNOPT solves nonlinear programs of the form NP, it requires solution of QP subproblems of the same form (with $f(x)$ replaced by a quadratic function and $c(x)$ replaced by its current linearization):

$$\boxed{\begin{array}{lll} \mathrm{QP}_k & \underset{x}{\text{minimize}} & f(x_k) + g(x_k)^T(x - x_k) + \tfrac{1}{2}(x - x_k)^T H_k (x - x_k) \\ & \text{subject to} & l \le \begin{pmatrix} x \\ c(x_k) + J(x_k)(x - x_k) \\ Ax \end{pmatrix} \le u. \end{array}}$$

If a QP subproblem proves to be infeasible, we redefine QP_k (and all subsequent subproblems) to correspond to the linearization of $\mathrm{NP}(\gamma)$. SNOPT is then in elastic mode thereafter.

At present, QP_k is solved by the package SQOPT [23], which employs a two-phase active-set algorithm and implements elastic programming implicitly when necessary. SQOPT can treat any of the variable and constraint bounds as elastic, but SNOPT uses this feature only for the constraint bounds in problem FLP (§1) and for the linearized nonlinear constraint bounds in QP_k).

SQOPT maintains a dense Cholesky factorization of the QP reduced Hessian:

$$Z^T H_k Z = R^T R, \tag{4.1}$$

where Z is the null-space matrix for the working sets W in the QP minor iterations. Normally, R is computed from (4.1) when the non-elastic constraints are first satisfied. It is then updated as the QP working set changes. For efficiency the dimension of R should not be excessive (say, $n_Z \leq 1200$). This is guaranteed if the number of nonlinear variables is moderate (because $n_Z \leq \bar{n}$ at a solution).

As in MINOS, Z is maintained in "reduced-gradient" form, using the package LUSOL [26] to maintain sparse LU factors of a square matrix B that alters as the working set W changes. The important pieces are

$$W_{BS}P = (B \quad S), \qquad Z_{BS} = P \begin{pmatrix} -B^{-1}S \\ I \end{pmatrix}, \tag{4.2}$$

where W_{BS} is part of the working set (some rows and columns of $J(x_k)$ and A) and P is a permutation that ensures B is nonsingular. Variables associated with B and S are called basic and superbasic; the remainder are called nonbasic. The number of degrees of freedom is the number of superbasic variables (the column dimension of S). Products of the form Zv and $Z^T g$ are obtained by solving with B or B^T.

4.1. Condition control. If the basis matrix is not chosen carefully, the condition of a reduced-gradient Z could be arbitrarily high. To guard against this, MINOS and SQOPT implement a "basis repair" feature in the following way. LUSOL is used to compute the rectangular factorization

$$W_{BS}^T = LU,$$

returning just the permutation P that makes PLP^T unit lower triangular. The pivot tolerance is set to require $|PLP^T|_{ij} \leq 2$, and the permutation is used to define P in (4.2). It can be shown that $\|Z\|$ is likely to be little more than 1. Hence, Z should be well-conditioned *regardless of the condition of* W.

SQOPT applies this feature at the beginning of a warm start (when a potential B-S ordering is known). To prevent basis repair *every* warm start—i.e., every major iteration of SNOPT—a normal $B = LU$ factorization is computed first (with the usual loose pivot tolerance to improve the sparsity of the factors). If U appears to be more ill-conditioned than after the last repair, a new repair is invoked.

4.2. Inertia control. If NP contains linear variables, H_k in (3.1) is positive semidefinite. In SQOPT, only the last diagonal of R (4.1) is allowed to be zero. (See [27] for discussion of a similar strategy for indefinite quadratic programming.) If the initial R is singular, enough temporary constraints are added to the working set to give a nonsingular R. Thereafter, R can become singular only when a constraint is deleted from the working set (in which case no further constraints are deleted until R becomes nonsingular). When R is singular at a non-optimal point, it is used to define a direction d_Z such that

$$Z^T H_k Z d_Z = 0 \quad \text{and} \quad g^T Z d_Z < 0, \tag{4.3}$$

where $g = g(x_k) + H_k(x - x_k)$ is the gradient of the quadratic objective. The vector $d = Zd_Z$ is a direction of unbounded descent for the QP in the sense that the QP objective is linear and decreases without bound along d. Normally, a step along d reaches a new constraint, which is then added to the working set for the next iteration.

4.3. Unbounded QP subproblems. If the QP objective is unbounded along d, subproblem QP_k terminates. The final QP search direction $d = Zd_Z$ is also a direction of unbounded descent for the objective of NP. To show this, we observe from (4.3) that if we choose $p = d$, then

$$H_k p = 0 \quad \text{and} \quad g(x_k)^T p < 0.$$

The imposed nonsingularity of $\bar{H}_k$ (3.1) implies that the nonlinear components of p are zero and so the nonlinear terms of the objective and constraint functions are unaltered by steps of the form $x_k + \alpha p$. Since $g(x_k)^T p < 0$, the objective of NP is unbounded along p because it must include a term in the linear variables that decreases without bound along p.

In short, NP behaves like an unbounded LP along p, with the nonlinear variables (and functions) frozen at their current values. Thus if x_k is feasible for NP, unboundedness in QP_k implies that $F(x)$ is unbounded for feasible points, and the problem is declared unbounded.

If x_k is infeasible, unboundedness in QP_k implies that $F(x)$ is unbounded for some expanded feasible region $c(x) \geq -\beta$ (see (2.5)). We enter or continue elastic mode (with an increased value of γ if it has not already reached its maximum permitted value). Eventually, the QP subproblem will be bounded, or x_k will become feasible, or the iterations will converge to a point that approximately minimizes the one-norm of the constraint violations.

5. Implementation details. SNOPT is a general-purpose Fortran 77 package for large-scale optimization. It implements most of the techniques described in §§2–4. It includes SQOPT, a sparse QP solver that maintains a dense Cholesky factorization of the reduced QP Hessian.

5.1. The initial point. To take advantage of a good starting points x_0, we apply the QP solver to the "proximal-point" QP problem

PP	$\underset{\bar{x}}{\text{minimize}} \quad \|\bar{x} - \bar{x}_0\|^2$
	subject to the linear constraints and bounds,

where $\bar{x}$ and $\bar{x}_0$ correspond to the nonlinear variables in x. The solution defines a new starting point x_0 for the SQP iteration. The nonlinear functions are evaluated at this point and a "crash" procedure is executed to find a working set W_0 for the linearized constraints.

Note that Problem PP may be "more nonlinear" than the original problem NP, in the sense that its exact solution may lie on fewer constraints (even though it is nonlinear in the same subset of variables, $\bar{x}$). To prevent the reduced Hessian from becoming excessively large, we terminate SQOPT early by specifying a loose optimality tolerance.

5.2. Linearly constrained problems. For problems with linear constraints only, the maximum steplength is not necessarily one. Instead, it is the maximum feasible step along the search direction. If the line search is not restricted by the maximum step, the line search ensures that the approximate curvature is sufficiently positive and the BFGS update can always be applied. Otherwise, the update is skipped if the approximate curvature is not sufficiently positive.

For linear constraints, the working-set matrix W_k does not change at the new major iterate x_{k+1} and the basis B need not be refactorized. If B is constant, then

so is Z, and the only change to the reduced Hessian between major iterations comes from the rank-two BFGS update. This implies that the reduced Hessian need not be refactorized if the BFGS update is applied explicitly to the reduced Hessian. This obviates factorizing the reduced Hessian at the start of each QP, saving considerable computation.

Given *any* nonsingular matrix Q, the BFGS update to H_k implies the following update to $Q^T H_k Q$:

$$\bar{H}_Q = H_Q + \theta_k y_Q y_Q^T - \phi_k q_Q q_Q^T, \tag{5.1}$$

where $\bar{H}_Q = Q^T H_{k+1} Q$, $H_Q = Q^T H_k Q$, $y_Q = Q^T y_k$, $\delta_Q = Q^{-1}\delta_k$, $q_Q = H_Q \delta_Q$, $\theta_k = 1/y_Q^T \delta_Q$ and $\phi_k = 1/q_Q^T \delta_Q$. If Q is of the form $(\, Z \quad Y \,)$ for some matrix Y, the reduced Hessian is the leading principal submatrix of H_Q.

The Cholesky factor R of the reduced Hessian is simply the upper-left corner of the $\bar{n} \times n$ upper-trapezoidal matrix R_Q such that $H_Q = R_Q^T R_Q$. The update for R is derived from the rank-one update to R_Q implied by (5.1). Given δ_k and y_k, if we had the Cholesky factor R_Q, it could be updated directly as

$$R_Q + \frac{w}{\|w\|}\left(\sqrt{\theta_k}\, y_Q - \frac{R_Q^T w}{\|w\|}\right)^T, \tag{5.2}$$

where $w = R_Q \delta_Q$ (see Goldfarb [30], Dennis and Schnabel [15]). This rank-one modification of R_Q could be restored to upper-triangular form by applying two sequences of plane rotations from the left [21].

To simplify the notation we write (5.2) as $R_Q + uv^T$, where R_Q is an $\bar{n} \times n$ upper-trapezoidal matrix, $u = w/\|w\|$ and $v = \sqrt{\theta_k}\, y_Q - R_Q^T u$. Let v_z be the first n_z elements of v. The following algorithm determines the Cholesky factor $\bar{R}$ of the first n_z rows and columns of $\bar{H}_Q$ (5.1).

1. Compute $q = H_k \delta_k$, $t = Z^T q$.
2. Define $\phi = \|w\|_2 = (\delta_k^T H_k \delta_k)^{1/2} = (q^T \delta_k)^{1/2}$.
3. Solve $R^T w_z = t$.
4. Define $u_z = w_z/\phi$; $\sigma = (1 - \|u_z\|_2^2)^{1/2}$.
5. Define a sweep of n_z rotations P_1 in the planes $(n_z + 1, i)$, $i = n_z$, $n_z - 1$, $\ldots$, 1, such that

$$P_1 \begin{pmatrix} R & u_z \\ & \sigma \end{pmatrix} = \begin{pmatrix} \widehat{R} & 0 \\ r^T & 1 \end{pmatrix},$$

   where $\widehat{R}$ is upper triangular and r^T is a "row spike" in row $n_z + 1$.
6. Define a sweep of n_z rotations P_2 in the planes $(i, n_z + 1)$, $i = 1, 2, \ldots, n_z + 1$, such that

$$P_2 \begin{pmatrix} \widehat{R} \\ r^T + v_z^T \end{pmatrix} = \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix},$$

   where $\bar{R}$ is upper triangular.

5.3. The major iteration of SNOPT. The main steps of the SQP algorithm in SNOPT are as follows. We assume that a starting point (x_0, π_0) is available, and that the reduced-Hessian QP solver SQOPT is being used. We describe elastic mode verbally. Specific values for γ are given at the start of §6.

0. Apply the QP solver to Problem PP to find the point closest to x_0 satisfying the linear constraints. If Problem PP is infeasible, declare Problem NP infeasible. Otherwise, Problem PP defines a working-set matrix W_0. Set $k = 0$.
1. Factorize W_k.
2. Find $\bar{x}_k$, a feasible point for the QP subproblem. (This is an intermediate point for the QP solver, which also provides a working-set matrix $\bar{W}_k$ and its null-space matrix $\bar{Z}_k$.) If no feasible point exists, initiate elastic mode and restart the QP.
3. Form the reduced Hessian $\bar{Z}_k^T H_k \bar{Z}_k$ and compute its Cholesky factorization.
4. Continue solving the QP subproblem to find $(\widehat{x}_k, \widehat{\pi}_k)$, an optimal QP solution. (This provides a working-set matrix $\widehat{W}_k$ and its null-space matrix $\widehat{Z}_k$.)
   If elastic mode has not been initiated but $\|\widehat{\pi}_k\|_\infty$ is "large", enter elastic mode and restart the QP.
   If the QP is unbounded and x_k satisfies the nonlinear constraints, declare the problem unbounded. Otherwise (if the QP is unbounded), go to Step 6.
5. If (x_k, π_k) satisfies the convergence tests for NP analogous to (2.9), declare the solution optimal. If similar convergence tests are satisfied for NP(γ), go to Step 6. Otherwise, go to Step 7.
6. If elastic mode has not been initiated, enter elastic mode and repeat Step 4. Otherwise, if γ has not reached its maximum value, increase γ and repeat Step 4. Otherwise, declare the problem infeasible.
7. Find a step length α_k that gives a sufficient reduction in the merit function. Set $x_{k+1} = x_k + \alpha_k(\widehat{x}_k - x_k)$ and $\pi_{k+1} = \pi_k + \alpha_k(\widehat{\pi}_k - \pi_k)$. Evaluate the Jacobian at x_{k+1}.
8. Define $\delta_k = x_{k+1} - x_k$ and $y_k = \nabla\mathcal{L}(x_{k+1}, x_k, \pi_{k+1}) - \nabla\mathcal{L}(x_k, x_k, \pi_{k+1})$. If $y_k^T\delta_k < \sigma_k$, recompute δ_k and y_k with x_k redefined as $x_k + \alpha_k(\bar{x}_k - x_k)$. (This requires an extra evaluation of the problem derivatives.) If necessary, increase $y_k^T\delta_k$ (if possible) by adding an augmented Lagrangian term to y_k.
9. If $y_k^T\delta_k \geq \sigma_k$, apply the BFGS update to H_k using the pair $(H_k\delta_k, y_k)$.
10. Set $k \leftarrow k + 1$ and repeat from Step 1.

Apart from computing the problem functions and their first derivatives, most of the computational effort lies in Steps 1 and 3. Steps 2 and 4 may also involve significant work if the QP subproblem requires many minor iterations. Typically this will happen only during the early major iterations.

Note that all points x_k satisfy the linear constraints and bounds (as do the points used to define extra derivatives in Step 8). Thus, SNOPT evaluates the nonlinear functions only at points where it is reasonable to assume that they are defined.

6. Numerical results. We give the results of applying SNOPT to several sets of optimization problems, including 3 standard sets of small dense problems, the CUTE collection, and some large sparse problems arising in optimal trajectory calculations. Sources for the problems are given in Table 6.1. Table 6.2 defines the notation used in the later tables of results.

Unless stated otherwise, all runs were made on an SGI Indigo2 Impact 10000 with 256MB of RAM. SNOPT is coded in Fortran and was compiled using f77 in 64-bit mode and full code optimization. Figure 6.1 gives the SNOPT optional parameters and their values, most of which are the default. In some cases we compare the performance of SNOPT with the SLC code MINOS 5.5 of Dec 1996 (see [41] and §1.4). The default MINOS optional parameters were used, such as `Crash option 3` and

TABLE 6.1
Sets of test problems.

Problems	Reference
bt	Boggs and Tolle [4, 5]
hs	Hock and Schittkowski [34]
CUTE	Bongartz, Conn, Gould and Toint [8]
Spring	Murtagh and Saunders [40]
Min-time	Hargraves and Paris [32]

TABLE 6.2
Notation in tables of results.

QP	The problem is a quadratic program.
LC	The objective is nonlinear but the constraints are linear.
NC	Some of the constraints are nonlinear.
m	The number of general linear and nonlinear constraints.
n	The number of variables.
n_Z	The number of degrees of freedom at a solution (columns in Z).
Mnr	The number of QP minor iterations.
Mjr	The number of major iterations required by the optimizer.
Fcn	The number of function and gradient evaluations.
cpu	The number of cpu seconds.
Obj	The final objective value (to help classify local solutions).
Con	The final constraint violation norm (to identify infeasible problems).
a	Almost optimal (within 10^{-2} of satisfying the optimality tolerance).
l	Linear constraints infeasible.
u	Unbounded functions.
c	Final point could not be improved.
t	Iterations terminated.
i	Nonlinear constraints locally infeasible.

`Line search tolerance 0.1`. The only exceptions were `Superbasic limit 1200` and `Major iterations 2000`. The convergence criteria for SNOPT and MINOS are identical.

For the SNOPT Hessian approximations H_k, if the number of nonlinear variables is small enough ($\bar{n} \leq 75$), a full dense BFGS Hessian is used. Otherwise, a limited-memory BFGS Hessian is used, with H_k reset to the current Hessian diagonal every 20 major iterations.

To aid comparison with results given elsewhere, runs on published test problems used the associated "standard start" for x_0. In all cases, the starting multiplier estimates π_0 were set to zero. On the Hock-Schittkowski test set, setting π_0 to be the QP multipliers from the first subproblem led to fewer major iterations and function evaluations in most cases. Overall, however, SNOPT was more reliable with $\pi_0 = 0$.

The default initial γ for elastic mode is $\omega\|g(x_{k1})\|_2$, where ω is the `Elastic weight` (default 100) and x_{k1} is the iterate at which γ is first needed. Thereafter, if the rth increase to γ occurs at iteration $k2$, $\gamma = \omega 10^r \|g(x_{k2})\|_2$.

The `Major feasibility tolerance` and `Major optimality tolerance` are the parameters δ_P and δ_D of §2.10 defined with respect to Problem NP. The `Minor` tolerances are analogous quantities for SQOPT as it solves QP_k. (The `Minor feasibility tolerance` incidentally applies to the bound and linear constraints in NP as well as

QP_k.)

The `Violation limit` is the parameter τ of §2.7 that defines an expanded feasible region in which the objective is expected to be bounded below.

As in MINOS, the default is to scale the linear constraints and variables, and the first basis is essentially triangular (`Crash option 3`), except for NC problems, where SNOPT's default is the all-slack basis (`Crash option 0`).

```
BEGIN SNOPT Problem
   Minimize
   Jacobian                      sparse
   Derivative level                   3
   Elastic weight                 100.0
   Hessian updates                   20
   Hessian dimension               1200
   Superbasics limit               1200
   Iterations                   1000000
   Major iterations                1000
   Minor iterations                9000
   Major feasibility tolerance   1.0e-6
   Major optimality  tolerance   1.0e-6
   Minor feasibility tolerance   1.0e-6
   Minor optimality  tolerance   1.0e-6
   Crash option                       0  (3 for LC)
   Line search tolerance            0.9
   Step limit                       2.0
   Unbounded objective          1.0e+15
   Violation limit                 10.0
   Solution                          No
END SNOPT Problem
```

FIG. 6.1. *The SNOPT optional parameter file*

6.1. Results on small problems. SNOPT was applied to 3 sets of small test problems that have appeared in the literature.

Table 6.3 gives results on the Boggs-Tolle problems [5], which are a set of small nonlinear problems. Where appropriate, the problems were run with the "close starts" (Start 1), "intermediate starts" (Start 2) and "far starts" (Start 3) suggested by Boggs and Tolle [4]. SNOPT solved all cases except *bt4* with the intermediate starting point. In that case, the merit function became unbounded, with $\|c\| \to \infty$.

Tables 6.4–6.6 give results on the Hock-Schittkowski (HS) collection [34] as implemented in the CUTE test suite (see §6.3). Results from all CUTE HS problems are included except for *hs67*, *hs85* and *hs87*, which are not smooth. In every case SNOPT found a point that satisfies the first-order conditions for optimality. However, since SNOPT uses only first derivatives, it is able to find only first-order solutions, which may or may not be minimizers. In some cases, e.g., *hs16* and *hs25*, the final point was not a minimizer. Similarly, the constraint qualification does not hold at the final point obtained for *hs13*.

Table 6.7 summarizes the results of applying SNOPT and MINOS to the HS collection. SNOPT required more minor iterations than MINOS but fewer function evaluations. MINOS solved 119 of the 122 problems. Problem *hs104lnp* could not be

TABLE 6.3
SNOPT on the Boggs-Tolle test problems.

No.	Problem	Mnr	Mjr	Fcn	Obj
1	*bt1*	7	5	8	-9.999999E-01
2	*bt2*	13	10	14	3.256820E-02
3	*bt2* Start 2	19	16	22	3.256820E-02
4	*bt2* Start 3	32	23	41	3.256820E-02
5	*bt3* (LC)	6	3	7	4.093023E+00
6	*bt3* Start 2	6	3	7	4.093023E+00
7	*bt3* Start 3	6	3	7	4.093023E+00
8	*bt4*	22	18	42	-3.739424E+01
9	*bt4* Start 2 [t]	51	50	180	-3.168674E+02
10	*bt4* Start 3	15	11	19	-3.739424E+01
11	*bt5* (*hs63*)	26	18	41	9.617152E+02
12	*bt5* Start 2	13	8	11	9.617152E+02
13	*bt5* Start 3	14	9	13	9.617152E+02
14	*bt6* (*hs77*)	19	14	19	2.415051E-01
15	*bt6* Start 2	82	69	80	2.415051E-01
16	*bt7*	20	14	22	3.065000E+02
17	*bt8*	14	11	15	1.000001E+00
18	*bt8* Start 2	19	16	19	1.000000E+00
19	*bt9* (*hs39*)	18	14	20	-1.000000E+00
20	*bt9* Start 2	26	22	32	-1.000000E+00
21	*bt9* Start 3	29	25	36	-1.000000E+00
22	*bt10*	2	6	9	-1.000000E+00
23	*bt10* Start 2	2	13	17	-1.000000E+00
24	*bt10* Start 3	4	15	18	-1.000000E+00
25	*bt11* (*hs79*)	13	8	12	9.171343E-02
26	*bt11* Start 2	22	17	22	9.171343E-02
27	*bt11* Start 3	39	34	51	9.171343E-02
28	*bt12*	94	63	142	6.188119E+00
29	*bt12* Start 2	25	18	25	6.188119E+00
30	*bt12* Start 3	19	14	19	6.188119E+00
31	*bt13*	53	47	93	0.000000E+00

solved because of an uninitialized variable in the Standard Input Format (SIF) file (SGI Fortran does not initialize local variables to zero). The two other "failures" occurred when *hs93* and *hs98* were declared to be infeasible. Since MINOS does not yet have provision for minimizing the norm of the constraint violations once nonlinear constraint infeasibility becomes apparent, infeasibility may be declared incorrectly or at points where the constraint violations are not close to being minimized. The results of Table 6.7 and subsequent tables indicate that the treatment of infeasibility using the elastic variable approach has a substantial effect upon the reliability of the method.

6.2. Optimal control problems. Next we consider two problems that arise from the discretization of certain optimal control problems. A feature of these problems is that the optimization variables define a discretization of a function of a continuous variable (in this case, time). The accuracy of the approximation increases with the number of optimization variables and constraints.

TABLE 6.4
SNOPT on the CUTE Hock-Schittkowski suite: Part I.

No.	Problem	Mnr	Mjr	Fcn	Obj
1	*hs1* (BC)	39	24	31	3.535967E-15
2	*hs2* (BC)	28	16	25	5.042619E-02
3	*hs3* (QP)	5	3	10	1.972152E-34
4	*hs3mod* (QP)	6	4	10	1.925930E-34
5	*hs4* (BC)	2	1	3	2.666667E+00
6	*hs5* (BC)	9	6	10	-1.913223E+00
7	*hs6*	5	4	8	0.000000E+00
8	*hs7*	17	15	27	-1.732051E+00
9	*hs8* (FP)	2	5	10	0.000000E+00
10	*hs9* (LC)	5	4	10	-5.000000E-01
11	*hs10*	14	12	17	-1.000000E+00
12	*hs11*	11	9	17	-8.498464E+00
13	*hs12*	11	8	13	-3.000000E+01
14	*hs13*	1	4	11	1.434080E+00
15	*hs14*	3	5	9	1.393465E+00
16	*hs15*	2	2	5	3.065000E+02
17	*hs16*	1	4	7	2.314466E+01
18	*hs17*	18	13	26	1.000000E+00
19	*hs18*	15	13	24	5.000000E+00
20	*hs19*	8	12	24	-6.961814E+03
21	*hs20*	1	3	6	4.019873E+01
22	*hs21* (QP)	2	2	6	-9.996000E+01
23	*hs21mod* (LC)	2	2	6	-9.596000E+01
24	*hs22*	3	0	3	1.000000E+00
25	*hs23*	8	7	17	2.000000E+00
26	*hs24* (LC)	6	2	8	-1.000000E+00
27	*hs25* (BC)	0	0	3	3.283500E+01
28	*hs26*	94	62	249	5.685474E-11
29	*hs27*	24	21	32	4.000000E-02
30	*hs28* (LC)	5	3	7	4.830345E-18
31	*hs29*	26	18	25	-2.262742E+01
32	*hs30*	14	12	15	1.000000E+00
33	*hs31*	13	9	14	6.000000E+00
34	*hs32*	5	3	6	1.000000E+00
35	*hs33*	1	2	6	-3.993590E+00
36	*hs34*	4	5	8	-8.340324E-01
37	*hs35* (QP)	12	7	10	1.111111E-01
38	*hs35mod* (QP)	9	6	9	2.500000E-01
39	*hs36* (LC)	3	1	4	-3.300000E+03
40	*hs37* (LC)	8	5	9	-3.456000E+03
41	*hs38* (BC)	45	29	32	1.823912E-16
42	*hs39* (*bt9*)	20	16	28	-1.000000E+00
43	*hs40*	9	5	9	-2.500000E-01
44	*hs41* (LC)	3	0	3	1.925926E+00
45	*hs42*	11	7	12	1.385786E+01
46	*hs43*	16	9	14	-4.400000E+01
47	*hs44* (QP)	9	5	10	-1.500000E+01

TABLE 6.5
SNOPT on the CUTE Hock-Schittkowski suite: Part II.

No.	Problem	Mnr	Mjr	Fcn	Obj
48	*hs44new* (QP)	10	4	8	-1.500000E+01
49	*hs45* (BC)	8	3	11	1.000000E+00
50	*hs46*	17	12	18	3.488613E-08
51	*hs47*	45	38	52	-2.671418E-02
52	*hs48* (LC)	10	7	11	2.631260E-15
53	*hs49* (LC)	34	30	34	1.299963E-11
54	*hs50* (LC)	29	19	27	2.155810E-16
55	*hs51* (QP)	4	3	7	5.321113E-29
56	*hs52* (QP)	6	3	8	5.326648E+00
57	*hs53* (QP)	6	3	7	4.093023E+00
58	*hs54* (LC)	61	37	50	-8.674088E-01
59	*hs55* (LC)	3	0	3	6.666667E+00
60	*hs56*	27	17	28	-3.456000E+00
61	*hs57*	40	33	46	2.845965E-02
62	*hs59*	24	16	28	-6.749505E+00
63	*hs60*	11	8	12	3.256820E-02
64	*hs61*	15	10	16	-1.436461E+02
65	*hs62* (LC)	13	8	13	-2.627251E+04
66	*hs63* (*bt5*)	6	14	39	9.723171E+02
67	*hs64*	49	38	42	6.299842E+03
68	*hs65*	17	9	12	9.535289E-01
69	*hs66*	7	4	6	5.181633E-01
70	*hs68*	55	36	56	-9.204250E-01
71	*hs69*	18	13	23	-9.567129E+02
72	*hs70*	33	28	34	7.498464E-03
73	*hs71*	9	5	8	1.701402E+01
74	*hs72*	55	43	59	7.267078E+02
75	*hs73*	11	3	5	2.989438E+01
76	*hs74*	14	6	9	5.126498E+03
77	*hs75*	10	5	8	5.174413E+03
78	*hs76* (QP)	8	4	7	-4.681818E+00
79	*hs77* (*bt6*)	20	15	20	2.415051E-01
80	*hs78*	12	7	12	-2.919700E+00
81	*hs79* (*bt11*)	16	11	15	7.877682E-02
82	*hs80*	11	6	9	5.394985E-02
83	*hs81*	17	12	16	5.394985E-02
84	*hs83*	3	3	7	-3.066554E+04
85	*hs84*	15	6	18	-5.280335E+06
86	*hs86* (LC)	20	8	12	-3.234868E+01
87	*hs88*	48	36	67	1.362657E+00
88	*hs89*	51	38	70	1.362657E+00
89	*hs90*	49	38	70	1.362657E+00
90	*hs91*	47	36	62	1.362657E+00
91	*hs92*	57	42	80	1.362657E+00
92	*hs93*	27	21	25	1.350760E+02

TABLE 6.6
SNOPT on the CUTE Hock-Schittkowski suite: Part III.

No.	Problem	Mnr	Mjr	Fcn	Obj
93	*hs95*	1	1	4	1.561953E-02
94	*hs96*	1	1	4	1.561953E-02
95	*hs97*	10	13	34	3.135809E+00
96	*hs98*	10	13	34	3.135809E+00
97	*hs99*	55	20	30	-8.310799E+08
98	*hs99exp*	545	148	801	-1.260006E+12
99	*hs100*	23	13	22	6.806301E+02
100	*hs100lnp*	22	15	28	6.806301E+02
101	*hs100mod*	24	16	26	6.786796E+02
102	*hs101*	180	94	381	1.809765E+03
103	*hs102*	99	48	166	9.118806E+02
104	*hs103*	103	46	166	5.436680E+02
105	*hs104*	31	23	28	3.951163E+00
106	*hs104lnp*	28	20	26	3.951163E+00
107	*hs105* (LC)	68	47	60	1.044725E+03
108	*hs106*	31	13	16	7.049248E+03
109	*hs107*	22	6	11	5.055012E+03
110	*hs108*	31	9	13	-8.660255E-01
111	*hs109*	40	14	20	5.362069E+03
112	*hs110* (BC)	50	1	4	-9.990002E+09
113	*hs111*	221	148	259	-4.776109E+01
114	*hs111lnp*	99	57	96	-4.737066E+01
115	*hs112* (LC)	65	24	41	-4.776109E+01
116	*hs113*	33	14	19	2.430621E+01
117	*hs114*	45	16	31	-1.768807E+03
118	*hs116*	284	69	88	9.759102E+01
119	*hs117*	54	16	21	3.234868E+01
120	*hs118* (QP)	29	2	6	6.648205E+02
121	*hs119* (LC)	36	9	12	2.448997E+02
122	*hs268* (QP)	77	37	45	-1.091394E-11

TABLE 6.7
Summary: MINOS and SNOPT on the CUTE HS test set.

	MINOS	SNOPT
Problems attempted	124	124
Optimal	121	124
Cannot be improved	1	0
False infeasibility	2	0
Major iterations	7175	3573
Minor iterations	1230	2052
Function evaluations	18703	4300
Cpu time (secs)	8.45	3.66

Problem Spring. This problem computes the optimal control of a spring mass and damper system described in [40]. Our implementation of *Spring* perpetuates a coding error that makes the problem badly conditioned unless exactly 100 discretized sample points are used (see Plantenga [43]). Corrected versions appear in the CUTE test collection as problems *optcdeg2* and *optcdeg3* (see the results of §6.3). Table 6.8 includes the dimensions of six spring problems of increasing size. Table 6.9 gives results for MINOS and SNOPT on these problems. SNOPT proved remarkably effective in terms of the number of major iterations and function values—around 20 regardless of problem size.

TABLE 6.8
Dimensions of Optimal Control problems.

Problem	m	n	Problem	m	n
Spring200	400	602	*Min-time10*	270	184
Spring300	600	902	*Min-time15*	410	274
Spring400	800	1202	*Min-time20*	550	384
Spring500	1000	1502	*Min-time25*	690	454
Spring600	1200	1802	*Min-time30*	830	544
Spring700	1400	2102	*Min-time35*	970	634
			Min-time40	1110	724
			Min-time45	1250	814
			Min-time50	1390	904

TABLE 6.9
MINOS and SNOPT on Spring.

	MINOS					SNOPT				
Problem	n_z	Mnr	Mjr	Fcn	cpu	n_z	Mnr	Mjr	Fcn	cpu
Spring200	26	935	18	935	4.2	40	886	16	19	5.9
Spring300	53	2181	29	1881	13.6	48	1259	14	17	9.9
Spring400	79	1601	32	2136	16.7	47	1602	17	20	15.6
Spring500	108	2188	42	3069	27.7	50	1990	17	20	22.9
Spring600	135	3147	58	4522	65.8	61	2432	17	15	24.9
Spring700	162	3453	64	5046	81.7	62	2835	17	20	38.4

For MINOS, the major iterations increase with problem size because they are terminated at most 40 minor iterations after the subproblem is feasible (whereas SNOPT always solves its subproblems to optimality). The function evaluations are proportional to the *minor* iterations, which increase steadily. The runtime would be greatly magnified if the functions or gradients were not trivial to compute.

For more complex examples (such as the F4 minimum-time-to-climb below), the solution to the smallest problem could provide a good starting point for the larger cases. This should help most optimizers, but the expensive functions would still leave MINOS at a disadvantage compared to SNOPT.

An optimal trajectory problem. Here we give the results of applying two SQP methods to a standard optimal trajectory problem: the F4 minimum-time-to-climb. In this problem, a pilot cruising in an F-4 at Mach 0.34 at sea level wishes

TABLE 6.10
NZOPT and SNOPT on the F4 minimum-time-to-climb.

	NZOPT			SNOPT		
Problem	Mjr	Fcn	cpu	Mjr	Fcn	cpu
Min-time10	21	33	49.1	22	33	16.1
Min-time15	22	42	163.8	32	46	36.3
Min-time20	34	38	449.4	33	38	43.9
Min-time25	43	61	1368.3	33	40	67.1
Min-time30	37	41	2194.0	40	46	94.2
Min-time35	47	51	4324.5	40	46	118.6
Min-time40	47	51	6440.3	54	60	182.9
Min-time45	47	51	9348.0	49	56	209.3
Min-time50	53	57	14060.5	43	47	217.3

TABLE 6.11
MINOS and SNOPT on the F4 minimum-time-to-climb.

	MINOS					SNOPT				
Problem	n_Z	Mnr	Mjr	Fcn	cpu	n_Z	Mnr	Mjr	Fcn	cpu
Min-time10	5	657	15	1996	1083.0	5	33	22	33	16.1
Min-time15	15	1586	16	5047	4076.3	15	46	32	42	36.3
Min-time20	23	2201	14	6972	7056.4	23	38	33	38	43.9
Min-time25	30	3044	14	9947	13754.0	30	40	33	61	67.1
Min-time30	40	7180	17	23443	37198.8	40	46	40	41	94.2
Min-time35	46	4698	14	15070	28371.6	46	46	40	51	118.6
Min-time40	55	4448	14	14351	30643.3	55	60	54	51	182.9
Min-time45	64	3806	13	10752	26712.5	64	56	49	51	209.3
Min-time50	72	6758	44	20515[c]	58026.1	72	47	43	57	217.3

to ascend to 65,000 feet at Mach 1.0 in minimum time. The problem has two path constraints on the maximum altitude and the maximum dynamic pressure.

The runs in this section were made on a Sun SPARCstation 20/61 using f77 with full code optimization.

Table 6.10 gives results for 9 optimization problems, each involving a finer level of discretization of the underlying continuous problem. The problems were generated by OTIS [33], and the constraint gradients are approximated by a sparse finite-difference scheme. In the table, SNOPT is compared with the code NZOPT, which implements an SQP method based on a dense reduced Hessian and a dense orthogonal factorization of the working-set matrix. (NZOPT is a special version of NPSOL that was developed in conjunction with McDonnell Douglas Space Systems for optimal trajectory problems.) Note that both codes required only 50–60 major iterations and function evaluations to solve a problem with $O(1000)$ variables. Since the problem functions are very expensive in this application, it appears that SQP methods (even without the aid of second derivatives) are well suited to trajectory calculations.

Since SNOPT and NZOPT are based on similar SQP methods, the different iteration and function counts are due to the starting procedures and to SNOPT's limited-memory Hessian resets. Note that the limited-memory approximation did not significantly increase the counts for SNOPT.

Differences in cpu times are due to the QP solvers. In NZOPT, where the Jacobian is treated as a dense matrix, the cost of solving the QP subproblems grows as a cubic function of the number of variables, whereas the total cost of evaluating the problem functions grows quadratically. Eventually the linear algebra required to solve the QP subproblems dominates the computation. In SNOPT, sparse-matrix techniques for factoring the Jacobian greatly reduce this cost. On the larger problems, a speedup of almost two orders of magnitude has been achieved.

Table 6.11 compares MINOS and SNOPT on the F4 *Min-time* problems. The same sparse-matrix methods are used, but the times are dominated by the expensive functions and gradients.

6.3. Results on the CUTE test set. Extensive runs have been made on the CUTE test collection dated 07/May/97. The various problem types in this distribution are summarized in Table 6.12.

TABLE 6.12
CUTE problem categories

Frequency	Type	Characteristics
24	LP	Linear obj, linear constraints
159	QP	Quadratic obj, linear constraints
135	UC	Nonlinear obj, no constraints
78	BC	Nonlinear obj, bound constraints
68	LC	Nonlinear obj, linear constraints
289	NC	Nonlinear obj, nonlinear constraints
75	FP	No objective
15	NS	Nonsmooth
843		

From the complete set of 843 problems, 47 were omitted as follows:

- The nonsmooth problems (*bigbank*, *bridgend*, *britgas*, *concon*, *core1*, *core2*, *gridgena*, *hs67*, *hs85*, *hs87*, *mconcon*, *net1*, *net2*, *net3*, *stancmin*).
- 26 problems with more than 1200 degrees of freedom at the solution (*aug2d*, *aug2dc*, *aug2dcqp*, *aug2dqp*, *aug3d*, *aug3dc*, *aug3dcqp*, *aug3dqp*, *bqpgauss*, *dixmaanb*, *dtoc5*, *dtoc6*, *jimack*, *jnlbrng1*, *jnlbrng2*, *jnlbrnga*, *obstclae*, *obstclbm*, *odnamur*, *orthrdm2*, *orthrgdm*, *stcqp1*, *stcqp2*, *stnqp1*, *stnqp2*, *torsion6*).
- 4 problems with undefined variables in the SIF file (*lhaifam*, *pfit1*, *pfit3*, *recipe*).
- 1 problem with incorrect gradients (*himmelbj*).
- 1 problem with excessively low accuracy in the objective gradients (*bleachng*). Requesting greater accuracy leads to excessive evaluation time.

SNOPT was applied to the remaining 796 problems, using the same options as before (see Fig. 6.1). No special information was used in the case of LP, QP and FP problems—i.e., each problem was assumed to have a general nonlinear objective.

Detailed results are presented from 3 subsets extracted using CUTE's interactive "select" facility:

1. Linearly constrained problems (QP, BC and LC);
2. NC and FP problems with fixed dimension;
3. NC and FP problems for which the problem size can be chosen by the user.

A selection of linearly constrained problems. Tables 6.13–6.15 give results for SNOPT on the 109 CUTE LC problems. The selection for this case was

```
Objective function type         : *
Constraints type                : L  (linear constraints)
Regularity                      : R  (smooth)
Degree of available derivatives : *
Problem interest                : *
Explicit internal variables     : *
Number of variables             : *
Number of constraints           : *
```

The problem *model* has infeasible linear constraints, but was included anyway. The objective for problem *static3* is unbounded below in the feasible region.

SNOPT solved all 109 problems in the CUTE LC set, and both SNOPT and MINOS correctly diagnosed the special features of problems *model* and *static3*. MINOS solved 101 of the problems, but could not improve the final (nonoptimal) point for problems *ncvxqp1*, *ncvxqp2*, *ncvxqp4*, *ncvxqp6*, *ncvxqp8*, *powell20* and *ubh1*.

Table 6.16 summarizes the MINOS and SNOPT results on the CUTE LC problems. The total cpu time for MINOS was less than one fifth of that required for SNOPT, largely because of the five *blockqp* problems. (When these were excluded from the LC selection, the total time for SNOPT dropped from 1211.4 secs to 276.4 secs, which is comparable to the MINOS time.) On *blockqp1*, which is typical of this group of problems, MINOS requires 1021 functions evaluations compared to 9 for SNOPT. The difference in cpu time comes from the number of minor iterations (1010 for MINOS, 2450 for SNOPT) and the size of the reduced Hessians. For MINOS, the reduced Hessian dimension (the number of superbasics) is never larger than four. By contrast, for SNOPT it expands to 1005 during the first QP subproblem, only to be reduced to four during the third major iteration. The intermediate minor iterations are very expensive, with the need to update a dense matrix R (4.1) of order 1000 at each step.

Although the ability to make many changes to the working set (between function evaluations) has been regarded as an attractive feature of SQP methods, these examples illustrate that some caution is required. We anticipate that efficiency would be improved by allowing the QP subproblem to terminate early if the reduced Hessian dimension has increased significantly. (Other criteria for early termination are discussed in [37].)

A selection of problems with variable dimensions. The next selection was used to choose problems whose dimension can be one of several values. (We chose n as close to 1000 as possible. Problems from the other 3 categories were deleted.)

```
Objective function type         : *
Constraints type                : Q O (quadratic, general nonlinear)
Regularity                      : R   (smooth)
Degree of available derivatives : *
Problem interest                : M R (modelling, real application)
Explicit internal variables     : *
Number of variables             : v   (variable dimension)
Number of constraints           : v   (variable dimension)
```

TABLE 6.13
SNOPT on the CUTE LC problems: Part I.

No.	Problem	Mnr	Mjr	Fcn	Obj
1	*agg*	101	0	0	-3.599177E+07
2	*avion2*	42	22	27	9.468013E+07
3	*biggsc4*	8	2	6	-2.437500E+01
4	*blockqp1*	2450	3	9	-9.965000E+02
5	*blockqp2*	2381	2	8	-9.961012E+02
6	*blockqp3*	2164	153	215	-4.975000E+02
7	*blockqp4*	2498	9	15	-4.980982E+02
8	*blockqp5*	2557	480	708	-4.975000E+02
9	*booth*	1	0	0	0.000000E+00
10	*bt3*	6	3	7	4.093023E+00
11	*cvxqp1*	1155	18	21	1.087512E+06
12	*cvxqp2*	1426	42	48	8.201554E+05
13	*cvxqp3*	887	15	18	1.362829E+06
14	*degenlpa*	15	0	0	3.060322E+00
15	*degenlpb*	15	0	0	-3.074235E+01
16	*dtoc1l*	53	19	22	7.359454E-02
17	*dtoc3*	15	4	7	2.245904E+02
18	*eqc*	2	0	3	-8.278941E+02
19	*expfita*	45	20	24	1.136612E-03
20	*expfitb*	284	23	29	5.019366E-03
21	*expfitc*	1582	27	33	2.330257E-02
22	*extrasim*	0	0	0	0.000000E+00
23	*fccu*	38	16	19	1.114911E+01
24	*genhs28*	7	3	7	9.271737E-01
25	*goffin*	25	0	0	-1.571798E-13
26	*gouldqp2*	566	44	47	1.845311E-04
27	*gouldqp3*	469	10	20	2.027592E+00
28	*hager1*	12	2	5	8.809722E-01
29	*hager2*	15	4	8	4.325700E-01
30	*hager3*	15	4	8	1.410332E-01
31	*hager4*	18	4	8	2.833914E+00
32	*hatfldh*	4	1	4	-2.437500E+01
33	*himmelba*	0	0	0	0.000000E+00
34	*himmelbi*	265	61	67	-1.735570E+03
35	*hong*	16	7	12	2.257109E+01
36	*hubfit*	9	6	9	1.689349E-02
37	*huestis*	1145	2	5	3.482456E+10
38	*hydroell*	573	4	11	-3.585547E+06
39	*hydroelm*	292	3	10	-3.582016E+06
40	*hydroels*	101	2	9	-3.582268E+06
41	*ksip*	2746	32	35	5.757979E-01
42	*lin*	4	1	5	-1.757754E-02
43	*liswet1*	37	2	5	2.474970E-01
44	*liswet2*	23	2	5	2.529889E-01
45	*liswet3*	21	2	5	2.529889E-01

TABLE 6.14
SNOPT on the CUTE LC problems: Part II.

No.	Problem	Mnr	Mjr	Fcn	Obj
46	*liswet4*	25	1	4	2.513441E-01
47	*liswet5*	28	3	6	2.519595E-01
48	*liswet6*	21	2	5	2.540073E-01
49	*liswet7*	24	2	5	2.692336E-01
50	*liswet8*	28	2	5	2.688664E-01
51	*liswet9*	20	1	4	2.154389E+01
52	*liswet10*	53	2	5	2.507553E-01
53	*liswet11*	16	2	5	1.666122E+00
54	*liswet12*	12	1	4	2.079821E+01
55	*loadbal*	99	65	70	4.528510E-01
56	*lotschd*	6	1	4	2.398416E+03
57	*lsqfit*	8	6	9	3.378699E-02
58	*makela4*	1	0	0	0.000000E+00
59	*model*^{l}	18	0	0	0.000000E+00
60	*mosarqp1*	190	35	40	-1.542001E+02
61	*mosarqp2*	931	12	15	-5.098246E+02
62	*ncvxqp1*	1009	5	8	-7.163832E+07
63	*ncvxqp2*	1122	3	6	-5.780383E+07
64	*ncvxqp3*	1273	17	20	-3.143376E+07
65	*ncvxqp4*	1135	3	6	-9.396719E+07
66	*ncvxqp5*	1231	10	13	-6.634101E+07
67	*ncvxqp6*	1337	24	27	-3.548614E+07
68	*ncvxqp7*	799	2	5	-4.338654E+07
69	*ncvxqp8*	811	3	6	-3.049409E+07
70	*ncvxqp9*	925	15	18	-2.157328E+07
71	*odfits*	14	9	15	-2.380027E+03
72	*oet1*	101	0	0	5.382431E-01
73	*oet3*	296	0	0	4.504972E-03
74	*pentagon*	31	22	34	1.365217E-04
75	*powell20*	6	0	3	5.781250E+01
76	*pt*	1	0	0	1.783942E-01
77	*qc*	9	1	4	-9.565377E+02
78	*qcnew*	2	0	3	-8.048683E+02
79	*qpcblend*	99	5	9	-7.842543E-03
80	*qpcboei1*	1200	12	17	1.150391E+07
81	*qpcboei2*	375	10	15	8.171964E+06
82	*qpcstair*	645	8	12	6.204392E+06
83	*qpnblend*	128	15	24	-9.136140E-03
84	*qpnboei1*	1937	48	57	6.777408E+06
85	*qpnboei2*	816	24	40	1.368276E+06
86	*qpnstair*	644	12	16	5.146033E+06
87	*reading2*	2607	0	0	-1.258335E-02
88	*s268*	77	37	45	-1.091394E-11
89	*s277*-280	6	0	0	5.076190E+00
90	*simpllpa*	3	0	0	1.000000E+00

TABLE 6.15
SNOPT on the CUTE LC problems: Part III.

No.	Problem	Mnr	Mjr	Fcn	Obj
91	*simpllpb*	1	0	0	1.100000E+00
92	*sipow1*	296	0	0	-1.000000E+00
93	*sipow1m*	297	0	0	-1.000001E+00
94	*sipow2*	149	0	0	-1.000000E+00
95	*sipow2m*	149	0	0	-1.000005E+00
96	*sipow3*	59	0	0	5.346586E-01
97	*sipow4*	29	0	0	2.723613E-01
98	*sosqp1*	1	0	3	-4.864710E-16
99	*sosqp2*	2259	30	33	-4.987032E+02
100	*sseblin*	145	0	0	1.617060E+07
101	*static3*^{u}	483	78	1053	-1.000000E+15
102	*supersim*	1	0	0	6.666667E-01
103	*tame*	1	0	3	3.081488E-33
104	*tfi2*	34	0	0	6.490311E-01
105	*tfi3*	43	3	6	4.301158E+00
106	*ubh1*	1458	10	13	1.116324E+00
107	*yao*	2	0	3	2.731285E+02
108	*zangwil3*	2	0	0	0.000000E+00
109	*zecevic2*	3	2	4	-4.125000E+00

TABLE 6.16
Summary: MINOS and SNOPT on the CUTE LC problems.

	MINOS	SNOPT
Problems attempted	109	109
Optimal	100	107
Infeasible	1	1
Unbounded	1	1
Cannot be improved	7	0
Major iterations	83	1597
Minor iterations	42892	49619
Function evaluations	59976	3206
Cpu time (secs)	227.1	1239.4

Table 6.17 gives the problem dimensions and Table 6.18 gives the SNOPT results on this set. SNOPT solved 38 of the 45 problems attempted. Among the successes we have included 7 infeasible cases. These include 3 cases with infeasible linear constraints (*flosp2hh*, *flosp2hl* and *flosp2hm*), and 4 cases that SNOPT identified as having infeasible nonlinear constraints (*drcavty3*, *lubrif*, *flosp2th* and *junkturn*). Since SNOPT is not assured of finding a *global* minimizer of the sum of infeasibilities, failure to find a feasible point does not imply that none exists. To gain further assurance that *drcavty3*, *lubrif*, *flosp2th* and *junkturn* are indeed infeasible, they were re-solved using SNOPT's `Feasible Point` option, in which the true objective is ignored but "elastic mode" is invoked (as usual) if the constraint linearizations prove to be infeasible (i.e., $f(x) = 0$ and $\gamma = 1$ in problem NP(γ) of §1.1). In all 4 cases, the final sum of constraint violations was comparable to that obtained with the composite objective.

TABLE 6.17
Dimensions of variable-dimensioned CUTE NC selection.

No.	Problem	Variables	Linear	Nonlinear
1	*bdvalue*	1002	1	1000
2	*bratu2d*	1024	1	900
3	*bratu2dt*	1024	1	900
4	*bratu3d*	1000	1	512
5	*car2*	1199	1	996
6	*cbratu2d*	1058	1	882
7	*cbratu3d*	686	1	250
8	*chandheq*	100	1	100
9	*chemrcta*	1000	5	996
10	*chemrctb*	1000	3	998
11	*clnlbeam*	903	301	300
12	*drcav1lq*	1225	1	0
13	*drcav2lq*	1225	1	0
14	*drcav3lq*	1225	1	0
15	*drcavty1*	1225	1	961
16	*drcavty2*	1225	1	961
17	*drcavty3*	1225	1	961
18	*drugdis*	904	1	600
19	*drugdise*	603	1	500
20	*flosp2hh*	867	579	225
21	*flosp2hl*	867	579	225
22	*flosp2hm*	867	579	225
23	*flosp2th*	867	579	225
24	*flosp2tl*	867	579	225
25	*flosp2tm*	867	579	225
26	*hadamard*	900	1801	465
27	*junkturn*	1999	1	1400
28	*lubrif*	751	1252	249
29	*manne*	1095	366	365
30	*orbit2*	898	1	898
31	*porous1*	1024	1	900
32	*porous2*	1024	1	900
33	*reading1*	2002	1	1000
34	*reading2*	3003	1	0
35	*reading3*	2002	2	1000
36	*reading4*	1001	1	1000
37	*reading5*	1001	1	1000
38	*reading9*	2001	1	1000
39	*sreadin3*	2002	2	1000
40	*ssnlbeam*	2003	1	1000
41	*svanberg*	1000	1	1000
42	*trainf*	2008	2	1001
43	*trainh*	3008	2	1001
44	*ubh1*	909	601	0
45	*ubh5*	1010	601	100

TABLE 6.18
SNOPT on the variable-dimensioned CUTE NC problems.

No.	Problem	Mnr	Mjr	Fcn	Obj	Con
1	*bdvalue*	1180	14	30	0.000000E+00	8.3E-09
2	*bratu2d*	900	3	5	0.000000E+00	9.3E-08
3	*bratu2dt*^{i}	925	51	175	0.000000E+00	1.5E-05
4	*bratu3d*	512	4	7	0.000000E+00	1.3E-11
5	*car2*	5742	48	74	2.666122E+00	8.2E-06
6	*cbratu2d*	441	3	5	0.000000E+00	3.0E-07
7	*cbratu3d*	125	3	5	0.000000E+00	2.0E-08
8	*chandheq*	100	10	12	0.000000E+00	7.0E-07
9	*chemrcta*	1124	2	6	0.000000E+00	1.0E-07
10	*chemrctb*	1053	2	6	0.000000E+00	1.1E-08
11	*clnlbeam*	5550	10	20	3.481480E+02	2.2E-09
12	*drcav1lq*^{t}	1961	1000	1103	4.628229E-04	0.0E+00
13	*drcav2lq*^{t}	1961	1000	1128	6.985585E-04	0.0E+00
14	*drcav3lq*^{t}	1961	1000	1089	6.441539E-03	0.0E+00
15	*drcavty1*	982	12	22	0.000000E+00	6.7E-14
16	*drcavty2*	961	9	20	0.000000E+00	1.1E-09
17	*drcavty3*^{i}	1655	65	138	0.000000E+00	2.9E-01
18	*drugdis*	12489	120	242	4.267653E+00	2.0E-05
29	*drugdise*^{t}	3599	1000	1990	3.285726E+00	8.9E-01
20	*flosp2hh*^{l}	567	255	0	0.000000E+00	0.0E+00
21	*flosp2hl*^{l}	568	255	0	0.000000E+00	0.0E+00
22	*flosp2hm*^{l}	568	255	0	0.000000E+00	0.0E+00
23	*flosp2th*^{i}	2244	78	106	0.000000E+00	3.6E+01
24	*flosp2tl*	820	5	11	0.000000E+00	3.0E-10
25	*flosp2tm*	1086	24	37	0.000000E+00	6.0E-08
26	*hadamard*^{c}	315535	34	73	1.906620E-04	3.0E+01
27	*junkturn*^{i}	2724	132	223	3.623666E-04	3.2E-03
28	*lubrif* ^{i}	16323	146	271	2.914855E+01	1.8E+01
29	*manne*	742	0	3	-9.745726E-01	0.0E+00
30	*orbit2*	9855	95	110	3.123404E+02	4.4E-09
31	*porous1*	900	11	16	0.000000E+00	5.7E-07
32	*porous2*	900	6	16	0.000000E+00	4.6E-09
33	*reading1*	4205	38	54	-1.604804E-01	3.7E-12
34	*reading2*	2607	0	0	-1.258335E-02	0.0E+00
35	*reading3*	5155	42	93	-1.525715E-01	1.5E-10
36	*reading4*	4953	26	34	-2.904724E-01	1.9E-13
37	*reading5*	1000	5	10	-8.001323E-13	2.8E-09
38	*reading9*^{t}	2679	1000	2003	-3.082172E-03	1.7E-13
39	*sreadin3*	5348	49	101	-1.525715E-01	5.2E-11
40	*ssnlbeam*	10971	12	20	3.462756E+02	2.4E-11
41	*svanberg*	4866	31	55	1.671434E+03	3.3E-11
42	*trainf*	7145	25	33	3.103455E+00	1.7E-13
43	*trainh*	10801	36	49	1.231224E+01	2.0E-08
44	*ubh1*	1458	10	13	1.116324E+00	0.0E+00
45	*ubh5*	2853	33	60	1.116324E+00	1.2E-12

TABLE 6.19
Summary: SNOPT and MINOS on the variable-dimensioned CUTE NC problems.

	MINOS	SNOPT
Problems attempted	45	45
Optimal	29	31
Infeasible	6	7
Cannot be improved	1	1
False infeasibility	2	1
Terminated	4	5
False unboundedness	3	0
Major iterations	12062	6959
Minor iterations	217169	460094
Function evaluations	303718	9468
Cpu time (secs)	15421.5	12870.9

The remaining infeasible case for SNOPT was *bratu2dt*, which is listed as a "false infeasible" solution. However, the run gives a point that appears to be near-optimal, with a final nonlinear constraint violation of 1.5×10^{-5}. In this case, SNOPT's `Feasible Point` option also declared the problem infeasible with final nonlinear constraint violation of 1.5×10^{-4}. However, points satisfying the nonlinear constraints have been found in other runs (and by other algorithms).

SNOPT was unable to solve 5 problems within 1000 major iterations (*drcav1lq*, *drcav2lq*, *drcav3lq*, *reading9* and *drugdise*). On termination, SNOPT was in elastic mode for *drugdise* with final constraint violation 5.5×10^{-4} (implying that no feasible point may exist). The non-optimal final value for problem *hadamard* could not be improved.

MINOS solved 29 problems, and declared the 8 problems *drugdise*, *flosp2hh*, *flosp2hl*, *flosp2hm*, *hadamard*, *lubrif*, *orbit2* and *trainh* to be infeasible. Feasible points for *orbit2* and *trainh* are known, so these two cases are considered to have failed. Problems *bratu2dt* and *flosp2tm* and *junkturn* became unbounded at infeasible points. The non-optimal final value for *ubh1* could not be improved, and the four problems *drcavty1*, *drcavty2*, *drcavty3* and *flosp2th* could not be solved within 2000 major iterations.

Table 6.19 summarizes the performance of MINOS and SNOPT on the variable-dimensioned NC problems. As with the other test sets, the better reliability of SNOPT is partly explained by the use of elastic variables to treat infeasible problems. The large number of function evaluations is the reason why MINOS required more time than SNOPT even though fewer problems were solved. The unbounded cases for MINOS are partly attributable to the absence of a suitable merit function.

A selection of problems with fixed dimensions. The next selection was used to choose nonlinearly constrained problems whose dimension is fixed:

```
Objective function type         : *
Constraints type                : Q O (quadratic, general nonlinear)
Regularity                      : R   (smooth)
Degree of available derivatives : *
Problem interest                : M R (modelling, real application)
Explicit internal variables     : *
Number of variables             : any fixed number (fixed dimension)
Number of constraints           : any fixed number (fixed dimension)
```

Tables 6.20–6.21 give results for this set. SNOPT solved 54 of the 56 problems attempted. The successes include two problems that SNOPT identified as having infeasible nonlinear constraints (*discs* and *nystrom5*). The final sum of the nonlinear constraint violations for these problems was 4.00, 3.193×10^{-3} and 1.72×10^{-2} respectively. To our knowledge, no feasible point has ever been found for these problems. SNOPT was unable to solve problems *cresc132* and *leaknet* in 1000 major iterations. For *leaknet*, the run gives a point that appears to be close to optimality, with a final nonlinear constraint violation of 6.3×10^{-9}.

MINOS declared 7 problems to be infeasible (*cresc132*, *discs*, *lakes*, *nystrom5*, *robot*, *truspyr1* and *truspyr2*). Feasible points found by SNOPT imply that this diagnosis is correct only for *discs* and *nystrom5*. Unbounded iterations occurred in 8 cases (*brainpc3*, *brainpc7*, *brainpc9*, *errinbar*, *tenbars1*, *tenbars2*, *tenbars3* and *tenbars4*). The major iteration limit was enforced for problem *reading6*.

Table 6.22 summarizes the MINOS and SNOPT results on the fixed-dimensioned NC problems. If the conjectured infeasible problems are counted as successes, the number of successes for MINOS and SNOPT are 42 and 54 respectively, out of a total of 56.

A selection of all smooth problems. Finally, SNOPT and MINOS were compared on (almost) the entire CUTE collection. The resulting selection includes the HS, LC and NC selections considered earlier, but only the additional problems are discussed below. Table 6.23 summarizes the MINOS and SNOPT results.

SNOPT found an unbounded solution for the problem *bratu1d*. The 8 problems *eigmina*, *orthrds2*, *orthregd*, *scon1ls*, *tointgor*, *vanderm1*, *vanderm2* and *vanderm3* were terminated at a point within 10^{-2} of satisfying the convergence test. These problems would have succeeded with a less stringent convergence tolerance.

SNOPT identified 11 infeasible problems: *argauss*, *bratu2dt*, *eigenb*, *fletcher*, *growth*, *himmelbd*, *lewispol*, *lootsma*, *powellsq*, *s365mod* and *vanderm4*. Of these, *powellsq*, *vanderm4* must be counted as failures because they are known to have feasible points, as verified by calling SNOPT in `Feasible point` mode. Similarly, *fletcher* and *lootsma* have feasible solutions but their initial points are infeasible and stationary for the sum of infeasibilities, so SNOPT terminated immediately. These problems are also listed as failures. The final sums of infeasibilities for the remaining 7 problems were identical to those found by running SNOPT with the `Feasible point` option. We conjecture that these problems are infeasible.

SNOPT was unable to solve 30 cases within the allotted 1000 major iterations (problems *biggsb1*, *catena*, *catenary*, *chainwoo*, *chenhark*, *dixchlng*, *djtl*, *eigenbls*, *eigencls*, *fletcbv3*, *genrose*, *heart6ls*, *helsby*, *hydc20ls*, *maratosb*, *noncvxu2*, *noncvxun*,

TABLE 6.20
SNOPT on the fixed-dimension CUTE NC problems: Part I.

No.	Problem	Mnr	Mjr	Fcn	Obj	Con
1	*aircrfta*	5	3	7	0.000000E+00	3.7E-12
2	*aircrftb*	48	43	53	4.086853E-17	0.0E+00
3	*airport*	129	44	84	4.795270E+04	4.7E-11
4	*brainpc0*	6975	39	43	1.499639E-03	2.5E-12
5	*brainpc1*	9512	31	35	1.011544E-09	2.1E-09
6	*brainpc2*	13897	58	77	4.105827E-08	7.6E-13
7	*brainpc3*	7007	59	63	1.687131E-04	1.8E-12
8	*brainpc4*	7006	61	65	1.287866E-03	3.4E-13
9	*brainpc5*	7004	56	60	1.362251E-03	2.5E-12
10	*brainpc6*	6999	59	63	5.925666E-05	2.9E-13
11	*brainpc7*	6981	38	44	3.822638E-05	1.6E-13
12	*brainpc8*	7004	59	74	1.651779E-04	3.1E-12
13	*brainpc9*	7036	85	116	8.227963E-04	1.1E-13
14	*cantilvr*	30	25	27	1.339956E+00	1.0E-09
15	*coolhans*	57	13	29	0.000000E+00	3.7E-13
16	*cresc100*	108	42	58	5.676027E-01	1.6E-12
17	*cresc132*^{t}	2047	1000	3993	6.852576E-01	1.2E-04
18	*cresc4*	35	12	18	8.718975E-01	1.0E-11
19	*cresc50*	281	127	413	5.934401E-01	5.6E-11
20	*csfi1*	25	11	16	-4.907520E+01	1.8E-06
21	*csfi2*	49	17	23	5.501761E+01	8.0E-09
22	*dembo7*	196	41	48	1.747870E+02	2.3E-10
23	*disc2*	56	19	22	1.562500E+00	1.1E-08
24	*discs*^{i}	1004	20	48	1.200005E+01	4.0E+00
25	*dnieper*	133	15	29	1.874402E+04	5.9E-08
26	*errinbar*	171	81	95	2.804526E+01	1.2E-09
27	*grouping*	44	0	3	1.385040E+01	0.0E+00
28	*heart6*	517	358	1327	0.000000E+00	9.5E-11
29	*heart8*	55	33	54	0.000000E+00	2.4E-10
30	*himmelbk*	142	11	21	5.181434E-02	2.5E-07
31	*lakes*	410	63	151	3.505247E+05	3.0E-12
32	*launch*	40	8	15	9.004903E+00	2.3E-13
33	*lch*	733	426	447	-4.258149E+00	1.0E-13
34	*leaknet*^{t}	1173	1000	2002	1.391811E+01	3.2E-13
35	*methanb8*	49	6	14	0.000000E+00	1.2E-07
36	*methanl8*	58	8	14	0.000000E+00	4.2E-08
37	*mribasis*	902	854	4177	1.821790E+01	1.3E-04
38	*nystrom5*^{i}	852	225	612	0.000000E+00	1.7E-02
39	*prodpl0*	118	28	55	5.879010E+01	2.6E-09
40	*prodpl1*	95	8	13	3.573897E+01	2.6E-10
41	*reading6*	245	78	87	-1.446597E+02	1.2E-11
42	*reading7*	1217	10	23	-1.291618E+03	1.7E-13
43	*reading8*	2437	120	243	-2.647934E+03	4.0E-11
44	*res*	6	0	0	0.000000E+00	0.0E+00
45	*robot*	140	87	147	5.462841E+00	7.2E-12
46	*rotdisc*	1828	22	43	7.872068E+00	2.6E-04
47	*ssebnln*	330	5	12	1.617060E+07	1.1E-01
48	*swopf*	153	24	35	6.786018E-02	2.0E-11

TABLE 6.21
SNOPT on the fixed-dimension CUTE NC problems: Part II.

No.	Problem	Mnr	Mjr	Fcn	Obj	Con
49	*tenbars1*	343	96	130	2.295373E+03	4.1E-11
50	*tenbars2*	258	98	126	2.277946E+03	5.0E-12
51	*tenbars3*	463	194	372	2.247129E+03	1.1E-10
52	*tenbars4*	163	45	66	3.684932E+02	7.6E-08
53	*trigger*	7	20	49	0.000000E+00	1.6E-06
54	*truspyr1*	52	34	39	1.122874E+01	8.0E-10
55	*truspyr2*	188	167	336	1.122874E+01	1.7E-13
56	*twobars*	10	8	15	1.508652E+00	2.1E-09

TABLE 6.22
Summary: SNOPT and MINOS on the fixed-dimensioned CUTE NC problems.

	MINOS	SNOPT
Problems attempted	56	56
Optimal	40	52
Infeasible	2	2
False infeasibility	5	0
Terminated	1	2
False unboundedness	8	0
Major iterations	3193	6094
Minor iterations	53795	96823
Function evaluations	94914	16231
Cpu time (secs)	2635.1	5003.0

nlmsurf, *nonmsqrt*, *orthrgds*, *palmer5a*, *palmer5b*, *palmer5e*, *palmer7a*, *pfit2*, *pfit3ls*, *pfit4*, *qr3dls*, *snake* and *sparsine*). Another 5 problems could not be improved at a non-optimal point: *brownbs*, *hydcar20*, *meyer3*, *penalty3* and *polak4*. SNOPT essentially found the solution of the badly scaled problems *brownbs*, *meyer3* and *polak4*, but was unable to declare optimality. The unconstrained problem *penalty3* was terminated at a point where the objective gradient was 1.2×10^{-4}.

MINOS incorrectly identified 12 infeasible problems (*eigmaxa*, *eigmina*, *fletcher*, *hvycrash*, *lootsma*, *optcdeg3*, *orbit2*, *semicon1*, *vanderm1*, *vanderm2*, *vanderm3* and *vanderm4*), and was unable to solve 6 problems within 2000 major iterations (*artif*, *himmelbd*, *minc44*, *oet2*, *palmer5a* and *powellsq*). MINOS correctly found an unbounded solution for *bratu1d*, but another 20 problems were incorrectly diagnosed as being unbounded (problems *bratu2dt*, *catena*, *catenary*, *dixchlng*, *dixchlnv*, *eigenb*, *eigenc2*, *eigencco*, *elattar*, *flosp2tm*, *indef*, *oet6*, *oet7*, *orthrds2*, *orthrega*, *orthrgds*, *pfit2*, *pfit4*, *s365mod* and *semicon2*). Finally, the 7 problems *fletcbv3*, *gulf*, *heart6ls*, *nonmsqrt*, *s365*, *spanhyd*, *dittert* could not be improved at a non-optimal point.

If the LC and NC infeasible and unbounded problems are counted as successes, MINOS and SNOPT solved a grand total of respectively 719 and 740 of the 796 problems attempted. Moreover, SNOPT found a feasible point that was within a factor 10^{-2} of satisfying the optimality tolerance for another 8 cases. This is strong evidence of the robustness of SQP methods when implemented with an augmented Lagrangian merit function and elastic variable strategy for treating infeasibility.

TABLE 6.23
Summary: SNOPT and MINOS on the smooth CUTE problems.

	MINOS	SNOPT
Problems attempted	796	796
Optimal	706	721
Unbounded	2	3
Infeasible	11	16
Almost optimal	0	8
Cannot be improved	15	6
False infeasibility	21	5
Terminated	11	37
False unboundedness	30	0
Major iterations	31328	74335
Minor iterations	903395	875344
Function evaluations	1641959	135143
Cpu time (secs)	26134.6	30863.1

7. Extensions. Where possible, we have defined the SQP algorithm to be independent of the QP solver. Of course, certain "warm start" features are highly desirable. For example, SQOPT can use a given starting point and working set, and for linearly constrained problems (§5.2) it can accept a known Cholesky factor R for the reduced Hessian.

Here we discuss other "black-box" QP solvers that could be used in future implementations of SNOPT. Recall that active-set methods solve KKT systems of the form

$$\begin{pmatrix} H_k & W^T \\ W & \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} g \\ h \end{pmatrix} \tag{7.1}$$

each minor iteration, where W is the current working-set matrix. Reduced-Hessian methods such as SQOPT are efficient if W is nearly square and products $H_k x$ can be formed efficiently.

7.1. Approximate reduced Hessians. As the major iterations converge, the QP subproblems require fewer changes to their working set, and with warm starts they eventually solve in one minor iteration. Hence, the work required by SQOPT becomes dominated by the computation of the reduced Hessian $Z^T H_k Z$ and its factor R (4.1), especially if there are many degrees of freedom.

For such cases, MINOS could be useful as the QP solver because it has two ways of *approximating* the reduced Hessian in the form $Z^T H_k Z \approx R^T R$:

- R may be input from the previous major iteration and maintained using quasi-Newton updates during the QP minor iterations.
- If R is very large, it is maintained in the form

$$R = \begin{pmatrix} R_r & 0 \\ & D \end{pmatrix},$$

where R_r is a dense triangle of specified size, and D is diagonal. This structure partitions the superbasic variables into two sets. After a few minor

iterations involving all superbasics (with quasi-Newton updates to R_r and D), the variables associated with D are temporarily frozen. Iterations proceed with updates to R_r only, and superlinear convergence can be expected within that subspace. A frozen superbasic variable is then interchanged with one from R_r and the process is repeated.

Both of these features could be implemented in a future version of SQOPT. Thus, SNOPT with MINOS or an enhanced SQOPT as the QP solver would provide a viable SQP algorithm for optimization problems of arbitrary dimension. The cost per minor iteration is controllable, and the only unpredictable quantity is the total number of minor iterations.

Note that the SQP updates to H_k could be applied to R between major iterations as for the linear-constraint case (§5.2). However, the quasi-Newton updates during the first few minor iterations of each QP should achieve a similar effect.

7.2. Range-space methods. If all variables appear nonlinearly, H_k is positive definite. A "range-space" approach could then be used to solve systems (7.1) as W changes. This amounts to maintaining factors of H_k's Schur complement, $S = WH_k^{-1}W^T$. It would be efficient if W did not have many rows, so that S could be treated as a dense matrix.

7.3. Schur-complement methods. For limited-memory Hessians of the form $H_k = H_0 + VDV^T$, where H_0 is some convenient Hessian approximation, $D = \text{diag}(I, -I) = D^{-1}$, and V contains the BFGS update vectors, (7.1) is equivalent to

$$\begin{pmatrix} H_0 & W^T & V \\ W & & \\ V^T & & -D \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} g \\ h \\ 0 \end{pmatrix}.$$

Following [24, §3.6.2], if we define

$$K_0 = \begin{pmatrix} H_0 & W^T \\ W & \end{pmatrix},$$

it would be efficient to work with a sparse factorization of K_0 and dense factors of its Schur complement S. (For a given QP subproblem, V is constant, but changes to W would be handled by appropriate updates to S.)

This approach has been explored by Betts and Frank [2, §5] with $H_0 = I$ (or possibly a sparse finite-difference Hessian approximation). As part of an SQP algorithm, its practical success depends greatly on the definition of H_0 and the BFGS updates that define V. Our experience with SNOPT emphasizes the importance of ensuring positive definiteness in H_k; hence the precautions of §2.9.

If H_0 were defined as in §3, the major iterates would be identical to those currently obtained with SQOPT.

8. Summary and conclusions. We have presented theoretical and practical details about an SQP algorithm for solving nonlinear programs with large numbers of constraints and variables, where the nonlinear functions are smooth and first derivatives are available.

As with interior-point methods, the most promising way to achieve efficiency with the linear algebra is to work with sparse second derivatives (i.e., an exact Hessian of

the Lagrangian, or a sparse finite-difference approximation). However, indefinite QP subproblems raise many practical questions, and alternatives are needed when second derivatives are not available.

The present implementation, SNOPT, uses a positive definite quasi-Newton Hessian approximation H_k. If the number of nonlinear variables is moderate, H_k is stored as a dense matrix. Otherwise, limited-memory BFGS updates are employed, with resets to the current diagonal at a specified frequencey (typically every 20 major iterations). An augmented Lagrangian merit function (the same as in NPSOL) ensures convergence from arbitrary starting points.

The present QP solver, SQOPT, maintains a dense reduced-Hessian factorization $Z^T H_k Z = R^T R$, where Z is obtained from a sparse LU factorization of part of the Jacobian. Efficiency improves with the number of constraints active at a solution; i.e., the number of degrees of freedom n_z should not be excessive (say, less than 1200). This is true for many important problems, such as trajectory optimization and process control. It is likely to be true for most control problems because the number of control variables is usually small compared to the number of state variables.

The numerical results of §6 confirm that SNOPT is efficient and reliable on several sets of such problems. Relative to the dense SQP solver NPSOL, the sparse-matrix techniques have produced speedups of over 50 on the larger optimal trajectory examples, and comparable reliability. Comparisons with MINOS demonstrate greater efficiency if the function evaluations are expensive, and greater reliability in general as a result of the merit function and the "elastic variables" treatment of infeasibility.

Future work will include alternative QP solvers to allow for many degrees of freedom.

Acknowledgements. We extend sincere thanks to our colleagues Dan Young and Rocky Nelson of McDonnell Douglas Space Systems, Huntington Beach, California, for their constant support during the development of SNOPT.

REFERENCES

[1] R. H. Bartels, *A penalty linear programming method using reduced-gradient basis-exchange techniques*, Linear Algebra Appl., 29 (1980), pp. 17–32.

[2] J. T. Betts and P. D. Frank, *A sparse nonlinear optimization algorithm*, J. Optim. Theory and Applics., 82 (1994), pp. 519–541.

[3] L. T. Biegler, J. Nocedal, and C. Schmid, *A reduced Hessian method for large-scale constrained optimization*, SIAM J. Optim., 5 (1995), pp. 314–347.

[4] P. T. Boggs and J. W. Tolle, *An implementation of a quasi-Newton method for constrained optimization*, Technical Report 81-3, University of North Carolina at Chapel Hill, 1981.

[5] P. T. Boggs, J. W. Tolle, and P. Wang, *On the local convergence of quasi-Newton methods for constrained optimization*, SIAM J. Control Optim., 20 (1982), pp. 161–171.

[6] I. Bongartz, A. R. Conn, N. I. M. Gould, M. A. Saunders, and P. L. Toint, *A numerical comparison between the LANCELOT and MINOS packages for large-scale constrained optimization*, report, 1997. To appear.

[7] ———, *A numerical comparison between the LANCELOT and MINOS packages for large-scale constrained optimization: the complete numerical results*, report, 1997. To appear.

[8] I. Bongartz, A. R. Conn, N. I. M. Gould, and P. L. Toint, CUTE*: Constrained and unconstrained testing environment*, ACM Trans. Math. Software, 21 (1995), pp. 123–160.

[9] A. Buckley and A. LeNir, *QN-like variable storage conjugate gradients*, Math. Prog., 27 (1983), pp. 155–175.

[10] ———, *BBVSCG–a variable storage algorithm for function minimization*, ACM Trans. Math. Software, 11 (1985), pp. 103–119.

[11] R. H. Byrd and J. Nocedal, *An analysis of reduced Hessian methods for constrained optimization*, Math. Prog., 49 (1991), pp. 285–323.

[12] A. R. Conn, *Constrained optimization using a nondifferentiable penalty function*, SIAM J. Numer. Anal., 10 (1973), pp. 760–779.
[13] ———, *Linear programming via a nondifferentiable penalty function*, SIAM J. Numer. Anal., 13 (1976), pp. 145–154.
[14] A. R. Conn, N. I. M. Gould, and P. L. Toint, *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*, Lecture Notes in Computation Mathematics 17, Springer Verlag, Berlin, Heidelberg, New York, London, Paris and Tokyo, 1992. ISBN 3-540-55470-X 65-04.
[15] J. E. Dennis, Jr. and R. B. Schnabel, *A new derivation of symmetric positive definite secant updates*, in Nonlinear Programming 4, O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds., Academic Press, London and New York, 1981, pp. 167–199.
[16] A. Drud, *CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems*, Math. Prog., 31 (1985), pp. 153–191.
[17] S. K. Eldersveld, *Large-scale sequential quadratic programming algorithms*, PhD thesis, Department of Operations Research, Stanford University, Stanford, CA, 1991.
[18] R. Fletcher, *An ℓ_1 penalty method for nonlinear constraints*, in Numerical Optimization 1984, P. T. Boggs, R. H. Byrd, and R. B. Schnabel, eds., Philadelphia, 1985, SIAM, pp. 26–40.
[19] ———, *Practical Methods of Optimization*, John Wiley and Sons, Chichester, New York, Brisbane, Toronto and Singapore, second ed., 1987.
[20] J. C. Gilbert and C. Lemaréchal, *Some numerical experiments with variable-storage quasi-Newton algorithms*, Math. Prog., (1989), pp. 407–435.
[21] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, *Methods for modifying matrix factorizations*, Math. Comput., 28 (1974), pp. 505–535.
[22] P. E. Gill and W. Murray, *The computation of Lagrange multiplier estimates for constrained minimization*, Math. Prog., 17 (1979), pp. 32–60.
[23] P. E. Gill, W. Murray, and M. A. Saunders, *SQOPT: An algorithm for large-scale quadratic programming*. To appear.
[24] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *Sparse matrix methods in optimization*, SIAM J. on Scientific and Statistical Computing, 5 (1984), pp. 562–589.
[25] ———, *User's guide for NPSOL (Version 4.0): a Fortran package for nonlinear programming*, Report SOL 86-2, Department of Operations Research, Stanford University, Stanford, CA, 1986.
[26] ———, *Maintaining LU factors of a general sparse matrix*, Linear Algebra and its Applications, 88/89 (1987), pp. 239–270.
[27] ———, *Inertia-controlling methods for general quadratic programming*, SIAM Review, 33 (1991), pp. 1–36.
[28] ———, *Some theoretical properties of an augmented Lagrangian merit function*, in Advances in Optimization and Parallel Computing, P. M. Pardalos, ed., North Holland, North Holland, 1992, pp. 101–128.
[29] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London and New York, 1981. ISBN 0-12-283952-8.
[30] D. Goldfarb, *Factorized variable metric methods for unconstrained optimization*, Math. Comput., 30 (1976), pp. 796–811.
[31] S. P. Han, *Superlinearly convergent variable metric algorithms for general nonlinear programming problems*, Math. Prog., 11 (1976), pp. 263–282.
[32] C. R. Hargraves and S. W. Paris, *Direct trajectory optimization using nonlinear programming and collocation*, J. of Guidance, Control, and Dynamics, 10 (1987), pp. 338–348.
[33] ———, *OTIS Optimal Trajectories by Implicit Integration*, 1988. Boeing Aerospace Company, Contract No. F33615-85-c-3009.
[34] W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems 187, Springer Verlag, Berlin, Heidelberg and New York, 1981.
[35] M. Lalee, J. Nocedal, and T. Plantenga, *On the implementation of an algorithm for large-scale equality constrained optimization*. Manuscript, 1995.
[36] W. Murray, *Sequential quadratic programming methods for large-scale problems*, J. Comput. Optim. Appl., 7 (1997), pp. 127–142.
[37] W. Murray and F. J. Prieto, *A sequential quadratic programming algorithm using an incomplete solution of the subproblem*, SIAM J. Optim., 5 (1995), pp. 590–640.
[38] ———, *A second-derivative method for nonlinearly constrained optimization*. To appear, 1997.
[39] B. A. Murtagh and M. A. Saunders, *Large-scale linearly constrained optimization*, Math. Prog., 14 (1978), pp. 41–72.
[40] ———, *A projected Lagrangian algorithm and its implementation for sparse nonlinear con-*

straints, Math. Prog. Study, 16 (1982), pp. 84–117.

[41] ———, *MINOS 5.4 User's Guide*, Report SOL 83-20R, Department of Operations Research, Stanford University, Stanford, CA, Revised 1995.

[42] E. O. Omojokun, *Trust region algorithms for nonlinear equality and inequality constraints*, PhD thesis, Department of Computer Science, University of Colorado, Boulder, 1989.

[43] T. Plantenga, *A trust region method for nonlinear programming based on primal interior-point techniques*. Manuscript, 1996.

[44] M. J. D. Powell, *Algorithms for nonlinear constraints that use Lagrangian functions*, Math. Prog., 14 (1978), pp. 224–248.

[45] ———, *The convergence of variable metric methods for nonlinearly constrained optimization calculations*, in Nonlinear Programming 3, O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds., Academic Press, London and New York, 1978, pp. 27–63.

[46] ———, *Variable metric methods for constrained optimization*, in Mathematical Programming: The State of the Art, A. Bachem, M. Grötschel, and B. Korte, eds., Springer Verlag, London, Heidelberg, New York and Tokyo, 1983, pp. 288–311.

[47] S. M. Robinson, *A quadratically-convergent algorithm for general nonlinear programming problems*, Math. Prog., 3 (1972), pp. 145–156.

[48] K. Schittkowski, *NLPQL: A Fortran subroutine for solving constrained nonlinear programming problems*, Ann. Oper. Res., 11 (1985/1986), pp. 485–500.

[49] R. A. Tapia, *A stable approach to Newton's method for general mathematical programming problems in $\mathbb{R}^n$*, J. Optim. Theory and Applics., 14 (1974), pp. 453–476.

[50] I.-B. Tjoa and L. T. Biegler, *Simultaneous solution and optimization strategies for parameter estimation of differential algebraic equation systems*, Ind. Eng. Chem. Res., 30 (1991), pp. 376–385.

[51] G. Van der Hoek, *Asymptotic properties of reduction methods applying linearly equality constrained reduced problems*, Math. Prog. Study, 16 (1982), pp. 162–189.

[52] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, *L-BFGS-B—FORTRAN subroutines for large-scale bound constrained optimization*, preprint, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, December 1994.