SIAM REVIEW
Vol. 47, No. 1, pp. 99–131

SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization*

Philip E. Gill†
Walter Murray‡
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. Second derivatives are assumed to be unavailable or too expensive to calculate.

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. The Hessian of the Lagrangian is approximated using a limited-memory quasi-Newton method.

SNOPT is a particular implementation that uses a reduced-Hessian semidefinite QP solver (SQOPT) for the QP subproblems. It is designed for problems with many thousands of constraints and variables but is best suited for problems with a moderate number of degrees of freedom (say, up to 2000). Numerical results are given for most of the CUTEr and COPS test collections (about 1020 examples of all sizes up to 40000 constraints and variables, and up to 20000 degrees of freedom).

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

DOI. 10.1137/S0036144504446096

1. Introduction.

The algorithm to be described applies to constrained optimization problems of the form

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

*Published electronically February 1, 2005. This paper originally appeared in *SIAM Journal on Optimization*, Volume 12, Number 4, 2002, pages 979–1006. All sections have been expanded to cover new algorithmic features and more extensive test problems.

http://www.siam.org/journals/sirev/47-1/44609.html

†Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112 (pgill@ucsd.edu). The research of this author was supported by National Science Foundation grants DMI-9424639, CCR-9896198, DMS-9973276, and DMS-0208449.

‡Department of Management Science and Engineering, Stanford University, Stanford, CA 94305-4026 (walter@stanford.edu, saunders@stanford.edu). The research of these authors was supported by National Science Foundation grants DMI-9500668, CCR-9988205, and CCR-0306662, and Office of Naval Research grants N00014-96-1-0274 and N00014-02-1-0076.

where $f(x)$ 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 vectors of lower and upper bounds. We assume that the nonlinear functions are smooth and that their first derivatives are available (and possibly expensive to evaluate).

The idea of sequential quadratic programming (SQP) methods is to solve the nonlinearly constrained problem using a sequence of *quadratic programming* (QP) subproblems. The constraints of each QP subproblem are linearizations of the constraints in the original problem, and the objective function of the subproblem is a quadratic approximation to the Lagrangian function.

SNOPT (Sparse Nonlinear OPTimizer) [55] is the implementation of a particular SQP algorithm that exploits sparsity in the constraint Jacobian and maintains a limited-memory quasi-Newton approximation H_k to the Hessian of the Lagrangian. A new method is used to update H_k in the presence of negative curvature. The QP subproblems are solved using an inertia-controlling reduced-Hessian active-set method (SQOPT) that allows for variables appearing linearly in the objective and constraint functions. (The limited-memory Hessian is then semidefinite.) Other features include the treatment of infeasible nonlinear constraints using elastic programming, use of a well-conditioned nonorthogonal basis for the null-space of the QP working set (assisted by sparse rank-revealing LU factors), early termination of the QP subproblems, and finite-difference estimates of missing gradients.

The method used by the QP solver SQOPT is based on solving a sequence of linear systems involving the reduced Hessian $Z^T H_k Z$, where Z is defined implicitly using the sparse LU factorization. Reduced-Hessian methods are best suited to problems with few degrees of freedom, i.e., problems for which many constraints are active. However, the implementation allows for problems with an arbitrary number of degrees of freedom.

1.1. Infeasible Constraints. SNOPT deals with infeasibility using ℓ_1 penalty functions. First, infeasible linear constraints are detected by solving a problem of the form

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

where e is a vector of ones and v and w are handled implicitly. This is equivalent to minimizing the one-norm of the general linear constraint violations subject to the simple bounds—often called *elastic programming* in the linear programming literature [15]. Elastic programming has long been a feature of the XS system of Brown and Graves [16]. Other algorithms based on minimizing one-norms of infeasibilities are given by Conn [26] and Bartels [4].

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. (Sometimes this feature helps ensure that the functions and gradients are well defined; see section 6.2.)

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

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

where $f(x) + \gamma e^T(v+w)$ is called a *composite objective*, and the penalty parameter γ ($\gamma \ge 0$) may take a finite sequence of increasing values. 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 used in the SQP method of Tone [98] and is fundamental to the Sℓ_1QP algorithm of Fletcher [38]. See also Conn [25] and Spellucci [94]. An attractive feature is that only linear terms are added to (NP), giving no increase in the expected degrees of freedom at each QP solution.

1.2. The SQP Approach. An SQP method was first suggested by Wilson [102] for the special case of convex optimization. The approach was popularized mainly by Biggs [7], Han [66], and Powell [85, 87] for general nonlinear constraints. Further history of SQP methods and extensive bibliographies are given in [61, 39, 73, 78, 28]. For a survey of recent results, see Gould and Toint [65].

Several general-purpose SQP solvers have proved reliable and efficient during the last 20 years. For example, under mild conditions the solvers NLPQL [92], NPSOL [57, 60], and DONLP [95] typically find a (local) optimum from an arbitrary starting point, and they require relatively few evaluations of the problem functions and gradients compared to traditional solvers such as MINOS [75, 76, 77] and CONOPT [34, 2].

SQP methods have been particularly successful in solving the optimization problems arising in optimal trajectory calculations. For many years, the optimal trajectory system OTIS (Hargraves and Paris [67]) has been applied successfully within the aerospace industry, using NPSOL to solve the associated optimization problems. NPSOL is a transformed Hessian method that treats the Jacobian of the general constraints as a dense matrix and updates an explicit quasi-Newton approximation to $Q_k^T H_k Q_k$, the transformed Hessian of the Lagrangian, where Q_k is orthogonal. The QP subproblem is solved using a linearly constrained linear least-squares method that exploits the properties of the transformed Hessian.

Although NPSOL has solved OTIS examples with as many as two thousand constraints and over a thousand variables, the need to handle increasingly large models has provided strong motivation for the development of new sparse SQP algorithms. Our aim is to describe a new SQP method that has the favorable theoretical properties of the NPSOL algorithm but is suitable for a broad class of large problems, including those arising in trajectory optimization.

There has been considerable interest in extending SQP methods to the large-scale case (sometimes using exact second derivatives). Some of this work has focused on problems with nonlinear *equality* constraints. The method of Lalee, Nocedal, and Plantenga [69], related to the trust-region method of Byrd [19] and Omojokun [79], uses either the exact Lagrangian Hessian or a limited-memory quasi-Newton approximation defined by the method of Zhu et al. [103]. The method of Biegler,

Nocedal, and Schmid [6] 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 [36], Tjoa and Biegler [97], Fletcher and Leyffer [40, 41], and Betts and Frank [5]. The first three approaches are also reduced-Hessian methods. Eldersveld forms a full Hessian approximation from the reduced Hessian, and his implementation LSSQP solves the same class of problems as SNOPT. In Tjoa and Biegler's method, the QP subproblems are solved by eliminating variables using the (linearized) equality constraints, and the remaining variables are optimized using a dense QP solver. As bounds on the eliminated variables become dense constraints in the reduced QP, the method is best suited to problems with many nonlinear equality constraints but few bounds on the variables. The filter-SQP method of Fletcher and Leyffer uses a reduced-Hessian QP solver in conjunction with an exact Lagrangian Hessian. This method is also best suited for problems with few degrees of freedom. In contrast, the method of Betts and Frank employs an exact or finite-difference Lagrangian Hessian and a QP solver based on sparse KKT factorizations (see section 8). It is therefore applicable to problems with many degrees of freedom.

Several large-scale methods solve the QP subproblems by an interior method. They typically require an exact or finite-difference Lagrangian Hessian but can accommodate many degrees of freedom. Examples are Boggs, Kearsley, and Tolle [8, 9] and Sargent and Ding [91].

1.3. Other Large-Scale Methods. MINOS and versions 1 and 2 of CONOPT are reduced-Hessian methods for general large-scale optimization. Like SNOPT, they use first derivatives and were originally designed for large problems with few degrees of freedom (up to 2000, say). For nonlinear constraints, MINOS uses a *linearly constrained Lagrangian* method, whose subproblems require frequent evaluation of the problem functions. CONOPT uses a *generalized reduced gradient* method, which has the advantage of maintaining near-feasibility with respect to the nonlinear constraints, but 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 SNOPT's rigorous control of the augmented Lagrangian merit function to ensure global convergence, and explicit provision for infeasible subproblems. This is especially important when the constraints are highly nonlinear.

A stabilized form of MINOS named KNOSSOS has recently been developed [47] (it makes use of MINOS or SNOPT as subproblem solvers), and CONOPT version 3 [2] is now able to use second derivatives.

LANCELOT [27, 64] is another widely used package for large-scale constrained optimization. It uses a *bound constrained augmented Lagrangian* method, is effective with either first or second derivatives, and is suitable 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 [12, 13].

LOQO [100], KNITRO [21, 20], and IPOPT [101] are examples of large-scale optimization packages that treat inequality constraints by a *primal-dual interior method*. They require second derivatives but can accommodate many degrees of freedom.

All of the solvers mentioned are well suited to algebraic modeling environments such as GAMS [48] and AMPL [1] because the functions and derivatives are then available cheaply and to high precision.

1.4. Notation. Some important quantities follow:

(x, π, s)	primal, dual, and slack variables for problem (GNP) (section 2.1),
(x^*, π^*, s^*)	optimal variables for problem (GNP),
(x_k, π_k, s_k)	the kth estimate of (x^*, π^*, s^*),
f_k, g_k, c_k, J_k	functions and gradients evaluated at x_k,
$(\widehat{x}_k, \widehat{\pi}_k, \widehat{s}_k)$	optimal variables for QP subproblem (GQP$_k$) (section 2.4).

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

$$\text{(GNP)} \qquad \begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & c(x) \geq 0, \end{array}$$

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 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 QP 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.

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*:

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

see Robinson [90] and Van der Hoek [99]. The first and second derivatives of the modified Lagrangian with respect to x are

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

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: $\mathcal{L}(x_k, x_k, \pi_k) = f_k,\ \nabla\mathcal{L}(x_k, x_k, \pi_k) = g_k$.

2.4. The QP Subproblem. Let the quadratic approximation to $\mathcal{L}$ at x_k be

$$\mathcal{L}_q(x, x_k, \pi_k) = f_k + g_k^T(x - x_k) + \tfrac{1}{2}(x - x_k)^T\nabla^2\mathcal{L}(x_k, x_k, \pi_k)(x - x_k).$$

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

$$\begin{array}{lll} (\mathrm{GQP}^*) & \underset{x}{\text{minimize}} & \mathcal{L}_q(x, x_k, \pi_k) \\ & \text{subject to} & \text{linearized constraints} \quad c_L(x, x_k) \geq 0 \end{array}$$

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:

$$\begin{array}{lll} (\mathrm{GQP}_k) & \underset{x}{\text{minimize}} & f_k + g_k^T(x - x_k) + \tfrac{1}{2}(x - x_k)^T H_k(x - x_k) \\ & \text{subject to} & c_k + J_k(x - x_k) \geq 0. \end{array}$$

Optimality conditions for (GQP_k) may be written as

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

where $\widehat{s}_k$ is a vector of slack variables for the linearized constraints. In this form, $(\widehat{x}_k, \widehat{\pi}_k, \widehat{s}_k)$ may be regarded as estimates of (x^*, π^*, s^*), where the slack variables s^* satisfy $c(x^*) - s^* = 0$, $s^* \geq 0$. The vector $\widehat{s}_k$ is needed explicitly for the line search (section 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 expect it to return an independent set of constraints that are active at the QP solution (even if the QP constraints are degenerate). 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_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 [36] (and in the current SNOPT) it is defined implicitly 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* $Z_k^T g_k$.

2.7. The Merit Function and Line Search. 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 from the current solution (x_k, π_k, s_k) toward the QP solution. The line search must achieve a *sufficient decrease* in the augmented Lagrangian merit function

$$\mathcal{M}_\rho(x,\pi,s) = f(x) - \pi^T\big(c(x)-s\big) + \tfrac{1}{2}\sum_{i=1}^{m}\rho_i\big(c_i(x)-s_i\big)^2, \tag{2.3}$$

where ρ is a vector of penalty parameters. For step lengths $\alpha \in (0,1]$, let $v(\alpha)$ be points along the line, and let $\varphi_\rho(\alpha)$ denote $\mathcal{M}_\rho$ as a univariate function of α:

$$v(\alpha) = \begin{pmatrix} x_k \\ \pi_k \\ s_k \end{pmatrix} + \alpha \begin{pmatrix} \widehat{x}_k - x_k \\ \widehat{\pi}_k - \pi_k \\ \widehat{s}_k - s_k \end{pmatrix}, \qquad \varphi_\rho(\alpha) = \mathcal{M}_\rho(v(\alpha)).$$

Also let $\varphi'_\rho(0)$ denote the directional derivative of the merit function at the base point $\alpha = 0$ for a given vector ρ.

The default initial value for the penalty parameters is $\rho = 0$ (for $k = 0$). Before each line search, some elements of ρ may need to be changed to ensure that the directional derivative $\varphi'_\rho(0)$ is sufficiently negative [60]. First we find the vector ρ^* that solves the linearly constrained least-squares problem

$$\boxed{\begin{array}{lll} (\mathrm{LSP}_\rho) & \underset{\rho}{\text{minimize}} & \|\rho\|_2^2 \\ & \text{subject to} & \varphi'_\rho(0) = -\tfrac{1}{2}p_k^T H_k p_k, \quad \rho \geq 0, \end{array}}$$

where $p_k \equiv \widehat{x}_k - x_k$. The solution of (LSP_ρ) can be obtained analytically, and it can be shown that $\varphi'_\rho(0) \leq -\frac{1}{2}p_k^T H_k p_k$ for any $\rho \geq \rho^*$ [36, 57, 60].

It is important to allow the penalty parameters to decrease during the early major iterations. However, to guarantee convergence, this must be done in such a way that the penalty parameters cannot oscillate indefinitely. The reduction scheme involves a damping factor $\Delta_\rho \geq 1$. Let ρ denote the vector of penalty parameters at the start of iteration k. The idea is to define the new parameter $\bar{\rho}_i$ as the geometric mean of ρ_i and a damped value of ρ_i^* as long as this mean is sufficiently positive and not too close to ρ_i:

$$\bar{\rho}_i = \max\{\rho_i^*, \hat{\rho}_i\}, \quad \text{where} \quad \hat{\rho}_i = \begin{cases} \rho_i & \text{if } \rho_i < 4(\rho_i^* + \Delta_\rho), \\ \big(\rho_i(\rho_i^* + \Delta_\rho)\big)^{1/2} & \text{otherwise.} \end{cases} \tag{2.4}$$

Initially $\Delta_\rho = 1$ (for $k = 0$). Thereafter it is increased by a factor of two whenever $\|\rho\|_2$ increases after a consecutive sequence of iterations in which the penalty norm decreased, or, alternatively, $\|\rho\|_2$ decreases after a consecutive sequence of iterations in which the penalty norm increased. This choice of Δ_ρ ensures that the penalty parameters can oscillate only a finite number of times.

With $\rho \leftarrow \bar{\rho}$ in the merit function (2.3), a safeguarded line search is used to find a step length α_{k+1} that reduces $\mathcal{M}_\rho$ to give the next solution estimate $v(\alpha_{k+1}) = (x_{k+1}, \pi_{k+1}, s_{k+1})$. As in NPSOL, s_{k+1} is then redefined to minimize the merit function as a function of s prior to the solution of (GQP_{k+1}) [57, 36].

2.8. Bounding the Constraint Violation. In the line search, the following condition is enforced for some vector $b > 0$:

$$c(x_k + \alpha_k p_k) \geq -b \qquad (p_k \equiv \widehat{x}_k - x_k). \tag{2.5}$$

We use $b_i = \tau_V \max\{1, -c_i(x_0)\}$, where τ_V is a specified constant, e.g., $\tau_V = 10$. This defines a region in which the objective is expected to be defined and bounded below. (A similar condition is used in [93].) Murray and Prieto [74] 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 b may be any large 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 section 4.7.

2.9. The Approximate Hessian. As suggested by Powell [86], 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 $\widehat{\pi}_k$ and least-squares multipliers λ_k (see, e.g., [52]). Here we use the updated multipliers π_{k+1} from the line search, because they are responsive to short steps in the search and are available at no cost. The definition of $\mathcal{L}$ from (2.2) yields

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

2.10. 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. The purpose of this modification is to exploit the properties of the reduced Hessian at a local minimizer of (GNP). 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 by recording $\bar{x}_k$, the first *feasible* iterate found for problem (GQP$_k$) (see section 4). The search direction may be regarded as

$$p_k = (\bar{x}_k - x_k) + (\widehat{x}_k - \bar{x}_k) \equiv p_R + p_N.$$

We set $z_k = x_k + \alpha_k p_R$, giving $\delta_k = \alpha_k p_N$ and

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

so that $y_k^T \delta_k$ 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 semidefinite 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 [23] 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). 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 suggested by Powell [87], 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 [76]:

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

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

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

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 (see (2.6)). They are determined by the linearly constrained least-squares problem

$$\boxed{\begin{array}{ll} (\mathrm{LSP}_\omega) & \underset{\omega}{\text{minimize}} \quad \|\omega\|_2^2 \\ & \text{subject to} \quad a^T \omega = \beta, \quad \omega \geq 0, \end{array}}$$

where $\beta = \sigma_k - y_k^T \delta_k$ and $a_i = v_i w_i$ $(i = 1:m)$, with $v = (J_{k+1} - J_k)\delta_k$ and $w = d_L(x_{k+1}, x_k)$. 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 [66] and Tapia [96]. 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.

2.11. 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 τ_P and τ_D be specified small positive constants, and define $\tau_x = \tau_P(1 + \|x\|_\infty)$, $\tau_\pi = \tau_D(1 + \|\pi\|_\infty)$. The SQP algorithm terminates if

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

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 a series of problems

$$\boxed{\begin{array}{ll} (\mathrm{GNP}(\gamma)) & \underset{x,v}{\text{minimize}} \quad f(x) + \gamma e^T v \\ & \text{subject to} \quad c(x) + v \geq 0, \quad v \geq 0, \end{array}}$$

with γ taking an increasing set of values $\{\gamma_\ell\}$ up to some maximum. The optimality conditions for (GNP(γ)) 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 $\|\pi_k\|_\infty$ exceeds some value γ_1 (or if (GQP$_k$) is infeasible). We use

$$\gamma_1 \equiv \gamma_0\|g(x_{k1})\|_\infty, \qquad \gamma_\ell = 10^{\ell(\ell-1)/2}\gamma_1 \quad (\ell = 2, 3, \dots), \tag{2.10}$$

where γ_0 is a parameter (10^4 in our numerical results) and x_{k1} is the iterate at which γ is first needed.

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 section 2.10 can be used to keep $\bar{H}_k$ positive definite. A QP subproblem with a 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 $\bar{n}$.

Under the assumption that the objective function is bounded below in some expanded feasible region $c(x) \geq -b$ (see (2.5)), a sequence of positive-definite matrices $\bar{H}_k$ with uniformly bounded condition numbers is sufficient for the SQP convergence theory to hold. (This case is analogous to converting inequality constraints to equalities by adding slack variables—the Hessian is singular only in the space of the slack variables.) However, 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 sections 4.6–4.7.

3.2. Dense Hessians. The Hessian approximations $\bar{H}_k$ are matrices of order $\bar{n}$, the number of nonlinear variables. If $\bar{n}$ is not too large, it is efficient to treat each $\bar{H}_k$ 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 one-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 an advantage in the SQP context when the constraints are linear: the reduced Hessian for the QP subproblem can be updated between major iterations (see section 6.4).

Initially, suppose $\bar{n} = n$. Let ℓ be preassigned (say $\ell = 10$), and let r and k denote two major iterations such that $r \le k \le r + \ell$. At iteration k the BFGS approximate Hessian may be expressed in terms of ℓ updates to a positive-definite H_r:

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

where $q_j = H_j \delta_j$, $\theta_j = 1/y_j^T \delta_j$, and $\phi_j = 1/q_j^T \delta_j$. It is better numerically to write H_k in the form $H_k = G_k^T G_k$, where G_k is the product of elementary matrices

$$G_k = H_r^{1/2}(I + \delta_r v_r^T)(I + \delta_{r+1} v_{r+1}^T) \cdots (I + \delta_{k-1} v_{k-1}^T), \tag{3.3}$$

with $v_j = \pm(\theta_j \phi_j)^{1/2} y_j - \phi_j q_j$ [14]. (The sign may be chosen to minimize the rounding error in computing v_j.) The quantities (δ_j, v_j) are stored for each j. During major iteration k, the QP solver accesses H_k by requesting products of the form $H_k u$. These are computed with work proportional to $k - r$ using the recurrence relations:

$$\begin{aligned} &u \leftarrow u + (v_j^T u)\delta_j, \quad j = k-1 : r; && u \leftarrow H_r^{1/2} u; \\ &w \leftarrow H_r^{1/2} u; && w \leftarrow w + (\delta_j^T w) v_j, \quad j = r : k-1. \end{aligned}$$

Note that products of the form $u^T H u$ are easily and safely computed as $\|z\|_2^2$ with $z = G_k u$.

A separate calculation is used to update the *diagonals* of H_k from (3.2). On completion of iteration $k = r + \ell$, these diagonals form the next positive-definite H_r (with $r = k+1$). Storage is then "reset" by discarding the previous updates. (Similar schemes are described by Buckley and LeNir [17, 18] and Gilbert and Lemaréchal [49]. More elaborate schemes are given by Liu and Nocedal [70], Byrd, Nocedal, and Schnabel [24], and Gill and Leonard [51], and some have been evaluated by Morales [71]. However, as already indicated, these schemes would require refactorization of the reduced Hessian in the linearly constrained case.)

An alternative approach is to store the quantities $(y_j, q_j, \theta_j, \phi_j)$ for each j and compute the product $H_k u$ as

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

This form requires the same amount of work to compute the product, and may be appropriate for certain types of QP solver (see section 8.3).

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 δ_j and v_j (and y_j and q_j) have length $\bar{n}$—a benefit when $\bar{n} \ll n$. The modified Lagrangian $\mathcal{L}_A$ from (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 convex quadratic function, and $c(x)$ replaced by its current linearization:

$$(\mathrm{QP}_k) \qquad \begin{array}{ll} \underset{x}{\text{minimize}} & f_k + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T H_k (x - x_k) \\ \text{subject to} & l \le \begin{pmatrix} x \\ c_k + J_k(x - x_k) \\ Ax \end{pmatrix} \le u. \end{array}$$

At present, (QP_k) is solved by the package SQOPT [53], which employs a two-phase active-set algorithm and implements elastic programming implicitly when necessary. The Hessian H_k may be positive semidefinite and is defined by a routine for forming products $H_k v$.

4.1. Elastic Bounds. SQOPT can treat any of the bounds in (QP_k) as elastic. Let x_j refer to the jth variable or slack. For each j, an input array specifies which of the bounds l_j, u_j is elastic (either, neither, or both). A parallel array maintains the current state of each x_j. If the variable or slack is currently outside its bounds by more than the `Minor feasibility tolerance`, it is given a linear penalty term $\gamma \times$ *infeasibility* in the objective function. This is a much-simplified but useful form of piecewise linear programming (Fourer [42, 43, 44]).

SNOPT uses elastic bounds in three different ways:

- to solve problem (FLP) (section 1.1) if the linear constraints are infeasible;
- to solve problem (PP1) (section 6.1);
- to solve the QP subproblems associated with problem (NP(γ)) after nonlinear elastic mode is initiated.

4.2. QP Search Directions. At each minor iteration, active-set methods for solving (QP_k) should obtain a search direction d satisfying the so-called KKT system

$$\begin{pmatrix} H_k & W^T \\ W & \end{pmatrix} \begin{pmatrix} d \\ y \end{pmatrix} = - \begin{pmatrix} g_q \\ 0 \end{pmatrix}, \tag{4.1}$$

where W is the current working-set matrix and g_q is the QP objective gradient. SQOPT implements several *null-space methods*, as described in the next three sections.

4.3. The Null-Space Approach. One way to obtain d in (4.1) is to solve the reduced-Hessian system

$$Z^T H_k Z d_z = -Z^T g_q, \qquad d = Z d_z, \tag{4.2}$$

where Z is a null-space matrix for W. SQOPT maintains Z in "reduced-gradient" form as in MINOS, using sparse LU factors of a square matrix B whose columns change as the working set W changes:

$$W = \begin{pmatrix} B & S & N \\ & & I \end{pmatrix} P, \qquad Z = P^T \begin{pmatrix} -B^{-1}S \\ I \\ 0 \end{pmatrix}, \tag{4.3}$$

where P is a permutation such that 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 n_z (the column dimension of S and Z). Products of the form Zv and Z^Tg are obtained by solving with B or B^T.

If n_z is small enough, SQOPT uses a dense Cholesky factorization:

$$Z^THZ = R^TR. \tag{4.4}$$

Normally, R is computed from (4.4) when the nonelastic 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, $n_z \leq 2000$). This is guaranteed if the number of nonlinear variables is moderate (because $n_z \leq \bar{n}$ at a solution), but it is often true even if $\bar{n} = n$.

4.4. 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^THZ and its factor R from (4.4), especially if there are many degrees of freedom.

For this reason, SQOPT can optionally maintain a quasi-Newton approximation $Z^THZ \approx R^TR$ as in MINOS [75]. It also allows R to be input from a previous problem of the same dimensions (a "hot start" feature of special benefit in the SQP context).

Note that the SQP updates to H_k could be applied to R between major iterations as for the linear-constraint case (section 6.4). However, the quasi-Newton updates during the first few minor iterations of each QP should achieve a similar effect.

4.5. CG Methods. By construction, the QP Hessians H_k are positive definite or positive semidefinite. Hence, the conjugate-gradient (CG) method is a natural tool for very large systems. SQOPT includes a CG option for finding approximate solutions to

$$(Z^THZ + \delta^2 I)d_z = -Z^Tg_q, \tag{4.5}$$

where $\delta \approx 10^{-3}$ is a small regularization parameter to allow for singular Z^THZ. When Z has many columns, the main concern is that many CG iterations may be needed to obtain a useful approximation to d_z. Normally CG methods require some sort of problem-dependent preconditioner, but unexpectedly good results have been obtained on many large problems *without preconditioning*. This can be largely explained by the diagonal-plus-low-rank structure of both H_k and Z^TZ (with both diagonal parts being close to I, especially if there are few general constraints).

For problems with many degrees of freedom, SQOPT optionally maintains a reduced-Hessian approximation in the form

$$R = \begin{pmatrix} R_r & 0 \\ & D \end{pmatrix}, \tag{4.6}$$

where R_r is a dense triangle of specified size and D is a positive diagonal. This structure partitions the superbasic variables into two sets and allows the cost per minor iteration to be controlled. The only unpredictable quantity is the total number of minor iterations.

In MINOS, this structure is used to generate search directions directly. 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.

In SQOPT, our aim has been to use R in (4.6) as a preconditioner for (4.5) [61, pp. 151–153], [72]. To date, this has produced mixed results, but fortunately the CG option without preconditioning has performed well, as already mentioned.

4.6. Inertia Control. If (NP) contains linear variables, H_k in (3.1) is positive semidefinite. In SQOPT, only the last diagonal of R in (4.4) is allowed to be zero. (See [59] for discussion of a similar strategy for indefinite QP.) 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 nonoptimal point, it is used to define a direction d_z such that

$$Z^T H_k Z d_z = 0 \quad \text{and} \quad g_q^T Z d_z < 0, \tag{4.7}$$

where $g_q = g_k + H_k(x - x_k)$ 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.

4.7. 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 (4.7) that

$$H_k d = 0 \quad \text{and} \quad g_k^T d < 0.$$

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

In short, (NP) behaves like an unbounded linear program (LP) along d, 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 the objective $f(x)$ 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 $c(x) \geq -b$ (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. Basis Handling in SQOPT. The null-space methods in sections 4.3–4.5 require frequent solution of systems involving B and B^T. SQOPT uses the package LUSOL [58] for four purposes:

- to obtain sparse LU factors of a given basis B;
- to replace certain columns of B when it appears singular or ill-conditioned;
- to find a better-conditioned B by reordering the columns of a given set $(B \;\; S)$;
- to update the LU factors when a column of B is replaced by a column of S.

The next sections discuss each function in turn.

5.1. Threshold Pivoting in LUSOL. Stability in the LU factorization of a square or rectangular matrix A is achieved by bounding the off-diagonal elements of L and/or U. There are many ways to do this, especially in the sparse case. In LUSOL [58], L has unit diagonals. Each elimination step chooses ℓ and u^T, the next column of L and row of U, and then updates the remaining rows and columns of A according to $A \leftarrow A - \ell u^T$ (creating an empty row and column). Let

$$\begin{aligned} \tau_L &= \text{the } \textit{factor tolerance} \text{ such that } |L_{ij}| \le \tau_L \quad (1 < \tau_L \le 100 \text{ say}), \\ A_l &= \text{the remaining matrix to be factored after } l \text{ steps.} \end{aligned}$$

For most factorizations, LUSOL uses a *threshold partial pivoting* (TPP) strategy similar to that in LA05 [88] and MA28 [35]. A classical Markowitz criterion is used to choose an entry a_{pq} from a sparse row and column of A_l to become the next pivot element (the next diagonal of U). To be acceptable, a_{pq} must be sufficiently large compared to *other nonzeros in its own column*: $|a_{pq}| \ge \max_i |a_{iq}|/\tau_L$.

With $\tau_L \in [2, 100]$, TPP usually performs well in terms of balancing stability and sparsity, but it cannot be classed as a *rank-revealing LU* (RRLU) factorization method, because it is not especially good at revealing near-singularity and its cause. For example, any triangular matrix A gives $L = I$ and $U = A$ for all values of τ_L (a perfect L with maximum sparsity but little hint of possible ill-conditioning).

For greater reliability, LUSOL includes two RRLU factorizations as follows:

- *Threshold rook pivoting* (TRP), in which a_{pq} must be sufficiently large compared to other nonzeros in its *own column* and its *own row*:
$$|a_{pq}| \ge \max_i |a_{iq}|/\tau_L \quad \text{and} \quad |a_{pq}| \ge \max_j |a_{pj}|/\tau_L.$$
- *Threshold complete pivoting* (TCP), in which a_{pq} must be sufficiently large compared to *all nonzeros in* A_l:
$$|a_{pq}| \ge \max_{i,j} |a_{ij}|/\tau_L.$$

The TCP option was implemented first [81] and has proved valuable for rank-detection during the optimization of Markov decision chains [80] and within SNOPT [54]. In some cases its strict pivot test leads to rather dense LU factors. The TRP option is typically more efficient [82] and in practice its rank-revealing properties are essentially as good as for TCP. (Note that all options can fail to detect near-singularity in certain matrices with regular structure. A classic example is a triangular matrix with 1s on the diagonal and -1s above the diagonal.)

In general, SQOPT uses TPP with tolerance $\tau_L < 4.0$ for all basis factorizations. If tests suggest instability for a certain basis B (large $\|b - Bx\|_\infty$ or $\|x\|_\infty$ or $\|\pi\|_\infty$ following solution of $Bx = b$ or $B^T\pi = c$), the factorization is repeated as often as necessary with a decreasing sequence of tolerances. In the current version of SQOPT, the sequence is $\tau_L = 3.99,\ 2.00,\ 1.41,\ 1.19,\ 1.09,\ 1.02,\ 1.01$ (powers of $\sqrt{3.99}$). If necessary, a switch is made to TRP with the same decreasing sequence of τ_L values, and then TCP is used with the same values. If all three sequences are exhausted, SQOPT terminates with a "numerical error" condition. However, certain basis repairs may be invoked beforehand, as described next.

5.2. Basis Repair (Square or Singular Case). Whenever a basis is factored, LUSOL signals "singularity" if any diagonals of U are judged small, and indicates which unit vectors (corresponding to slack variables) should replace the associated columns of B. The modified B is then factored.

The process may need to be repeated if the factors of B are not sufficiently rank-revealing. Behavior of this kind is exhibited by one of the CUTEr problems (section

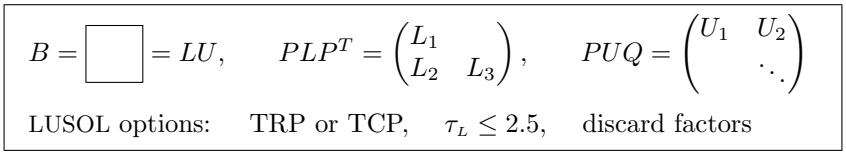

Fig. 1 *BR factorization (rank detection for square B).*

7.2) if we use only the TPP pivoting option in LUSOL when the first basis is factored. Problem *drcavty2* is a large square system of nonlinear equations (10000 constraints and variables, 140000 Jacobian nonzeros). The first TPP factorization with $\tau_L = 3.99$ indicated 249 singularities. After slacks were inserted, the next factorization with $\tau_L = 2.0$ indicated 88 additional singularities, then a further (39, 16, 8, 7, 4) as τ_L decreased from 1.41 to 1.02, and then (7, 7, 7, 5, 1) with $\tau_L = 1.01$ before the basis was regarded as suitably nonsingular (a total of 13 TPP factorizations and 438 slacks replacing rejected columns of B). Since L and U each had about a million nonzeros in all factorizations, the repeated failures were rather expensive.

In contrast, a single TRP factorization with $\tau_L = 2.5$ indicated 102 singularities, after which the modified B proved to be very well-conditioned. The factors were of similar sparsity, and the optimization proceeded significantly more quickly.

For such reasons, SQOPT includes a special "BR factorization" for estimating the rank of a given B, using the LUSOL options shown in Figure 1. P and Q are the row and column permutations that make L unit triangular and U upper triangular, with small elements in the bottom right if B is close to singular. To save storage, the factors are discarded as they are computed. A normal "B factorization" then follows.

BR factorization is the primary recourse when B seems singular or when unexpected growth occurs in $\|x\|$ or $\|\pi\|$. It has proved valuable for some other CUTEr problems arising from partial differential equations (*bratu2d*, *bratu3d*, *porous1*, and *porous2*). A regular "marching pattern" is sometimes present in B, particularly in the first *triangular* basis following a cold start. With TPP, the factors display no small diagonals in U, yet the BR factors reveal a large number of dependent columns. Thus, although condition estimators are known that could tell us "this B is ill-conditioned" (e.g., [68]), LUSOL's RRLU options are more useful in telling us *which columns* are causing the poor condition, and which slacks should replace them.

5.3. Basis Repair (Rectangular Case). When superbasic variables are present, the permutation P in (4.3) clearly affects the condition of B and Z. SQOPT therefore applies an occasional rectangular "BS factorization" to choose a new P, using the options shown in Figure 2.

For simplicity we assume that there are no nonbasic columns in W. A basis partition is given by

$$PW^T \equiv \begin{pmatrix} B^T \\ S^T \end{pmatrix} = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} U_1 Q^T,$$

and the required null-space matrix satisfying $WZ = 0$ is

$$Z \equiv P^T \begin{pmatrix} -B^{-1}S \\ I \end{pmatrix} = P^T \begin{pmatrix} -L_1^{-T} L_2^T \\ I \end{pmatrix}. \tag{5.1}$$

$$W^T = \boxed{} = LU, \qquad PLP^T = \begin{pmatrix} L_1 & \\ L_2 & I \end{pmatrix}, \qquad PUQ = \begin{pmatrix} U_1 \\ 0 \end{pmatrix}$$

LUSOL options: TPP or TRP, $\tau_L \leq 2.5$, discard factors

Fig. 2 *BS factorization (basis detection for rectangular W).*

With $\tau_L \leq 2.5$, L and L_1 are likely to be well-conditioned, and $\zeta \equiv \|L_1^{-T} L_2^T\|$ is unlikely to be large. (It can be bounded by a polynomial function of τ_L.) The extreme singular values of Z are $\sigma_{\min} \geq 1$ and $\sigma_{\max} \approx 1 + \zeta$. It follows that Z should be well-conditioned *regardless of the condition of* W.

SQOPT applies this basis repair at the beginning of a warm start (when a potential B-S ordering is known). To prevent basis repair at *every* warm start—i.e., every major iteration of SNOPT—a normal $B = LU$ factorization is computed first with the current (usually larger) tolerance τ_L. If U appears to be more ill-conditioned than after the last repair, a new repair is invoked. The relevant test on the diagonals of U is tightened gradually to ensure that basis repair occurs periodically (even during a single major iteration if a QP subproblem requires many iterations).

Although the rectangular factors are discarded, we see from (5.1) that a normal factorization of B allows iterations to proceed with an equivalent Z. (A BR factorization may be needed to repair B first if W happens to be singular.)

5.4. Basis Updates. When a QP iteration requires replacement of a column of B, the LU factors must be updated in a stable way. LUSOL uses the approach suggested by Bartels and Golub [3]. The sparse implementation is analogous to that of Reid [88, 89].

6. SQP Algorithm Details. A practical SQP algorithm requires many features to achieve reliability and efficiency. We discuss some more of them here before summarizing the main algorithmic steps.

6.1. The Initial Point. To take advantage of a good starting point x_0, we apply SQOPT to one of the "proximal-point" problems

(PP1) $\underset{x}{\text{minimize}} \quad \|\bar{x} - \bar{x}_0\|_1$
subject to the linear constraints and bounds

or

(PP2) $\underset{x}{\text{minimize}} \quad \|\bar{x} - \bar{x}_0\|_2^2$
subject to the linear constraints and bounds,

where $\bar{x}$ and $\bar{x}_0$ correspond to the nonlinear variables in x and x_0. 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.

In practice we prefer problem (PP1), as it is linear and can use SQOPT's implicit elastic bounds. (We temporarily set the bounds on the nonlinear variables to be

$\bar{x}_0 \leq \bar{x} \leq \bar{x}_0$.) Note that problem (PP2) 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 with this option, we terminate SQOPT early by specifying a loose optimality tolerance.

6.2. Undefined Functions. If the constraints in (PP1) or (PP2) prove to be infeasible, SNOPT solves problem (FLP) (see section 1.1) and terminates without computing the nonlinear functions. The problem was probably formulated incorrectly.

Otherwise, the linear constraints and bounds define a certain "linear feasible region" $\mathcal{R}_L$, and all iterates satisfy $x_k \in \mathcal{R}_L$ to within a feasibility tolerance (as with NPSOL). Although SQP algorithms might converge more rapidly sometimes if all constraints were treated equally, the aim is to help prevent function evaluations at obvious singularities.

In practice, the functions may not be defined everywhere within $\mathcal{R}_L$, and it may be an unbounded region. Hence, the function routines are permitted to return an "undefined function" signal. If the signal is received from the *first* function call (before any line search takes place), SNOPT terminates. Otherwise, the line search backtracks and tries again.

6.3. Early Termination of QP Subproblems. SQP theory usually assumes that the QP subproblems are solved to optimality. For large problems with a poor starting point and $H_0 = I$, many thousands of iterations may be needed for the first QP, building up many degrees of freedom (superbasic variables) that are promptly eliminated by more thousands of iterations in the second QP.

In general, it seems wasteful to expend much effort on any QP before updating H_k and the constraint linearization. Murray and Prieto [74] suggest one approach to terminating the QP solutions early, requiring that at least one QP stationary point be reached. The associated theory implies that any subsequent point $\widehat{x}_k$ generated by a QP solver is suitable, provided that $\|\widehat{x}_k - x_k\|$ is nonzero. In SNOPT we have implemented a method within this framework that has proved effective on many problems. Conceptually we could perform the following steps:

- Fix many variables at their current value.
- Perform one SQP major iteration on the reduced problem (solving a smaller QP to get a search direction for the nonfixed variables).
- Free the fixed variables, and complete the major iteration with a "full" search direction that happens to leave many variables unaltered.
- Repeat.

Normal merit-function theory should guarantee progress at each stage on the associated reduced *nonlinear* problem. We are simply suboptimizing.

In practice, we are not sure which variables to fix at each stage, the reduced QP could be infeasible, and degeneracy could produce a zero search direction. Instead, the choice of which variables to fix is made within the QP solver. The following steps are performed:

- Perform QP iterations on the full problem until a feasible point is found or elastic mode is entered.
- Continue iterating until certain limits are reached and not all steps have been degenerate.
- Freeze nonbasic variables that have not yet moved.
- Solve the reduced QP to optimality.

Rather arbitrary limits may be employed and perhaps combined. We have implemented the following as user options:

- `Minor iterations limit` (default 500) suggests termination if a reasonable number of QP iterations have been performed (beyond the first feasible point).
- `New superbasics limit` (default 99) suggests termination if the number of free variables has increased significantly (since the first feasible point).
- `Minor optimality tolerance` (default 10^{-6}) specifies an optimality tolerance for the final QPs.

Internally, SNOPT sets a loose but decreasing optimality tolerance for the early QPs (somewhat smaller than a measure of the current primal-dual infeasibility for (NP)). This "loose tolerance" strategy provides a dynamic balance between major and minor iterations in the manner of inexact Newton methods [29].

6.4. Linearly Constrained Problems. For problems with linear constraints only, the maximum step length 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 $Q^T H_k Q$:

$$\bar{H}_Q = H_Q + \theta_k y_Q y_Q^T - \phi_k q_Q q_Q^T, \tag{6.1}$$

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 (6.1). Given δ_k and y_k, if we had all of the Cholesky factor R_Q, it could be updated directly as

$$R_Q + uv^T, \qquad w = R_Q \delta_Q, \quad u = w/\|w\|_2, \quad v = \sqrt{\theta_k}\, y_Q - R_Q^T u$$

(see Goldfarb [62] and Dennis and Schnabel [30]). This rank-one modification of R_Q could be restored to upper-triangular form by applying two sequences of plane rotations from the left [50].

The same sequences of rotations can be generated even though not all of R_Q is present. 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 $\bar{H}_Q$ from (6.1):

1. Compute $q_k = H_k \delta_k$ and $t = Z^T q_k$.
2. Define $\mu = \phi_k^{1/2} = 1/\|w\|_2 = 1/(\delta_k^T H_k \delta_k)^{1/2} = 1/(q_k^T \delta_k)^{1/2}$.
3. Solve $R^T w_Z = t$.
4. Define $u_Z = \mu w_Z$ and $\sigma = (1 - \|u_Z\|_2^2)^{1/2}$.

5. Apply a backward sweep of n_Z rotations P_1 in the planes (n_Z+1, i), $i = n_Z : 1$, to give a triangular $\widehat{R}$ and a "row spike" r^T:

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

6. Apply a forward sweep of n_Z rotations P_2 in the planes (i, n_Z+1), $i = 1 : n_Z+1$, to restore the upper-triangular form:

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

6.5. Summary of the SQP Algorithm. The main steps of the SNOPT algorithm follow. 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 qualitatively. Specific values for γ are given in section 2.11.

0. Apply the QP solver to problem (PP1) or (PP2) to find a point close to x_0 satisfying the linear constraints. If the PP problem is infeasible, declare problem (NP) infeasible. Otherwise, a working-set matrix W_0 is returned. Set $k = 0$, evaluate functions and gradients at x_0, and initialize penalty parameters $\rho_i = 0$.
1. Factorize W_k.
2. Define s_k to minimize the merit function as a function of the slacks s.
3. 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.
4. Form the reduced Hessian $\bar{Z}_k^T H_k \bar{Z}_k$, and compute its Cholesky factor.
5. 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 (f is unbounded below in the feasible region). Otherwise (if the QP is unbounded), go to step 7 (f is unbounded below in the feasible region if a feasible point exists).
6. 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 7. Otherwise, go to step 8.
7. If elastic mode has not been initiated, enter elastic mode and repeat step 5. Otherwise, if γ has not reached its maximum value, increase γ and repeat step 5. Otherwise, declare the problem infeasible.
8. Update the penalty parameters as in (2.4).
9. Find a step length α_k that gives a sufficient reduction in the merit function (2.3). Set $x_{k+1} = x_k + \alpha_k(\widehat{x}_k - x_k)$, and $\pi_{k+1} = \pi_k + \alpha_k(\widehat{\pi}_k - \pi_k)$. In the process, evaluate functions and gradients at x_{k+1}.
10. 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 $y_k^T\delta_k < \sigma_k$ (2.6), 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.

Table 1 *Notation in tables of results.*

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).
e	Essentially optimal (i.e., optimal if τ_P or τ_D were increased by a factor of 10).
c	Final nonoptimal point could not be improved.
h	The number of Hessian-vector products required by the QP solver.

11. If $y_k^T\delta_k \geq \sigma_k$, apply the BFGS update to H_k, using the pair $(H_k\delta_k, y_k)$.
12. Define W_{k+1} from $\widehat{W}_k$, set $k \leftarrow k+1$, and repeat from step 1.

Apart from the function and gradient evaluations, most of the computational effort lies in steps 1 and 4. Steps 3 and 5 may also involve significant work if the QP subproblem requires many minor iterations. Typically this will happen only during the early major iterations.

7. Numerical Results. SNOPT and SQOPT implement all of the techniques described in sections 2–6. The Fortran 77 coding is compatible with Fortran 90 and 95 compilers and permits recursive calls, or re-entrant calls in a multithreaded environment, as well as translation into C via $f2c$ [37] (though these features are not used here).

We give the results of applying SNOPT 7.1 of January 2005 to problems in the CUTEr and COPS 3.0 test collections [11, 10, 31, 33]. Function and gradient values were used throughout (but not second derivatives). All runs were made on a Linux PC with 2GB of RAM and two 3.06GHz Xeon processors (only one being used for each problem solution). The g77 compiler was used with `-O` option specifying full code optimization. The floating-point precision was 2.22×10^{-16}. Table 1 defines the notation used in the tables of results.

7.1. Options for SNOPT. Figure 3 gives the SNOPT run-time options used, most of which are default values. Linear constraints and variables are scaled (`Scale option 1`), and the first basis is essentially triangular (`Crash option 3`).

`Elastic weight` sets $\gamma_0 = 10^4$ in (2.10).

For the 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 `Hessian updates` specifying that H_k should be reset to the current Hessian diagonal every $\ell = 5$ major iterations (see section 3.3).

The `Major feasibility` and `optimality` tolerances set τ_P and τ_D in section 2.11 for problem (NP). The `Minor` tolerances are analogous options 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).

`Penalty parameter` initializes the penalty parameters ρ_i for the merit function.

`Reduced Hessian dimension` specifies the maximum size of the dense reduced Hessian available for SQOPT. If the number of superbasics exceeds this value during the QP solution, SQOPT solves (4.5) using the CG-type solver SYMMLQ [83].

`Violation limit` sets τ_V in section 2.7 to define an expanded feasible region in which the objective is expected to be bounded below.

```
BEGIN SNOPT Problem
   Minimize
   Crash option                      3
   Derivative level                  3
   Elastic weight                1.0E+4
   Hessian updates                   5
   Iterations                    90000
   Major iterations               2000
   Minor iterations                500
   LU partial pivoting
   Major feasibility tolerance   1.0E-6
   Major optimality  tolerance   2.0E-6
   Minor feasibility tolerance   1.0E-6
   Minor optimality  tolerance   1.0E-6
   New superbasics                   99
   Line search tolerance            0.9
   Penalty parameter                0.0
   Proximal point method              1
   Reduced Hessian dimension        750
   Scale option                       1
   Step limit                       2.0
   Unbounded objective           1.0E+15
   Verify level                      -1
   Violation limit               1.0E+6
END SNOPT Problem
```

Fig. 3 *The SNOPT options file.*

Table 2 *The* 1020 *CUTEr problems listed by type and frequency.*

Frequency	Type	Characteristics
24	LP	Linear obj, linear constraints
167	QP	Quadratic obj, linear constraints
160	UC	Nonlinear obj, no constraints
129	BC	Nonlinear obj, bound constraints
70	LC	Nonlinear obj, linear constraints
380	NC	Nonlinear obj, nonlinear constraints
90	FP	No objective

7.2. Results on the CUTEr Test Set. The CUTEr distribution of December 20, 2004, contains 1020 problems in standard interface format (SIF). A list of the CUTEr problem types and their frequencies is given in Table 2. Although many problems allow for the number of variables and constraints to be adjusted in the SIF file, our tests used the dimensions set in the CUTEr distribution. This gave problems ranging in size from *hs1* (two variables and no constraints) to *cont5-qp* (40601 variables and 40201 constraints) and *portsnqp* (100000 variables and 3 constraints).

From the complete set of 1020 problems, 13 were omitted as follows:

- 4 nonsmooth problems (*bigbank*, *gridgena*, *hs87*, and *net4*);
- 4 problems with undefined variables or floating-point exceptions in the SIF file (*lhaifam*, *recipe*, *s365*, and *s365mod*);
- 5 problems too large for the SIF decoder (*chardis0*, *chardis1*, *harkerp2*, *yatp1sq*, and *yatp2sq*).

Some of the CUTEr problems have many degrees of freedom at the solution. Of the 1007 problems attempted, 183 have more than 2000 degrees of freedom, with the largest nonlinearly constrained problem (*jannson3*) having almost 20000 superbasic variables at the solution. SNOPT was applied to these problems using the options listed in Figure 3. The 824 problems with fewer than 2000 degrees of freedom were

Table 3 *Summary: SNOPT on the smooth CUTEr problems.*

	$n_Z \leq 2000$	$n_Z > 2000$
Problems attempted	824	183
Optimal	735	115
Unbounded	3	1
Infeasible	16	0
Optimal, low accuracy	12	17
Cannot be improved	9	3
False infeasibility	17	0
Terminated	32	47

solved with the option `Reduced Hessian dimension 2000`, which allows the dimension of R in (4.4) to grow to 2000, thereby preventing use of the CG solver. In all the runs, no special information was used in the case of QP and FP problems—i.e., each problem was assumed to have a general nonlinear objective. The results are summarized in Table 3.

Discussion. Problems *a2nndnil*, *a5nndnil*, *arglale*, *arglble*, *arglcle*, *flosp2hh*, *flosp2hl*, *flosp2hm*, *ktmodel*, *lincont*, *model*, *nash*, *sawpath*, and *woodsne* have infeasible linear constraints, but were included anyway. The objectives for *fletchbv*, *indef*, *mesh*, and *static3* are unbounded below in the feasible region. SNOPT correctly diagnosed the special features of these problems.

The declaration of optimality by SNOPT means that the final point satisfies the first-order optimality conditions (2.9) for the default feasibility and optimality tolerances $\tau_P = 10^{-6}$ and $\tau_D = 2 \times 10^{-6}$. We emphasize that this point may not be a constrained local minimizer for the problem. For example, the final "optimal" point for the problem *hs13* is not a constrained local minimizer because the constraint qualification does not hold there. Similarly, the final point for the problem *optmass* satisfies the *first-order* but not *second-order* conditions for optimality. Verifying second-order conditions requires second derivatives.

Performance on Problems with Few Superbasics. A total of 12 problems (*chebyqad*, *cresc132*, *cresc50*, *djtl*, *hues-mod*, *lakes*, *liswet4*, *mancino*, *marine*, *ncb20b*, *ncvxbqp3*, and *pfit4*) were terminated at a point that is essentially optimal; i.e., a point that would be considered optimal if the feasibility *or* the optimality tolerance were ten times the default. The AMPL implementation of *marine* was solved successfully as part of the COPS 3.0 collection (see section 7.3).

SNOPT reported 19 problems (*argauss*, *arwdhne*, *cont6-qq*, *drcavty3*, *eigenb*, *eigenc*, *eigmaxb*, *fletcher*, *flosp2th*, *growth*, *himmelbd*, *hs90*, *junkturn*, *lewispol*, *lootsma*, *lubrifc*, *nystrom5*, *optcdeg3*, and *vanderm3*) with infeasible nonlinear constraints. 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. Of these 19 problems, all but 2 cases must be counted as failures because they are known to have feasible points. The two exceptions, *flosp2th* and *junkturn*, have no known feasible points. To gain further assurance that these problems 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 section 1.1). In both cases, the final sum of constraint violations was comparable to that obtained with the composite objective. We conjecture that these problems are infeasible.

Problems *fletcher* and *lootsma* have feasible solutions, but their initial points are infeasible and stationary for the sum of infeasibilities, and thus SNOPT terminated immediately. These problems are also listed as failures.

SNOPT was unable to solve 32 cases within the allotted 2000 major iterations (*a4x12*, *allinqp*, *bqpgauss*, *catena*, *catenary*, *discs*, *dixon3dq*, *eigenals*, *eigenbls*, *eigencls*, *extrosnb*, *glider*, *heart6*, *heart6ls*, *hydc20ls*, *lubrif*, *msqrtals*, *msqrtbls*, *nonmsqrt*, *nuffield*, *optctrl3*, *optctrl6*, *palmer5e*, *palmer7e*, *pfit3*, *qr3dls*, *reading4*, *robotarm*, *sinrosnb*, *ubh1*, *vanderm1*, and *vanderm2*). AMPL implementations of *glider* and *robotarm* were solved successfully (see section 7.3).

Another 9 problems could not be improved at a nonoptimal point: *arglinb*, *arglinc*, *bleachng*, *brownbs*, *meyer3*, *penalty3*, *semicn2u*, *ubh5*, and *vanderm4*. SNOPT essentially found the solution of the badly scaled problems *brownbs* and *meyer3* but was unable to declare optimality. The problems *ubh1* and *ubh5* appear to have singular Jacobians near the solution. SNOPT was unable to find a well-conditioned null-space basis at the final (nonoptimal) iterate of *ubh5*.

Performance on Problems with Many Superbasics. The 183 problems with more than 2000 degrees of freedom at the solution provide a substantial test of the CG solver in SQOPT. In this situation, once the QP working set settles down, the efficiency of SNOPT depends largely on whether or not the limited-memory method is able to adequately represent the Lagrangian Hessian.

A total of 17 problems (*biggsb1*, *clplatea*, *clplateb*, *fminsurf*, *jimack*, *lukvle1*, *lukvle4*, *lukvle6*, *lukvle17*, *lukvle18*, *lukvli13*, *lukvli16*, *lukvli17*, *minsurfo*, *odc*, *orthregc*, and *orthregf*) were terminated at a point that would be considered optimal if the feasibility or the optimality tolerance were ten times the default.

The 3 problems *lukvle15*, *powellsq*, and *qrtquad* could not be improved at a nonoptimal point. SNOPT was unable to solve 47 problems within the assigned number of 2000 major iterations (*bratu1d*, *chainwoo*, *chenhark*, *clplatec*, *coshfun*, *curly10*, *curly20*, *curly30*, *drcav1lq*, *drcav2lq*, *drcav3lq*, *fletcbv3*, *fletchcr*, *genhumps*, *hanging*, *jnlbrngb*, *lch*, *lminsurf*, *lukvle9*, *lukvle11*, *lukvle16*, *lukvli1*, *lukvli9*, *lukvli10*, *lukvli11*, *lukvli12*, *lukvli15*, *lukvli18*, *modbeale*, *nlmsurf*, *noncvxu2*, *noncvxun*, *odnamur*, *orthrgds*, *powellsg*, *raybendl*, *raybends*, *sbrybnd*, *scond1ls*, *scosine*, *scurly10*, *scurly20*, *scurly30*, *sinquad*, *sparsine*, *testquad*, and *tquartic*). Many of these problems have no constraints or only simple bounds, and in these cases, the large number of major iterations is consistent with results obtained by other limited-memory quasi-Newton methods (see, e.g., [22, 51]).

If the infeasible LC problems, the unbounded problems, and the 2 (conjectured) infeasible problems are counted as successes, SNOPT solved a grand total of 870 of the 1007 problems attempted. In another 29 cases, SNOPT found a point that was within a factor 10 of satisfying the convergence test.

Given the size and diversity of the test set, these results provide good evidence of the robustness of first-derivative SQP methods when implemented with an augmented Lagrangian merit function and an elastic variable strategy for treating infeasibility of the original problem and the QP subproblems.

7.3. Results on the COPS 3.0 Test Set. Next we describe tests on the 22 problems in the COPS 3.0 test collection [10, 31, 32] implemented in the AMPL modeling language [45, 46, 1]. The dimension of a particular instance of a COPS problem is determined by one or more parameters assigned in its AMPL data file. For each of the 22 COPS problems, [33] gives the results of several optimization algorithms on a

Table 4 *Dimensions of the AMPL versions of the COPS problems.*

No.	Problem	Type	Variables	Constraints		
				Linear	Nonlinear	Total
1	*bearing*	QP	5000	0	0	0
2	*camshape*	NC	1200	1200	1201	2401
3	*catmix*	NC	2401	1	1600	1601
4	*chain*	NC	800	401	1	402
5	*channel*	FP	6398	3198	3200	6398
6	*dirichlet*	FP	8981	1	41	42
7	*elec*	NC	600	1	200	201
8	*gasoil*	NC	4001	799	3200	3999
9	*glider*	NC	1999	1	1600	1601
10	*henon*	NC	10801	1	81	82
11	*lane_emden*	NC	19240	1	81	82
12	*marine*	NC	6415	3193	3200	6415
13	*methanol*	NC	4802	1198	3600	4798
14	*minsurf*	BC	5000	0	0	0
15	*pinene*	NC	8000	1996	6000	7996
16	*polygon*	NC	398	199	19900	20099
17	*robot*	NC	7198	1	4800	4801
18	*rocket*	NC	6401	1	4800	4801
19	*steering*	NC	3999	1	3200	3201
20	*tetra*	NC	2895	193	8409	8602
21	*torsion*	QP	5000	0	0	0
22	*triangle*	NC	3578	243	3726	3969

range of cases obtained by varying only one of the model parameters. In all but one of the models we consider the largest problem from each set of cases (see Table 4). The exception was the model *dirichlet*, where the second largest size was used. (For this problem, the Hessian of the Lagrangian is increasingly ill-conditioned as the problem dimension grows and the limited-memory algorithm was unable to solve the largest case, regardless of the number of limited-memory updates used.) Problems *bearing* and *torsion* are quadratic programs with only bound constraints. In the case of a QP, the AMPL interface to SNOPT calls SQOPT directly.

Table 5 gives the results of SNOPT on the 22 COPS problems. The default AMPL options (including problem preprocessing) were used in each case. With the exception of *triangle*, the default options of Figure 3 were used. For *triangle* the option `Penalty parameter` initialized the penalty parameters to 10^5 to prevent the objective becoming unbounded.

Discussion. It is not clear why the AMPL formulations of *glider* and *robot* (problem *robotarm* in the CUTEr set) can be solved relatively easily, but not the CUTEr versions. Reruns with AMPL option `presolve 0` did not need significantly more cpu time, which implies that preprocessing is not the reason for the performance difference.

The COPS problems were also used to investigate the effect of the number ℓ of limited-memory updates (section 3.3) on the performance of SNOPT. Tables 6 and 7 give times and major iterations for different choices for ℓ.

The results are typical of the performance of SNOPT in practical situations.

- Small values of ℓ can give low computation times but may adversely affect robustness on more challenging problems. For example, $\ell = 5$ gave the one run in which the AMPL formulations of *pinene* and *polygon* could not be solved to full accuracy.

Table 5 *SNOPT on the COPS 3.0 problems.*

No.	Problem	Mnr	Mjr	Fcn	Obj	Con	n_Z	cpu
1	*bearing*	3352	–	4106^h	1.550420E-01	0.0E+00	3350	50.3
2	*camshape*	4908	10	20	4.234466E+00	1.5E-07	0	5.8
3	*catmix*	2949	37	39	-4.805546E-02	1.8E-07	618	39.0
4	*chain*	1667	48	73	5.068532E+00	1.2E-07	799	9.0
5	*channel*	3999	5	7	1.000000E+00	3.3E-05	0	29.8
6	*dirichlet*	5902	82	108	1.714656E-02	8.0E-07	5355	279.4
7	*elec*	893	490	550	1.843916E+04	5.4E-13	400	38.6
8	*gasoil*	2589	18	21	5.236596E-03	3.1E-07	3	12.9
9	*glider*	17834	79	164	1.247974E+03	6.4E-12	359	47.8
10	*henon*	11002	239	283	1.179065E+02	5.4E-10	9410	893.3
11	*lane_emden*	5795	152	179	9.284899E+00	4.2E-09	5414	296.7
12	*marine*	3375	47	62	1.974651E+07	7.3E-12	22	30.9
13	*methanol*	3990	104	183	9.022290E-03	6.1E-12	4	53.2
14	*minsurf*	159809	1298	1421	2.506950E+00	0.0E+00	4782	1672.1
15	*pinene*^{e}	4907	38	107	1.987217E+01	5.8E-13	5	82.8
16	*polygon*^{e}	7649	2000	2191	7.853051E-01	2.7E-08	197	1265.8
17	*robot*	10268	16	33	9.140942E+00	3.3E-08	0	105.5
18	*rocket*	3884	11	25	1.005380E+00	7.1E-10	332	27.5
19	*steering*	1911	72	95	5.545713E-01	2.8E-07	799	25.7
20	*tetra*	2959	50	55	1.049511E+04	0.0E+00	799	47.3
21	*torsion*	3504	–	4258^h	-4.182392E-01	0.0E+00	3450	71.4
22	*triangle*	5526	153	171	4.215232E+03	0.0E+00	3578	35.3

- As ℓ is increased, the number of major iterations tends to decrease. However, the numerical performance remains relatively stable. (For example, the same local solution was always found for the highly nonlinear problem *polygon*.)
- The value $\ell = 5$ often gives the lowest computation time—particularly for problems with large numbers of superbasic variables. As ℓ is increased, the solution time often decreases initially, but then increases as the cost of the products $H_k v$ increases. This would be more closely reflected in the total computation time for Table 6 if it were not for *polygon*, whose time improves dramatically because of a better Hessian approximation.

The choice of default value $\ell = 5$ is intended to provide efficiency on problems with many superbasics without a significant loss of robustness.

8. Alternative QP Solvers. Where possible, we have defined the SQP algorithm to be independent of the QP solver. Of course, SQOPT's implicit elastic bounds and warm start features are highly desirable.

Here we discuss future possibilities for solving the KKT system (4.1) within the QP solver, allowing for many degrees of freedom (when W has many more columns than rows). Note that the limited-memory Hessians (3.2)–(3.3) have the form

$$H_k = H_0 + UU^T - VV^T = G_k^T G_k, \tag{8.1}$$

$$G_k = H_0^{1/2} \textstyle\prod_j (I + u_j v_j^T) \tag{8.2}$$

for certain quantities U, V, u_j, v_j.

8.1. Range-Space Methods. If all variables appear nonlinearly, H_k is positive definite. A "range-space" approach could then be used to solve systems (4.1) as W changes. This amounts to maintaining factors of the Schur complement matrix $S = WH_k^{-1}W^T = R_k^T R_k$, where R_k comes from a QR factorization of T_k satisfying

Table 6 *COPS problems: cpu time for increasing numbers of limited-memory updates.*

Problem	Limited-memory updates					
	5	10	15	20	25	30
bearing	50.3	48.7	47.4	51.5	49.4	49.6
camshape	5.8	6.1	6.0	5.8	5.9	5.8
catmix	39.0	96.3	99.7	83.9	164.1	97.3
chain	9.0	9.0	8.8	9.4	10.5	10.6
channel	29.8	29.6	30.1	30.2	30.4	32.1
dirichlet	279.4	287.0	396.8	539.6	749.4	848.4
elec	38.6	38.1	24.4	40.2	45.3	43.5
gasoil	12.9	12.9	12.9	13.1	13.8	13.1
glider	47.8	54.8	78.1	73.9	78.4	90.5
henon	893.3	888.1	1023.5	1206.3	2077.7	2793.8
lane_emden	296.7	461.6	731.6	891.6	1333.6	1734.0
marine	30.9	32.7^e	34.7^e	33.7	36.4	38.0
methanol	53.2	42.5	43.0	45.0	46.1	42.0
minsurf	1672.1	1626.7	1170.5	1525.1	2232.3	2494.7
pinene	82.8^e	78.0	74.4^e	74.4^e	72.8^e	71.3^e
polygon	1265.8^e	645.3	358.2	393.4	404.1	221.3
robot	105.5	83.0	88.2	91.8	95.9	96.7
rocket	27.5	27.6	27.5	29.0	29.0	29.3
steering	25.7	30.5	33.1	43.1	50.5	52.8
tetra	47.3	51.3	56.4	59.3	67.0	73.1
torsion	71.4	73.9	71.7	76.7	81.0	78.4
triangle	35.3	40.2	46.0	48.9	53.6	60.0
Total cpu	5120.1	4663.9	4463.0	5365.9	7727.2	8976.3

Table 7 *COPS problems: major iterations for increasing numbers of limited-memory updates.*

Problem	Limited-memory updates					
	5	10	15	20	25	30
camshape	10	10	10	10	10	10
catmix	37	106	97	64	131	63
chain	48	47	35	45	55	57
channel	5	5	5	5	5	5
dirichlet	82	64	52	57	55	60
elec	490	446	270	372	392	392
gasoil	18	16	16	16	16	16
glider	79	73	112	172	224	227
henon	239	155	158	145	100	114
lane_emden	152	134	114	127	119	99
marine	47	42	49	43	46	50
methanol	104	63	59	79	90	59
minsurf	1298	915	510	506	513	468
pinene	38	34	28	28	26	26
polygon	2000	958	465	484	495	218
robot	16	20	22	26	26	26
rocket	11	11	11	11	11	11
steering	72	82	84	92	88	92
tetra	50	47	48	45	47	48
triangle	153	152	163	150	146	152
Total	4949	3380	2308	2477	2595	2193

$G_k^T T_k = W$. It would be efficient for problems with only a few hundred general constraints, so that T_k and R_k could be treated as a dense matrices.

8.2. Least-Squares CG Formulation. With $g_q = g_k + H_k(x - x_k)$, system (4.5) is equivalent to the least-squares problem

$$\min_{d_Z} \left\| \begin{pmatrix} G_k Z \\ \delta I \end{pmatrix} d_Z + \begin{pmatrix} t \\ r/\delta \end{pmatrix} \right\|_2^2, \qquad t = G_k(x - x_k), \quad r = Z^T g_k, \tag{8.3}$$

where t and r both become small as the SQP method converges. This formulation would allow the use of the CG-type solver LSQR [84], which has effective stopping rules to control the accuracy of d_Z.

8.3. Schur-Complement Updates. For limited-memory Hessians of the form $H_k = H_0 + UU^T - VV^T$ (8.1), system (4.1) is equivalent to

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

Following [56, section 3.6.2], if we define

$$K_0 = \begin{pmatrix} H_0 & W^T \\ W & \end{pmatrix}, \qquad S = \begin{pmatrix} I & \\ & -I \end{pmatrix} - \begin{pmatrix} U^T \\ V^T \end{pmatrix} K_0^{-1} \begin{pmatrix} U & V \end{pmatrix},$$

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, U and V are constant, but changes to W would be handled by appropriate updates to S.)

This approach has been explored by Betts and Frank [5, section 5] with $H_0 = I$ (or possibly a sparse finite-difference Hessian approximation). Schur-complement updates have also been implemented in the GALAHAD QP solver QPA [64].

As part of an SQP algorithm, practical success depends greatly on the definition of H_0 and on the BFGS updates that define U and V. Our experience with SNOPT emphasizes the importance of updating H_k even in the presence of negative curvature; hence the precautions of section 2.10.

8.4. Schur-Complement Updates II. For a limited-memory Hessian of the form $H_1 = (I + vu^T)H_0(I + uv^T)$, system (4.1) is equivalent to

$$\begin{pmatrix} H_0 & W^T & \bar{u} & v \\ W & & & \\ \bar{u}^T & & \gamma & -1 \\ v^T & & -1 & \end{pmatrix} \begin{pmatrix} p \\ q \\ r \\ s \end{pmatrix} = \begin{pmatrix} g \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad \bar{u} = H_0 u, \qquad \gamma = u^T H_0 u.$$

It remains to be seen whether this will permit multiple product-form updates required by (3.3).

9. 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. The algorithm minimizes a sequence of augmented Lagrangian functions, using a QP subproblem at each stage to predict the set of active constraints and to generate a search direction in both the primal and the dual variables. Convergence is assured from arbitrary starting points.

The constraints in the QP subproblems are linearizations of the original constraints (requiring only first derivatives), but the QP objective must approximate the Lagrangian more closely. As with interior-point methods, the most promising way to achieve efficiency would be 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-semidefinite 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 frequency (typically every 5 or 10 major iterations).

The QP solver, SQOPT, works with a sequence of reduced-Hessian systems of the form $Z^T H_k Z d = -Z^T g$, where Z is a rectangular matrix operator with n_Z columns (the number of degrees of freedom). SQOPT can deal with the reduced-Hessian systems in various ways, depending on the size of n_Z. If many constraints are currently active in the QP, n_Z is not excessively large and it is efficient to use the dense Cholesky factorization $Z^T H_k Z = R^T R$. Alternatively, SQOPT can maintain a dense quasi-Newton approximation $Z^T H_k Z \approx R^T R$ to avoid the cost of forming and factorizing the reduced Hessian. Another option is to use the CG method. The structure of the reduced Hessian often makes this the most effective method for solving problems with many degrees of freedom (with no preconditioning for the CG method). Finally, SQOPT has the option of using a dense quasi-Newton approximation to part of the reduced Hessian as a preconditioner for the CG solver.

The numerical results in section 7 show that the current version of SNOPT is effective on most of the problems in the CUTEr and COPS 3.0 test sets, including examples with up to 40000 constraints and variables, and some with 20000 degrees of freedom. Earlier comparisons with MINOS have shown greater reliability as a result of methodical treatment of the merit function parameters and of infeasibility (via "elastic variables"), and much greater efficiency when function and gradient evaluations are expensive. Reliability has also improved relative to NPSOL, and the sparse-matrix techniques have permitted production runs on increasingly large trajectory problems.

Future work must take into account the fact that second derivatives are increasingly available. The QP solver should allow for indefinite QP Hessians, and additional techniques are needed to handle even more degrees of freedom.

Acknowledgments. We extend sincere thanks to our colleagues Dan Young and Rocky Nelson of the Boeing Company (formerly McDonnell Douglas Space Systems, Huntington Beach, CA) for their constant support and feedback during the development of SNOPT. We would also like to express our thanks to the AMPL [46], CUTE [11], and CUTEr [63] authors, and to the many contributors to the test problem collections. We also appreciate many suggestions from the referees and SIOPT Associate Editor Jorge Nocedal.

REFERENCES

[1] AMPL Home Page, http://www.ampl.com.
[2] ARKI Consulting & Development A/S, http://www.conopt.com.
[3] R. H. Bartels, *A stabilization of the simplex method*, Numer. Math., 16 (1971), pp. 414–434.
[4] R. H. Bartels, *A penalty linear programming method using reduced-gradient basis-exchange techniques*, Linear Algebra Appl., 29 (1980), pp. 17–32.

[5] J. T. Betts and P. D. Frank, *A sparse nonlinear optimization algorithm*, J. Optim. Theory Appl., 82 (1994), pp. 519–541.
[6] 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.
[7] M. C. Biggs, *Constrained minimization using recursive equality quadratic programming*, in Numerical Methods for Nonlinear Optimization, F. A. Lootsma, ed., Academic Press, London, New York, 1972, pp. 411–428.
[8] P. T. Boggs, A. J. Kearsley, and J. W. Tolle, *A global convergence analysis of an algorithm for large-scale nonlinear optimization problems*, SIAM J. Optim., 9 (1999), pp. 833–862.
[9] P. T. Boggs, A. J. Kearsley, and J. W. Tolle, *A practical algorithm for general large scale nonlinear optimization problems*, SIAM J. Optim., 9 (1999), pp. 755–778.
[10] A. Bondarenko, D. Bortz, and J. J. Moré, *COPS: Large-Scale Nonlinearly Constrained Optimization Problems*, Technical Report ANL/MCS-TM-237, Mathematics and Computer Science division, Argonne National Laboratory, Argonne, IL, 1998. Revised October 1999.
[11] I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *CUTE: Constrained and unconstrained testing environment*, ACM Trans. Math. Software, 21 (1995), pp. 123–160.
[12] I. Bongartz, A. R. Conn, N. I. M. Gould, M. A. Saunders, and Ph. L. Toint, *A Numerical Comparison between the LANCELOT and MINOS Packages for Large-Scale Constrained Optimization*, Report 97/13, Département de Mathématique, Facultés Universitaires de Namur, 1997.
[13] I. Bongartz, A. R. Conn, N. I. M. Gould, M. A. Saunders, and Ph. L. Toint, *A Numerical Comparison between the LANCELOT and MINOS Packages for Large-Scale Constrained Optimization: The Complete Numerical Results*, Report 97/14, Département de Mathématique, Facultés Universitaires de Namur, 1997.
[14] K. W. Brodlie, A. R. Gourlay, and J. Greenstadt, *Rank-one and rank-two corrections to positive definite matrices expressed in product form*, J. Inst. Math. Appl., 11 (1973), pp. 73–82.
[15] G. G. Brown and G. W. Graves, *Elastic Programming: A New Approach to Large-Scale Mixed-Integer Optimization*, 1975. Presented at the ORSA/TIMS meeting, Las Vegas, NV.
[16] G. G. Brown and G. W. Graves, *The XS Mathematical Programming System*, working paper, Department of Operations Research, Naval Postgraduate School, Monterey, CA, 1975.
[17] A. Buckley and A. LeNir, *QN-like variable storage conjugate gradients*, Math. Program., 27 (1983), pp. 155–175.
[18] A. Buckley and A. LeNir, *BBVSCG—a variable storage algorithm for function minimization*, ACM Trans. Math. Software, 11 (1985), pp. 103–119.
[19] R. H. Byrd, *Robust Trust-Region Methods for Constrained Optimization*. Presented at the SIAM Conference on Optimization, Houston, TX, 1987.
[20] R. H. Byrd, J. C. Gilbert, and J. Nocedal, *A trust region method based on interior point techniques for nonlinear programming*, Math. Program., 89 (2000), pp. 149–185.
[21] R. H. Byrd, M. E. Hribar, and J. Nocedal, *An interior point algorithm for large-scale nonlinear programming*, SIAM J. Optim., 9 (1999), pp. 877–900.
[22] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A limited memory algorithm for bound constrained optimization*, SIAM J. Sci. Comput., 16 (1995), pp. 1190–1208.
[23] R. H. Byrd and J. Nocedal, *An analysis of reduced Hessian methods for constrained optimization*, Math. Program., 49 (1991), pp. 285–323.
[24] R. H. Byrd, J. Nocedal, and R. B. Schnabel, *Representations of quasi-Newton matrices and their use in limited-memory methods*, Math. Program., 63 (1994), pp. 129–156.
[25] A. R. Conn, *Constrained optimization using a nondifferentiable penalty function*, SIAM J. Numer. Anal., 10 (1973), pp. 760–779.
[26] A. R. Conn, *Linear programming via a nondifferentiable penalty function*, SIAM J. Numer. Anal., 13 (1976), pp. 145–154.
[27] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*, Lecture Notes in Comput. Math. 17, Springer-Verlag, Berlin, Heidelberg, New York, London, Paris, Tokyo, 1992.
[28] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Trust-Region Methods*, SIAM, Philadelphia, 2000.
[29] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.
[30] J. E. Dennis, Jr. and R. B. Schnabel, *A new derivation of symmetric positive definite secant updates*, in Nonlinear Programming, 4 (Proc. Sympos., Special Interest Group on Math. Programming, University of Wisconsin, Madison, WI, 1980), Academic Press, New York, 1981, pp. 167–199.

[31] E. D. Dolan and J. J. Moré, *Benchmarking Optimization Software with COPS*, Technical Memorandum ANL/MCS-TM-246, Argonne National Laboratory, Argonne, IL, 2000.
[32] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.
[33] E. D. Dolan, J. J. Moré, and T. S. Munson, *Benchmarking Optimization Software with COPS* 3.0, Technical Memorandum ANL/MCS-TM-273, Argonne National Laboratory, Argonne, IL, 2004.
[34] A. Drud, *CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems*, Math. Program., 31 (1985), pp. 153–191.
[35] I. S. Duff, *MA28—A Set of Fortran Subroutines for Sparse Unsymmetric Linear Equations*, Report AERE R8730, Atomic Energy Research Establishment, Harwell, England, 1977.
[36] S. K. Eldersveld, *Large-Scale Sequential Quadratic Programming Algorithms*, Ph.D. thesis, Department of Operations Research, Stanford University, Stanford, CA, 1991.
[37] S. I. Feldman, D. M. Gay, M. W. Maimone, and N. L. Schryer, *A Fortran-to-C Converter*, Computing Science Technical Report 149, AT&T Bell Laboratories, Murray Hill, NJ, 1990.
[38] 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., SIAM, Philadelphia, 1985, pp. 26–40.
[39] R. Fletcher, *Practical Methods of Optimization*, 2nd ed., John Wiley and Sons, Chichester, England, New York, 1987.
[40] R. Fletcher and S. Leyffer, *User Manual for FilterSQP*, Technical Report NA/181, Department of Mathematics, University of Dundee, Scotland, 1998.
[41] R. Fletcher and S. Leyffer, *Nonlinear programming without a penalty function*, Math. Program., 91 (2002), pp. 239–269.
[42] R. Fourer, *A simplex algorithm for piecewise-linear programming.* I. *Derivation and proof*, Math. Program., 33 (1985), pp. 204–233.
[43] R. Fourer, *A simplex algorithm for piecewise-linear programming.* II. *Finiteness, feasibility and degeneracy*, Math. Program., 41 (1988), pp. 281–315.
[44] R. Fourer, *A simplex algorithm for piecewise-linear programming.* III. *Computational analysis and applications*, Math. Program., 53 (1992), pp. 213–235.
[45] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, The Scientific Press, San Francisco, CA, 1993.
[46] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Brooks/Cole—Thomson Learning, Pacific Grove, CA, 2003.
[47] M. P. Friedlander and M. A. Saunders, *A globally convergent linearly constrained Lagrangian method for nonlinear optimization*, SIAM J. Optim., to appear.
[48] GAMS Development Corp., http://www.gams.com.
[49] J. C. Gilbert and C. Lemaréchal, *Some numerical experiments with variable-storage quasi-Newton algorithms*, Math. Program., 45 (1989), pp. 407–435.
[50] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, *Methods for modifying matrix factorizations*, Math. Comput., 28 (1974), pp. 505–535.
[51] P. E. Gill and M. W. Leonard, *Limited-memory reduced-Hessian methods for large-scale unconstrained optimization*, SIAM J. Optim., 14 (2003), pp. 380–401.
[52] P. E. Gill and W. Murray, *The computation of Lagrange multiplier estimates for constrained minimization*, Math. Program., 17 (1979), pp. 32–60.
[53] P. E. Gill, W. Murray, and M. A. Saunders, *User's Guide for SQOPT* 5.3: *A Fortran Package for Large-Scale Linear and Quadratic Programming*, Numerical Analysis Report NA 97-4, Department of Mathematics, University of California, San Diego, La Jolla, CA, 1997.
[54] P. E. Gill, W. Murray, and M. A. Saunders, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM J. Optim., 12 (2002), pp. 979–1006.
[55] P. E. Gill, W. Murray, and M. A. Saunders, *User's Guide for SNOPT* 7.1: *A Fortran Package for Large-Scale Nonlinear Programming*, Numerical Analysis Report NA 04-1, Department of Mathematics, University of California, San Diego, La Jolla, CA, 2004.
[56] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *Sparse matrix methods in optimization*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 562–589.
[57] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *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.
[58] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *Maintaining LU factors of a general sparse matrix*, Linear Algebra Appl., 88/89 (1987), pp. 239–270.
[59] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *Inertia-controlling methods for general quadratic programming*, SIAM Rev., 33 (1991), pp. 1–36.

[60] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *Some theoretical properties of an augmented Lagrangian merit function*, in Advances in Optimization and Parallel Computing, P. M. Pardalos, ed., North–Holland, Amsterdam, 1992, pp. 101–128.
[61] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, New York, 1981.
[62] D. Goldfarb, *Factorized variable metric methods for unconstrained optimization*, Math. Comput., 30 (1976), pp. 796–811.
[63] N. I. M. Gould, D. Orban, and Ph. L. Toint, *CUTEr and SifDec: A constrained and unconstrained testing environment, revisited*, ACM Trans. Math. Software, 29 (2003), pp. 373–394.
[64] N. I. M. Gould, D. Orban, and Ph. L. Toint, *GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization*, ACM Trans. Math. Software, 29 (2003), pp. 353–372.
[65] N. I. M. Gould and Ph. L. Toint, *SQP methods for large-scale nonlinear programming*, in System Modelling and Optimization (Cambridge, 1999), Kluwer Academic, Boston, MA, 2000, pp. 149–178.
[66] S. P. Han, *Superlinearly convergent variable metric algorithms for general nonlinear programming problems*, Math. Program., 11 (1976), pp. 263–282.
[67] C. R. Hargraves and S. W. Paris, *Direct trajectory optimization using nonlinear programming and collocation*, J. Guidance, Control, and Dynamics, 10 (1987), pp. 338–348.
[68] N. Higham, *FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation*, ACM Trans. Math. Software, 14 (1988), pp. 381–396.
[69] M. Lalee, J. Nocedal, and T. Plantenga, *On the implementation of an algorithm for large-scale equality constrained optimization*, SIAM J. Optim., 8 (1998), pp. 682–706.
[70] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Math. Program., 45 (1989), pp. 503–528.
[71] J. L. Morales, *A numerical study of limited memory BFGS methods*, Appl. Math. Lett., 15 (2002), pp. 481–487.
[72] J. L. Morales and J. Nocedal, *Automatic preconditioning by limited memory quasi-Newton updating*, SIAM J. Optim., 10 (2000), pp. 1079–1096.
[73] W. Murray, *Sequential quadratic programming methods for large-scale problems*, J. Comput. Optim. Appl., 7 (1997), pp. 127–142.
[74] 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.
[75] B. A. Murtagh and M. A. Saunders, *Large-scale linearly constrained optimization*, Math. Program., 14 (1978), pp. 41–72.
[76] B. A. Murtagh and M. A. Saunders, *A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints*, Math. Program., 16 (1982), pp. 84–117.
[77] B. A. Murtagh and M. A. Saunders, *MINOS* 5.5 *User's Guide*, Report SOL 83-20R, Department of Operations Research, Stanford University, Stanford, CA, 1998 (revised).
[78] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.
[79] E. O. Omojokun, *Trust Region Algorithms for Nonlinear Equality and Inequality Constraints*, Ph.D. thesis, Department of Computer Science, University of Colorado, Boulder, CO, 1989.
[80] M. J. O'Sullivan, *New Methods for Dynamic Programming over an Infinite Time Horizon*, Ph.D. thesis, Department of Management Science and Engineering, Stanford University, Stanford, CA, 2002.
[81] M. J. O'Sullivan and M. A. Saunders, *Sparse Rank-Revealing LU Factorization*. Presented at the 7th SIAM Conference on Optimization, Toronto, Canada, 2002; available online from http://www.stanford.edu/group/SOL/talks.html.
[82] M. J. O'Sullivan and M. A. Saunders, *Sparse Rank-Revealing LU Factorization (via Threshold Complete Pivoting and Threshold Rook Pivoting)*. Presented at Householder Symposium XV on Numerical Linear Algebra, Peebles, Scotland, 2002; available online from http://www.stanford.edu/group/SOL/talks.html.
[83] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.
[84] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.
[85] M. J. D. Powell, *A Fast Algorithm for Nonlinearly Constrained Optimization Calculations*, Technical Report 77/NA 2, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England, 1977.
[86] M. J. D. Powell, *Algorithms for nonlinear constraints that use Lagrangian functions*, Math. Program., 14 (1978), pp. 224–248.

[87] M. J. D. Powell, *The convergence of variable metric methods for nonlinearly constrained optimization calculations*, in Nonlinear Programming, 3 (Proc. Sympos., Special Interest Group Math. Programming, University of Wisconsin, Madison, WI, 1977), Academic Press, New York, 1978, pp. 27–63.

[88] J. K. Reid, *Fortran Subroutines for Handling Sparse Linear Programming Bases*, Report AERE R8269, Atomic Energy Research Establishment, Harwell, England, 1976.

[89] J. K. Reid, *A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases*, Math. Program., 24 (1982), pp. 55–69.

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

[91] R. W. H. Sargent and M. Ding, *A new SQP algorithm for large-scale nonlinear programming*, SIAM J. Optim., 11 (2000), pp. 716–747.

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

[93] P. Spellucci, *Han's method without solving QP*, in Optimization and Optimal Control (Proc. Conf., Math. Res. Inst., Oberwolfach, 1980), Springer-Verlag, Berlin, 1981, pp. 123–141.

[94] P. Spellucci, *A new technique for inconsistent QP problems in the SQP method*, Math. Methods Oper. Res., 47 (1998), pp. 355–400.

[95] P. Spellucci, *An SQP method for general nonlinear programs using only equality constrained subproblems*, Math. Program., 82 (1998), pp. 413–448.

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

[97] 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.

[98] K. Tone, *Revisions of constraint approximations in the successive QP method for nonlinear programming problems*, Math. Program., 26 (1983), pp. 144–152.

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

[100] R. J. Vanderbei and D. F. Shanno, *An interior-point algorithm for nonconvex nonlinear programming*, Comput. Optim. Appl., 13 (1999), pp. 231–252.

[101] A. Wächter, L. T. Biegler, Y.-D. Lang, and A. Raghunathan, *IPOPT: An Interior Point Algorithm for Large-Scale Nonlinear Optimization*, http://www.coin-or.org, 2002.

[102] R. B. Wilson, *A Simplicial Method for Convex Programming*, Ph.D. thesis, Harvard University, 1963.

[103] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, *Algorithm* 778: *L-BFGS-B—Fortran subroutines for large-scale bound constrained optimization*, ACM Trans. Math. Software, 23 (1997), pp. 550–560.