SIAM J. OPTIM.
Vol. 12, No. 4, pp. 979–1006
© 2002 Society for Industrial and Applied Mathematics

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

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

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

We discuss an SQP algorithm that uses a smooth augmented Lagrangian merit function and makes explicit provision for infeasibility in the original problem and the QP subproblems. SNOPT is a particular implementation that makes use of a semidefinite QP solver. It is based on a limited-memory quasi-Newton approximation to the Hessian of the Lagrangian and uses a reduced-Hessian algorithm (SQOPT) for solving the QP subproblems. It is designed for problems with many thousands of constraints and variables but a moderate number of degrees of freedom (say, up to 2000). An important application is to trajectory optimization in the aerospace industry. Numerical results are given for most problems in the CUTE and COPS test collections (about 900 examples).

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

PII. S1052623499350013

1. Introduction.

We present a sequential quadratic programming (SQP) method for large-scale optimization problems involving general linear and nonlinear constraints. SQP methods have proved reliable and efficient for many such problems. For example, under mild conditions the general-purpose solvers NLPQL [70], NPSOL [44, 47], and DONLP [73] 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 [58, 59, 60] and CONOPT [26].

1.1. The optimization problem.

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

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

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

*Received by the editors January 11, 1999; accepted for publication (in revised form) November 15, 2001; published electronically April 19, 2002.
http://www.siam.org/journals/siopt/12-4/35001.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, and DMS-9973276.

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

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). For the present implementation we further assume that the number of active constraints at a solution is reasonably close to n. In other words, the number of degrees of freedom is not too large (say, less than 2000).

Important examples are control problems such as those arising in optimal trajectory calculations. For many years, the optimal trajectory system OTIS (Hargraves and Paris [51]) 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 (quadratic programming) 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 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. The implementation is called SNOPT (sparse nonlinear optimizer) [41]. Extensive numerical results are given in section 6.

The method of SNOPT exploits sparsity in the constraint Jacobian and maintains a limited-memory quasi-Newton approximation to H_k (not a full transformed Hessian $Q_k^T H_k Q_k$). 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 that allows for variables to appear 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, and early termination of the QP subproblems.

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

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

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 [11]). Elastic programming has long been a feature of the XS system of Brown and Graves [12]. Other algorithms based on minimizing one-norms of infeasibilities are given by Conn [21] and Bartels [1].

If the linear constraints are infeasible ($v \neq 0$ or $w \neq 0$), SNOPT terminates without computing the nonlinear functions. Otherwise, all subsequent iterates satisfy the linear constraints. (Sometimes this feature helps ensure that the functions and gradients are well defined; see section 5.2.)

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 [76] and is fundamental to the Sℓ_1QP algorithm of Fletcher [30]. See also Conn [20] and Spellucci [72]. 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.

1.3. Other work on large-scale SQP. 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 [53], related to the trust-region method of Byrd [15] and Omojokun [61], uses either the exact Lagrangian Hessian or a limited-memory quasi-Newton approximation defined by the method of Zhu et al. [79]. The method of Biegler, Nocedal, and Schmid [3] is in the class of *reduced-Hessian methods*, which maintain a dense approximation to the reduced Hessian, using quasi-Newton updates.

For large problems with general inequality constraints as in problem (NP), SQP methods have been proposed by Eldersveld [28], Tjoa and Biegler [75], Fletcher and Leyffer [32], and Betts and Frank [2]. 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 7). 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 and can accommodate many degrees of freedom. Examples are Boggs, Kearsley, and Tolle [4, 5] and Sargent and Ding [69].

1.4. Other large-scale methods. MINOS and CONOPT are both reduced-Hessian methods. Like SNOPT, they use first derivatives and are designed for large problems with few degrees of freedom (again up to 2000, say, although MINOS can allow for any number; see section 7.1). 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 maintains near-feasibility with respect to the nonlinear constraints, again at the expense of many function evaluations. SNOPT is likely to outperform MINOS and CONOPT when the functions (and their derivatives) are expensive to evaluate. Relative to MINOS, an added advantage is the existence of a merit function to ensure global convergence. This is especially important when the constraints are highly nonlinear.

LANCELOT Release A [22] is another widely used package in the area of large-scale constrained optimization. It uses a *bound constrained augmented Lagrangian* method. In general, LANCELOT is recommended for large problems with many degrees of freedom. It complements SNOPT and the other methods discussed above. A comparison between LANCELOT and MINOS has been made in [8, 9].

LOQO [78] and KNITRO [17, 16] are examples of large-scale optimization packages that treat inequality constraints by a primal-dual interior method. Both packages require second derivatives but can accommodate many degrees of freedom.

1.5. Notation. Some important quantities follow:

(x, π, s)	primal, dual and slack variables for problem (GNP) (see 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$_k$) (see 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

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

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

We assume that a 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.

For an overview of SQP methods, see, for example, Boggs and Tolle [6], Fletcher [31], Gill, Murray, and Wright [48], Murray [56], and Powell [66].

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*:

$$c_L(x, x_k) = c_k + J_k(x - x_k),$$
$$d_L(x, x_k) = c(x) - c_L(x, x_k);$$

see Robinson [68] and Van der Hoek [77]. The first and second derivatives of the modified Lagrangian with respect to x are

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

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 QP

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

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

(GQP$_k$)	$\underset{x}{\text{minimize}}$	$f_k + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T H_k(x - x_k)$
	subject to	$c_k + J_k(x - x_k) \geq 0.$

Optimality conditions for (GQP$_k$) may be written as

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

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 [28] (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* $Z_k^T g$.

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

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

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

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

gives a *sufficient decrease* in the merit function (2.3). Let $\varphi_k(\alpha)$ denote the merit function computed at the point $(x_k + \alpha(\widehat{x}_k - x_k), \pi_k + \alpha(\widehat{\pi}_k - \pi_k), s_k + \alpha(\widehat{s}_k - s_k))$; i.e., $\varphi_k(\alpha)$ defines $\mathcal{M}$ as a univariate function of the step length. Initially D is zero (for $k = 0$). When necessary, the penalties in D are increased by the minimum-norm perturbation that ensures *sufficient descent* for $\varphi_k(\alpha)$ [47]. (Note: As in NPSOL, s_{k+1} in (2.4) is redefined to minimize the merit function as a function of s, prior to the solution of (GQP_{k+1}). For more details, see [44, 28].)

In the line search, for some vector $b > 0$ the following condition is enforced:

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

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 [71].) Murray and Prieto [57] 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.

2.8. The approximate Hessian. As suggested by Powell [64], 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., [40]). 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

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

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

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

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

First modification. 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 [18] 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 [65], 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 [59]:

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

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

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

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 $w = d_L(x_{k+1}, x_k)$. The optimal ω can be computed analytically [44, 28]. 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 [50] and Tapia [74]. However, we emphasize that the second modification is not required in the neighborhood of a solution, because as $x \to x^*$, $\nabla^2\mathcal{L}_A$ converges to $\nabla^2\mathcal{L}$, and the first modification will already have been successful.

2.10. Convergence tests. A point (x, π) is regarded as a satisfactory solution if it satisfies the first-order optimality conditions (2.1) to within certain tolerances. Let τ_P and τ_D be specified small positive constants, and define $\tau_x = \tau_P(1 + \|x\|)$, $\tau_\pi = \tau_D(1 + \|\pi\|)$. 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

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

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 $\|\pi_k\|$ exceeds some value γ_1 (or if (GQP$_k$) is infeasible). We use

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

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 2.9 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 5.4).

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

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

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

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

On completion of iteration $k = r + \ell$, the diagonals of H_k are computed from (3.2) and saved to form the next positive-definite H_r (with $r = k + 1$). Storage is then "reset" by discarding the previous updates. (Similar schemes are described by Buckley and LeNir [13, 14] and Gilbert and Lemaréchal [37]. More elaborate schemes are given by Liu and Nocedal [54], Byrd, Nocedal, and Schnabel [19], and Gill and Leonard [39], and some have been evaluated by Morales [55]. However, as already indicated, these schemes would require refactorization of the reduced Hessian in the linearly constrained case.)

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

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

At present, (QP_k) is solved by the package SQOPT [42], 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 [33, 34, 35]).

SNOPT uses elastic bounds in three different ways:

- to solve problem (FLP) (section 1.2) if the linear constraints are infeasible,
- to solve problem (PP1) (section 5.1),
- to solve the QP subproblems associated with problem (NP(γ)) after nonlinear elastic mode is initiated.

4.2. The null-space method. SQOPT maintains a dense Cholesky factorization of the QP reduced Hessian:

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

where Z is the null-space matrix for the working sets W in the QP minor iterations. Normally, R is computed from (4.1) when the 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 $\bar{n} = n$.

To review notation, Z is maintained in "reduced-gradient" form as in MINOS, using the package LUSOL [45] to maintain sparse LU factors of a square matrix B whose columns change as the working set W changes:

$$W = \begin{pmatrix} B & S & N \\ & & I \end{pmatrix} P, \qquad Z = P^T \begin{pmatrix} -B^{-1}S \\ I \\ 0 \end{pmatrix}, \tag{4.2}$$

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 (the column dimension of S). Products of the form Zv and $Z^T g$ are obtained by solving with B or B^T.

4.3. Threshold pivoting (TPP and TCP). Stability in LU factorization is achieved by bounding the off-diagonal elements of L or U. There are many ways to do this, especially in the sparse case. In LUSOL, L has unit diagonals, and each elimination step produces the next column of L and the next row of U. Let

$$\begin{aligned} \tau_L &= \text{the } LU \textit{ factor tolerance} \text{ such that } |L_{ij}| \leq \tau_L \\ &\quad (\text{where } 1 < \tau_L \leq 100, \text{ say}), \\ A_l &= \text{the remaining submatrix to be factored after } l \text{ steps} \\ &\quad (\text{updated by the first } l \text{ columns of } L). \end{aligned}$$

For most factorizations, LUSOL uses a *threshold partial pivoting* strategy (TPP) similar to that in LA05 [67] and MA28 [27]. To become the next diagonal of U, a nonzero in A_l must be sufficiently large compared to *other nonzeros in the same column of* A_l.

With $\tau_L \in [4, 25]$, TPP usually performs well in terms of balancing stability and sparsity, but is not especially good at rank-detection (revealing near-singularity and its cause). For example, a triangular matrix A gives $L = I$ and $U = A$ for all values of τ_L (a perfect L and maximum sparsity, but little hint of possible ill-conditioning).

For greater reliability, a *threshold complete pivoting* strategy (TCP) has been implemented recently in LUSOL [63], in which the next diagonal of U must be reasonably large compared to *all nonzeros in* A_l. The original aim was to improve rank-detection for the sparse matrices arising during the optimization of Markov decision chains [62]. Although reduced sparsity and speed are expected, TCP has proved valuable within SNOPT, as described below.

In general we use TPP where possible, with τ_L decreasing through a short sequence of values (currently 4, 2, $\sqrt{2}$, ..., 1.1) if various tests continue to indicate instability (e.g., large $\|b - Bx\|$ or $\|x\|$ when basic variables are recomputed from $Bx = b$). When necessary, a switch is made to TCP with another sequence of values (currently $\tau_L = 20, 10, 5, 2.5, \sqrt{2.5}, \ldots, 1.1$).

4.4. 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." Extreme behavior of this kind was exhibited by one of the CUTE problems (section 6.2) when the first basis was factored with the normal partial pivoting options. Problem *drcavty2* is a large square system of nonlinear equations (10000 constraints and variables, 140000 Jacobian nonzeros). The first TPP factorization with $\tau_L = 4.0$ indicated 243 singularities. After slacks were inserted, the next factorization indicated 47 additional singularities, the next a further 25, then 18, 14, 10, and so on. Nearly 30 TPP factorizations and 460 new slacks were required before the basis was regarded as suitably nonsingular. Since L and U each had about a million nonzeros in all factorizations, the repeated failures were rather expensive.

In contrast, a single TCP factorization with $\tau_L = 2.5$ indicated 100 singularities, after which the modified B proved to be very well-conditioned. Although L and U were more dense (1.35 million nonzeros each) and much more expensive to compute, the subsequent optimization required significantly fewer major and minor iterations.

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.

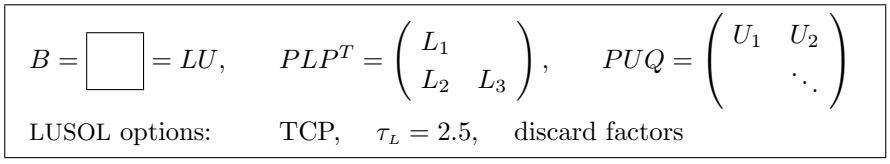

FIG. 1. *BR factorization (rank detection for square B).*

BR factorization is the primary recourse when unexpected growth occurs in $\|x\|$ following solution of $Bx = b$. It has proved valuable for some other CUTE problems arising from partial differential equations (namely, *porous1*, *porous2*, *bratu2d*, and *bratu3d*). A regular "marching pattern" is sometimes present in B, particularly in the first *triangular* basis following a cold start. With partial pivoting, 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., [52]), we are using LUSOL's complete pivoting option to decide *which columns* are causing the poor condition.

4.5. Basis repair (rectangular case). When superbasic variables are present, the permutation P in (4.2) 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.

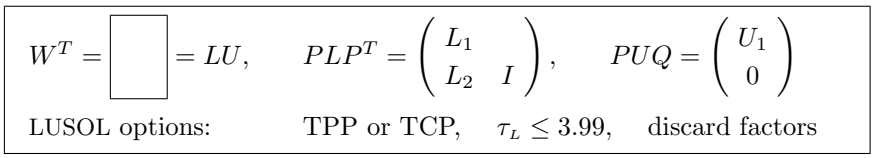

FIG. 2. *BS factorization (basis detection for rectangular W).*

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

$$Z \equiv P^T \begin{pmatrix} -B^{-1}S \\ I \end{pmatrix} = P^T \begin{pmatrix} -L_1^{-T} L_2^T \\ I \end{pmatrix}. \tag{4.3}$$

With $\tau_L \leq 3.99$, 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 (4.3) 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.)

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 (see (4.1)) is allowed to be zero. (See [46] 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

$$Z^T H_k Z d_Z = 0 \quad \text{and} \quad g^T Z d_Z < 0, \tag{4.4}$$

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

4.7. 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 (4.4) that if we choose $p = d$, then

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

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

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

5.1. 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.

5.2. Undefined functions. If the constraints in (PP1) or (PP2) prove to be infeasible, SNOPT solves problem (FLP) (see section 1.2) 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.

5.3. 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 [57] 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 (Dembo, Eisenstat, and Steihaug [23]).

5.4. 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 $Q^T H_k Q$:

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

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

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

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

(see Goldfarb [49], Dennis and Schnabel [24]). This rank-one modification of R_Q could be restored to upper-triangular form by applying two sequences of plane rotations from the left [38].

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 (5.1):

1. Compute $q = H_k \delta_k$ and $t = Z^T q$.
2. Define $\phi = \|w\|_2 = (\delta_k^T H_k \delta_k)^{1/2} = (q^T \delta_k)^{1/2}$.

3. Solve $R^T w_z = t$.
4. Define $u_z = w_z/\phi$ 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 an upper 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:

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

5.5. Summary of the SQP algorithm. The main steps of the SNOPT algorithm follow. We assume that a starting point $(x_0,\ \pi_0)$ 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 2.10.

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 $k = 0$, and evaluate functions and gradients at x_0.
1. Factorize W_k.
2. Find $\bar{x}_k$, a feasible point for the QP subproblem. (This is an intermediate point for the QP solver, which also provides a working-set matrix $\bar{W}_k$ and its null-space matrix $\bar{Z}_k$.) If no feasible point exists, initiate elastic mode and restart the QP.
3. Form the reduced Hessian $\bar{Z}_k^T H_k \bar{Z}_k$, and compute its Cholesky factorization.
4. Continue solving the QP subproblem to find $(\widehat{x}_k, \widehat{\pi}_k)$, an optimal QP solution. (This provides a working-set matrix $\widehat{W}_k$ and its null-space matrix $\widehat{Z}_k$.)
   If elastic mode has not been initiated but $\|\widehat{\pi}_k\|_\infty$ is "large," enter elastic mode and restart the QP.
   If the QP is unbounded and x_k satisfies the nonlinear constraints, declare the problem unbounded (f is unbounded below in the feasible region). Otherwise (if the QP is unbounded), go to Step 6 (f is unbounded below in the feasible region if a feasible point exists).
5. If (x_k, π_k) satisfies the convergence tests for (NP) analogous to (2.9), declare the solution optimal. If similar convergence tests are satisfied for (NP(γ)), go to Step 6. Otherwise, go to Step 7.
6. If elastic mode has not been initiated, enter elastic mode and repeat Step 4. Otherwise, if γ has not reached its maximum value, increase γ and repeat Step 4. Otherwise, declare the problem infeasible.
7. Find a step length α_k that gives a sufficient reduction in the merit function. Set $x_{k+1} = x_k + \alpha_k(\widehat{x}_k - x_k)$ and $\pi_{k+1} = \pi_k + \alpha_k(\widehat{\pi}_k - \pi_k)$. In the process, evaluate functions and gradients at x_{k+1}.
8. Define $\delta_k = x_{k+1} - x_k$ and $y_k = \nabla\mathcal{L}(x_{k+1}, x_k, \pi_{k+1}) - \nabla\mathcal{L}(x_k, x_k, \pi_{k+1})$. If $y_k^T\delta_k < \sigma_k$ (see (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.

9. If $y_k^T\delta_k \geq \sigma_k$, apply the BFGS update to H_k, using the pair $(H_k\delta_k, y_k)$.
10. 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 3. Steps 2 and 4 may also involve significant work if the QP subproblem requires many minor iterations. Typically this will happen only during the early major iterations.

6. Numerical results. SNOPT and SQOPT implement all of the techniques described in sections 2–4. 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 $f2c$ [29] (though these features are not used here).

We give the results of applying SNOPT 6.1 of May, 2001, to problems in the CUTE and COPS 2.0 test collections [10, 7, 25]. Function and gradient values were used throughout (but not second derivatives).

All runs were made on an SGI Octane workstation with 512MB of RAM and two 250MHz R10000 processors (only one being used for each problem solution). The f90 compiler was used with `-n32 -O` options specifying 32-bit addressing and full code optimization. The floating-point precision was 2.22×10^{-16}. Table 1 defines the notation used in the tables of results.

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).
a	Almost optimal (within 10^{-2} of satisfying the convergence test).
c	Final nonoptimal point could not be improved.
s	User-defined superbasics limit exceeded.

6.1. Parameters for SNOPT. Figure 3 gives the SNOPT optional parameters 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).

The `Major feasibility` and `optimality` tolerances set τ_P and τ_D in section 2.10 for problem (NP). The `Minor` tolerances are analogous parameters 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).

`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.

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 H_k reset to the current Hessian diagonal every 20 major iterations.

6.2. Results on the CUTE test set. The CUTE distribution of 01/May/2001 contains 945 problems in standard interface format (SIF). A list of the CUTE problem

types and their frequency 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 CUTE distribution. This gave problems ranging in size from *hs1*, with two variables and no constraints, to *cont5-qp*, with 40601 variables and 40201 constraints.

```
BEGIN SNOPT Problem
   Minimize
   Crash option                      3
   Derivative level                  3
   Elastic weight                1.0E+4
   Hessian updates                  20
   Superbasics limit              2000
   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
   Proximal point method             1
   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 optional parameter file.*

TABLE 2
The 945 CUTE problems listed by type and frequency.

Frequency	Type	Characteristics
24	LP	Linear obj, linear constraints
116	QP	Quadratic obj, linear constraints
160	UC	Nonlinear obj, no constraints
125	BC	Nonlinear obj, bound constraints
70	LC	Nonlinear obj, linear constraints
375	NC	Nonlinear obj, nonlinear constraints
75	FP	No objective

From the complete set of 945 problems, 74 were omitted as follows:

- 6 nonsmooth problems (*bigbank*, *gridgena*, *hs87*, *net1*, *net2* and *net3*),
- 57 problems with more than 2000 degrees of freedom at the solution (*aug3d*, *aug3dc*, *aug3dcqp*, *dixmaanb*, *dtoc5*, *dtoc6*, *jannson3*, *jannson4*, *jimack*, *jnlbrng1*, *jnlbrng2*, *jnlbrnga*, *minsurfo*, *obstclae*, *obstclbm*, *odnamur*, *orthrdm2*, *orthrgdm*, *stcqp1*, *stnqp1*, *torsion6*, and the 36 *lukvli* and *lukvle1* problems),
- 9 problems with undefined variables or floating-point exceptions in the SIF file (*himmelbj*, *lhaifam*, *lin*, *pfit1*, *pfit3*, *recipe*, *robotarm*, *s365mod*, and *scon1dls*),

- 2 problems too large to decode (*qpband* and *qpnband*),
- 1 problem with excessively low accuracy in the objective gradients (*bleachng*). Requesting greater accuracy leads to excessive evaluation time.

SNOPT was applied to the remaining 870 problems, using the options listed in Figure 3. No special information was used in the case of LP, QP, and FP problems—i.e., each problem was assumed to have a general nonlinear objective. The results are summarized in Table 3.

TABLE 3
Summary: SNOPT on the smooth CUTE problems.

Problems attempted	870
Optimal	794
Unbounded	3
Infeasible	10
Optimal, low accuracy	11
Cannot be improved	7
False infeasibility	17
Terminated	28
Major iterations	108980
Minor iterations	678524
Function evaluations	153867
Cpu time (secs)	70864.7

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

A total of 11 problems (*allinitc*, *eigmaxc*, *eigminc*, *hs268*, *mancino*, *marine*, *orthrds2*, *orthregd*, *penalty3*, *pinene*, and *s268*) were terminated at a point that satisfied either the feasibility *or* the optimality test and was within 10^{-2} of satisfying the other test. AMPL implementations of *marine* and *pinene* were solved successfully as part of the COPS 2.0 collection (see section 6.3).

SNOPT reported 22 problems (*argauss*, *bratu2dt*, *cont6-qq*, *drcavty2*, *eigenb*, *eigmaxb*, *fletcher*, *flosp2th*, *growth*, *hadamard*, *heart6*, *himmelbd*, *hs90*, *junkturn*, *lewispol*, *lootsma*, *lubrif*, *lubrifc*, *nystrom5*, *optcdeg3*, *powellsq*, *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 22 problems, all but five cases must be counted as failures because they are known to have feasible points. The five exceptions, *flosp2th*, *junkturn*, *lewispol*, *lubrif*, and *nystrom5*, 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 all five 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. Problem *drcavty2* is also

listed as a failure, although it is probably infeasible for the size of problem tested (196 variables, 101 general constraints). SNOPT ran successfully on the larger versions of the problem (the largest having 10816 variables and 10001 general constraints).

SNOPT was unable to solve 28 cases within the allotted 2000 major iterations (*biggsb1*, *bqpgauss*, *catena*, *chainwoo*, *chenhark*, *curly10*, *curly20*, *curly30*, *drcav1lq*, *drcav2lq*, *drcav3lq*, *eigenbls*, *eigencls*, *hydc20ls*, *noncvxu2*, *noncvxun*, *palmer5b*, *palmer5e*, *palmer7a*, *palmer7e*, *qr3dls*, *sbrybnd*, *scosine*, *scurly10*, *scurly20*, *scurly30*, *sparsine*, and *vibrbeam*). Another 7 problems could not be improved at a nonoptimal point: *brownbs*, *catena*, *glider*, *meyer3*, *nuffield*, *vanderm1*, and *vanderm2*. SNOPT essentially found the solution of the badly scaled problems *brownbs* and *meyer3* but was unable to declare optimality. An AMPL implementation of *glider* was solved successfully (see section 6.3)

If the infeasible LC problems, the unbounded problems, and the 5 (conjectured) infeasible problems are counted as successes, SNOPT solved a grand total of 807 of the 870 problems attempted. In another 11 cases, SNOPT found a point that was within a factor 10^{-2} of satisfying the convergence test. These results provide strong 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.

6.3. Results on the COPS 2.0 test set. Tests on the 17 problems in the COPS 2.0 collection were made using the AMPL modeling system [36]. When necessary, the AMPL model and data files were modified to increase the problem size to be the largest considered in [7] (see Table 4).

TABLE 4
Dimensions of the AMPL versions of the COPS problems.

No.	Problem	Type	Variables	Constraints		
				Linear	Nonlinear	Total
1	*bearing*	BC	5000	0	0	0
2	*camshape*	NC	800	800	801	1601
3	*catmix*	NC	2401	1	1600	1601
4	*chain*	NC	800	401	1	402
5	*channel*	FP	3198	1598	1600	3198
6	*elec*	NC	600	1	200	201
7	*gasoil*	NC	4001	799	3200	3999
8	*glider*	NC	1999	1	1600	1601
9	*marine*	NC	4815	1593	3200	4793
10	*methanol*	NC	4802	1198	3600	4798
11	*minsurf*	BC	5000	0	0	0
12	*pinene*	NC	4000	996	3000	3996
13	*polygon*	NC	198	99	4950	5048
14	*robot*	NC	3599	2	2400	2402
15	*rocket*	NC	1601	0	1200	1201
16	*steering*	NC	2000	2	1600	1602
17	*torsion*	BC	5000	0	0	0

The bound constrained problems *bearing*, *minsurf*, and *torsion* have more than 2000 degrees of freedom at the solution, but were tested anyway. (SNOPT is not appropriate for problems with only bound constraints unless many of the bounds are active.) Table 5 gives results obtained by applying SNOPT with the options listed in Figure 3. The default AMPL options (including problem preprocessing) were used in each case.

TABLE 5
SNOPT on the COPS 2.0 *problems.*

No.	Problem	Mnr	Mjr	Fcn	Obj	Con	n_Z	cpu
1	*bearing*^{s}	2279	19	23	1.147002E+01	0.0E+00	2000	175.0
2	*camshape*	3019	9	18	4.222963E+00	9.4E-08	6	5.5
3	*catmix*	594	11	14	-4.796022E-02	2.8E-07	395	14.0
4	*chain*	839	40	44	5.068630E+00	4.2E-06	399	24.6
5	*channel*	2192	5	7	1.000000E+00	3.2E-05	0	18.8
6	*elec*	731	326	354	1.843890E+04	4.6E-10	400	194.9
7	*gasoil*	2607	21	25	5.236596E-03	7.2E-08	3	32.5
8	*glider*	33959	516	785	1.247974E+03	5.3E-09	359	891.7
9	*marine*	5437	71	132	1.974653E+07	1.1E-11	22	144.7
10	*methanol*	6250	1381	8280	9.022290E-03	9.0E-10	4	1170.2
11	*minsurf*^{s}	3029	19	26	2.516317E+00	0.0E+00	2000	1251.5
12	*pinene*	3090	41	63	1.987216E+01	4.0E-13	5	51.0
13	*polygon*	3490	64	66	7.850233E-01	1.1E-08	98	51.0
14	*robot*	5855	28	51	9.141018E+00	2.1E-06	0	279.3
15	*rocket*	2663	8	16	1.005422E+00	1.3E-07	66	16.7
16	*steering*	764	29	35	5.545734E-01	7.6E-07	398	30.2
17	*torsion*^{s}	3112	16	20	-4.004933E-01	0.0E+00	2000	171.4

Discussion. SNOPT solved every COPS problem that has fewer than 2000 degrees of freedom at the solution. The default `New superbasics` limit (99) often improves efficiency, but for *bearing*, *minsurf*, and *torsion*, a larger value would reduce the time and major iterations needed to terminate with excess superbasics.

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

The COPS problems were also used to investigate the effect of the number ℓ of limited-memory updates on the performance of SNOPT. Table 6 gives times for the 14 nonlinearly constrained problems when solved with different choices for ℓ. In the case of the BC problems *bearing*, *minsurf*, and *torsion*, the principal effect of increasing ℓ is to increase the cost of the Hessian/vector products in the minor iterations needed to expand the reduced Hessian to its maximum size.

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 formulation of *glider* could not be solved.
- As ℓ is increased, the number of major iterations tends to decrease. However, numerical performance remains relatively stable. (For example, the same local solution was always found for the highly nonlinear problem *polygon*.)
- 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 reflected in the total computation time for Table 6 if it were not for *methanol*, whose time improves dramatically because of a better Hessian approximation.

The choice of default value $\ell = 20$ is intended to provide robustness without a significant computational penalty.

7. Extensions. Where possible, we have defined the SQP algorithm to be independent of the QP solver. Of course, implicit elastic bounds and certain "warm start" features are highly desirable. For example, SQOPT can use a given starting point and

TABLE 6
Number of LM updates vs. cpu time.

Problem	Limited-memory updates					
	5	10	15	20	25	30
camshape	5.6	5.4	5.6	5.5	5.4	5.4
catmix	7.7	14.3	14.5	14.0	15.2	15.0
chain	14.2	13.2	18.8	24.6	17.6	22.5
channel	19.1	18.8	19.0	18.8	19.0	18.7
elec	221.3	216.3	127.3	194.9	217.6	241.0
gasoil	34.1	32.8	32.0	32.5	32.2	31.8
glider	254.0[c]	845.5	429.2	891.7	369.6	595.0
marine	155.2	139.3	157.3[a]	144.7	163.4[a]	166.6
methanol	398.2	390.5	1218.2	1170.2	1253.0	501.1
pinene	42.4[a]	48.7	50.2	51.0	43.8	45.6
polygon	120.3	74.8	87.3	51.0	56.5	63.0
robot	215.7	248.4	275.9	279.3	277.2	274.9
rocket	16.4	16.2	16.5	16.7	15.8	16.0
steering	43.8	26.0	27.6	30.2	31.4	30.7
	1548.2	2090.2	2479.4	2924.7	2517.7	2027.4

TABLE 7
Number of LM updates vs. major iterations.

Problem	Limited-memory updates					
	5	10	15	20	25	30
camshape	9	9	9	9	9	9
catmix	7	11	11	11	11	11
chain	29	25	33	40	27	32
channel	5	5	5	5	5	5
elec	459	399	227	326	340	361
gasoil	26	23	20	21	21	21
glider	50[c]	513	224	516	173	275
marine	83	71	90[a]	71	84[a]	88
methanol	479	245	1224	1381	1469	604
pinene	30[a]	37	39	41	29	30
polygon	243	123	158	64	81	94
robot	22	23	28	28	28	28
rocket	8	8	8	8	8	8
steering	38	28	28	29	26	26
	1488	1520	2104	2550	2311	1592

working set, and for linearly constrained problems (section 5.4) it can accept a known Cholesky factor R for the reduced Hessian.

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

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

at each minor iteration, where W is the current working-set matrix. Reduced-Hessian methods such as SQOPT are efficient if W is nearly square and products $H_k x$ can be formed efficiently, but our aim is to accommodate many degrees of freedom.

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

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

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

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

where R_r is a dense triangle of specified size and D is diagonal. This structure partitions the superbasic variables into two sets. After a few minor iterations involving all superbasics (with quasi-Newton updates to R_r and D), the variables associated with D are temporarily frozen. Iterations proceed with updates to R_r only, and superlinear convergence can be expected within that subspace. A frozen superbasic variable is then interchanged with one from R_r, and the process is repeated.

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

Note that the SQP updates to H_k could be applied to R between major iterations as for the linear-constraint case (section 5.4). However, the quasi-Newton updates during the first few minor iterations of each QP should achieve a similar effect.

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

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

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

Following [43, section 3.6.2], if we define

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

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

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

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

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

As with interior-point methods, the most promising way to achieve efficiency with the linear algebra is to work with sparse second derivatives (i.e., an exact Hessian of the Lagrangian, or a sparse finite-difference approximation). However, indefinite QP subproblems raise many practical questions, and alternatives are needed when second derivatives are not available.

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

The present QP solver, SQOPT, maintains a dense reduced-Hessian factorization $Z^T H_k Z = R^T R$, where Z is obtained from a sparse LU factorization of part of the Jacobian. Efficiency improves with the number of constraints active at a solution; i.e., the number of degrees of freedom n_z should not be excessive. For the numerical tests we set a limit of 2000. This is adequate for many problem classes, such as control problems when the number of control variables is not excessive.

The numerical results of section 6 show that SNOPT is effective on most of the problems in the CUTE and COPS 2.0 test sets. Separate comparisons with MINOS have shown greater reliability as a result of the merit function and the "elastic variables" treatment of infeasibility, and much greater efficiency when function 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 will include the use of second derivatives (when available) and alternative QP solvers to allow for indefiniteness of the QP Hessian and many 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 also appreciate many suggestions from the referees and Associate Editor Jorge Nocedal.

REFERENCES

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

[2] J. T. Betts and P. D. Frank, *A sparse nonlinear optimization algorithm*, J. Optim. Theory Appl., 82 (1994), pp. 519–541.
[3] L. T. Biegler, J. Nocedal, and C. Schmid, *A reduced Hessian method for large-scale constrained optimization*, SIAM J. Optim., 5 (1995), pp. 314–347.
[4] P. T. Boggs, 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.
[5] 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.
[6] P. T. Boggs and J. W. Tolle, *Sequential quadratic programming*, in Acta Numerica, Cambridge University Press, Cambridge, England, 1995, pp. 1–51.
[7] 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, 1999.
[8] I. Bongartz, A. R. Conn, N. I. M. Gould, M. A. Saunders, and P. L. Toint, *A Numerical Comparison Between the LANCELOT and MINOS Packages for Large-Scale Constrained Optimization*, Report 97/13, Département de Mathématique, Facultés Universitaires de Namur, Namur, Belgium, 1997.
[9] I. Bongartz, A. R. Conn, N. I. M. Gould, M. A. Saunders, and P. L. Toint, *A Numerical Comparison Between the LANCELOT and MINOS Packages for Large-Scale Constrained Optimization: The Complete Numerical Results*, Report 97/14, Département de Mathématique, Facultés Universitaires de Namur, Namur, Belgium, 1997.
[10] I. Bongartz, A. R. Conn, N. I. M. Gould, and P. L. Toint, *CUTE: Constrained and unconstrained testing environment*, ACM Trans. Math. Software, 21 (1995), pp. 123–160.
[11] G. G. Brown and G. W. Graves, *Elastic Programming: A New Approach to Large-Scale Mixed-Integer Optimization*, presented at the ORSA/TIMS meeting, Las Vegas, NV, 1975.
[12] G. G. Brown and G. W. Graves, *The XS Mathematical Programming System*, working paper, Department of Operations Research, Naval Postgraduate School, Monterey, CA, 1975.
[13] A. Buckley and A. LeNir, *QN-like variable storage conjugate gradients*, Math. Programming, 27 (1983), pp. 155–175.
[14] A. Buckley and A. LeNir, *BBVSCG—A variable storage algorithm for function minimization*, ACM Trans. Math. Software, 11 (1985), pp. 103–119.
[15] R. H. Byrd, *Robust Trust-Region Methods for Contstrained Optimization*, presented at the SIAM Conference on Optimization, Houston, TX, 1987.
[16] 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.
[17] 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.
[18] R. H. Byrd and J. Nocedal, *An analysis of reduced Hessian methods for constrained optimization*, Math. Programming, 49 (1991), pp. 285–323.
[19] R. H. Byrd, J. Nocedal, and R. B. Schnabel, *Representations of quasi-Newton matrices and their use in limited-memory methods*, Math. Programming, 63 (1994), pp. 129–156.
[20] A. R. Conn, *Constrained optimization using a nondifferentiable penalty function*, SIAM J. Numer. Anal., 10 (1973), pp. 760–784.
[21] A. R. Conn, *Linear programming via a nondifferentiable penalty function*, SIAM J. Numer. Anal., 13 (1976), pp. 145–154.
[22] A. R. Conn, N. I. M. Gould, and P. L. Toint, *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*, Lecture Notes in Comput. Math. 17, Springer-Verlag, Berlin, Heidelberg, New York, London, Paris, Tokyo, 1992.
[23] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.
[24] J. E. Dennis, Jr., and R. B. Schnabel, *A new derivation of symmetric positive definite secant updates*, in Nonlinear Programming, 4 (1980), Academic Press, New York, 1981, pp. 167–199.
[25] E. Dolan and J. J. Moré, *Benchmarking optimization software with COPS*, Math. Program., 91 (2002), pp. 201–213.
[26] A. Drud, *CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems*, Math. Programming, 31 (1985), pp. 153–191.
[27] I. S. Duff, *MA28—A set of Fortran Subroutines for Sparse Unsymmetric Linear Equations*, Report AERE R8730, Atomic Energy Research Establishment, Harwell, England, 1977.
[28] S. K. Eldersveld, *Large-Scale Sequential Quadratic Programming Algorithms*, Ph.D. thesis, Department of Operations Research, Stanford University, Stanford, CA, 1991.
[29] 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.
[30] R. Fletcher, *An ℓ_1 penalty method for nonlinear constraints*, in Numerical Optimization, P. T. Boggs, R. H. Byrd, and R. B. Schnabel, eds., SIAM, Philadelphia, 1985, pp. 26–40.
[31] R. Fletcher, *Optimization*, 2nd ed., John Wiley and Sons, Chichester, New York, Brisbane, Toronto, Singapore, 1987.
[32] R. Fletcher and S. Leyffer, *Nonlinear programming without a penalty function*, Math. Program., 91 (2002), pp. 239–269.
[33] R. Fourer, *A simplex algorithm for piecewise-linear programming.* I. *Derivation and proof*, Math. Programming, 33 (1985), pp. 204–233.
[34] R. Fourer, *A simplex algorithm for piecewise-linear programming.* II. *Finiteness, feasibility and degeneracy*, Math. Programming, 41 (1988), pp. 281–315.
[35] R. Fourer, *A simplex algorithm for piecewise-linear programming.* III. *Computational analysis and applications*, Math. Programming, 53 (1992), pp. 213–235.
[36] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, The Scientific Press, San Francisco, 1993.
[37] J. C. Gilbert and C. Lemaréchal, *Some numerical experiments with variable-storage quasi-Newton algorithms*, Math. Programming, (1989), pp. 407–435.
[38] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, *Methods for modifying matrix factorizations*, Math. Comput., 28 (1974), pp. 505–535.
[39] P. E. Gill and M. W. Leonard, *Limited-Memory Reduced-Hessian Methods for Unconstrained Optimization*, Numerical Analysis Report NA 97-1, University of California, San Diego, La Jolla, CA, 1997.
[40] P. E. Gill and W. Murray, *The computation of Lagrange multiplier estimates for constrained minimization*, Math. Programming, 17 (1979), pp. 32–60.
[41] P. E. Gill, W. Murray, and M. A. Saunders, *User's Guide for SNOPT* 5.3*: A Fortran Package for Large-Scale Nonlinear Programming*, Numerical Analysis Report 97-5, Department of Mathematics, University of California, San Diego, La Jolla, CA, 1997.
[42] 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 97-4, Department of Mathematics, University of California, San Diego, La Jolla, CA, 1997.
[43] 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.
[44] 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.
[45] 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.
[46] 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.
[47] 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.
[48] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, New York, 1981.
[49] D. Goldfarb, *Factorized variable metric methods for unconstrained optimization*, Math. Comput., 30 (1976), pp. 796–811.
[50] S. P. Han, *Superlinearly convergent variable metric algorithms for general nonlinear programming problems*, Math. Programming, 11 (1976), pp. 263–282.
[51] 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.
[52] 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.
[53] 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.
[54] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Math. Programming, 45 (1989), pp. 503–528.
[55] J. L. Morales, *A numerical study of limited memory BFGS methods*, Appl. Math. Lett., to appear.
[56] W. Murray, *Sequential quadratic programming methods for large-scale problems*, J. Comput. Optim. Appl., 7 (1997), pp. 127–142.
[57] 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.

[58] B. A. Murtagh and M. A. Saunders, *Large-scale linearly constrained optimization*, Math. Programming, 14 (1978), pp. 41–72.

[59] B. A. Murtagh and M. A. Saunders, *A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints*, Math. Prog. Study, 16 (1982), pp. 84–117.

[60] B. A. Murtagh and M. A. Saunders, *MINOS* 5.4 *User's Guide*, Report SOL 83-20R, Department of Operations Research, Stanford University, Stanford, CA, 1995.

[61] 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.

[62] M. 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, 2001.

[63] M. O'Sullivan and M. A. Saunders, *Sparse LU Factorization with Threshold Complete Pivoting*, SOL Report, Department of Management Science and Engineering, Stanford University, Stanford, CA, to appear.

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

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

[66] M. J. D. Powell, *Variable metric methods for constrained optimization*, in Mathematical Programming: The State of the Art (Bonn, 1982), A. Bachem, M. Grötschel, and B. Korte, eds., Springer, Berlin, 1983, pp. 288–311.

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

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

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

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

[71] P. Spellucci, *Han's method without solving QP*, in Optimization and Optimal Control (Proceedings of the Conf. Math. Res. Inst., Oberwolfach, Oberwolfache-Walke, Germany, 1980), Springer, Berlin, 1981, pp. 123–141.

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

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

[74] 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.

[75] 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.

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

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

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

[79] 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.