ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

Nonlinear Constrained Optimization: Methods and Software

Sven Leyffer and Ashutosh Mahajan

Mathematics and Computer Science Division

Preprint ANL/MCS-P1729-0310

March 17, 2010

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

Contents

Nonlinear Constrained Optimization: Methods and Software*

Sven Leyffer† and Ashutosh Mahajan ‡

March 17, 2010

Abstract

We survey the foundations of nonlinearly constrained optimization methods, emphasizing general methods and highlighting their key components, namely, the local model and global convergence mechanism. We then categorize current software packages for solving constrained nonlinear optimization problems. The packages include interior-point methods, sequential linear/quadratic programming methods, and augmented Lagrangian methods. For every package we highlight the main methodological components and provide a brief summary of interfaces and availability. We also comment on termination conditions of nonlinear solvers and provide a list of online optimization tools.

Keywords: Mathematical programming methods, Newton-type methods, nonlinear programming, interior-point methods, sequential quadratic programming, sequential linear programming

AMS-MSC2000: 49M05, 49M15, 49M37, 65K05, 90C30, 90C51, 90C55

1 Background and Introduction

Nonlinearly constrained optimization problems (NCOs) are an important class of problems with a broad range of engineering, scientific, and operational applications. For ease of presentation, we consider NCOs of the form

$$\underset{x}{\text{minimize}} \;\; f(x) \quad \text{subject to} \;\; c(x) = 0 \;\; \text{and} \;\; x \geq 0, \tag{1.1}$$

where the objective function, $f : \mathbb{R}^n \to \mathbb{R}$, and the constraint functions, $c : \mathbb{R}^n \to \mathbb{R}^m$, are twice continuously differentiable. We denote the multipliers corresponding to the equality constraints, $c(x) = 0$, by y and the multipliers of the inequality constraints, $x \geq 0$, by $z \geq 0$. An NCO may also have unbounded variables, upper bounds, or general range constraints of the form $l_i \leq c_i(x) \leq u_i$, which we omit for the sake of simplicity.

*Preprint ANL/MCS-P1729-0310

†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, leyffer@mcs.anl.gov.

‡Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, mahajan@mcs.anl.gov.

In general, one cannot solve (1.1) directly or explicitly. Instead, an iterative method is used that solves a sequence of simpler, approximate subproblems to generate a sequence of approximate solutions, $\{x_k\}$, starting from an initial guess, x_0. A simplified algorithmic framework for solving (1.1) is as follows.

Given initial estimate $x_0 \in \mathbb{R}^n$, set $k = 0$;
while x_k *is not optimal* **do**
  repeat
    Approximately solve and refine a local model of (1.1) around x_k.
  until *an improved solution estimate* x_{k+1} *is found* ;
  Check whether x_{k+1} is optimal; set $k = k + 1$.
end

Algorithm 1: Framework for Nonlinear Optimization Methods

In this paper, we review the basic components of methods for solving NCOs. In particular, we review the four fundamental components of Algorithm 1: the *convergence test* that checks for optimal solutions or detects failure; the *local model* that computes an improved new iterate; the *globalization strategy* that ensures convergence from remote starting points, by indicating whether a new solution estimate is better than the current estimate; and the *globalization mechanism* that truncates the step computed by the local model to enforce the globalization strategy, effectively refining the local model.

Algorithms for NCOs are categorized by the choice they implement for each of these fundamental components. In the next section, we review the fundamental building blocks of methods for nonlinearly constrained optimization.

Notation: Throughout this paper, we denote iterates by $x_k, k = 1, 2, \ldots$, and we use subscripts to indicate functions evaluated at an iterate, for example, $f_k = f(x_k)$ and $c_k = c(x_k)$. We also denote the gradients by $g_k = \nabla f(x_k)$ and the Jacobian by $A_k = \nabla c(x_k)$. The Hessian of the Lagrangian is denoted by H_k.

2 Convergence Test and Termination Conditions

We start by describing the convergence test, a common component among all NCO algorithms. The convergence test also provides the motivation for many local models that are described next. The convergence analysis of NCO algorithms typically provides convergence only to KKT points. A suitable approximate convergence test is thus given by

$$\|c(x_k)\| \leq \epsilon \;\text{ and }\; \|g_k - A_k y_k - z_k\| \leq \epsilon \;\text{ and }\; \|\min(x_k, z_k)\| \leq \epsilon, \tag{2.1}$$

where $\epsilon > 0$ is the tolerance and the $\min$ in the last expression corresponding to complementary slackness is taken componentwise.

In practice, it may not be possible to ensure convergence to an approximate KKT point, for example, if the constraints fail to satisfy a constraint qualification (Mangasarian, 1969, Ch. 7). In that case, we replace the second condition by

$$\|A_k y_k + z_k\| \leq \epsilon,$$

which corresponds to a Fritz-John point.

Infeasible Stationary Points Unless the NCO is convex or some restrictive assumptions are made, methods cannot guarantee convergence even to a feasible point. Moreover, an NCO may not even have a feasible point, and we are interested in a (local) certificate of infeasibility. In this case, neither the local model nor the convergence test is adequate to achieve and detect convergence. A more appropriate convergence test and local model can be based on the following feasibility problem:

$$\underset{x}{\text{minimize}} \ \|c(x)\| \quad \text{subject to} \ \ x \geq 0, \tag{2.2}$$

which can be formulated as a smooth optimization problem by introducing slack variables. Algorithms for solving (2.2) are analogous to algorithms for NCOs, because the feasibility problem can be reformulated as a smooth NCO by introducing additional variables. In general, we can replace this objective by any weighted norm. A suitable convergence test is then

$$\|A_k y_k - z_k\| \leq \epsilon \ \text{ and } \ \|\min(x_k, z_k)\| \leq \epsilon,$$

where y_k are the multipliers or weights corresponding to the norm used in the objective of (2.2). For example, if we use the ℓ_1 norm, then $y_k \in \{-1, 1\}^m$ depending on which side of the equality constraint is active. The multipliers are readily computed as a by-product of solving the local model.

3 Local Model: Improving a Solution Estimate

One key difference among nonlinear optimization methods is how the local model is constructed. The goal of the local model is to provide a step that improves on the current iterate. We distinguish three broad classes of local models: sequential linear models, sequential quadratic models, and interior-point models. Models that are based on the augmented Lagrangian method are more suitably described in the context of globalization strategies in Section 4.

3.1 Sequential Linear and Quadratic Programming

Sequential linear and quadratic programming methods construct a linear or quadratic approximation of (1.1) and solve a sequence of such approximations, converging to a stationary point.

Sequential Quadratic Programming (SQP) Methods: SQP methods successively minimize a quadratic model, $m_k(d)$, subject to a linearization of the constraints about x_k (Han, 1977; Powell, 1978; Boggs and Tolle, 1995) to obtain a displacement $d := x - x_k$.

$$\underset{d}{\text{minimize}}\ m_k(d) := g_k^T d + \frac{1}{2} d^T H_k d \quad \text{subject to}\ \ c_k + A_k^T d = 0,\ x_k + d \geq 0, \tag{3.1}$$

where $H_k \simeq \nabla^2 \mathcal{L}(x_k, y_k)$ approximates the Hessian of the Lagrangian and y_k is the multiplier estimate at iteration k. The new iterate is $x_{k+1} = x_k + d$, together with the multipliers y_{k+1} of the linearized constraints of (3.1). If H_k is not positive definite on the null-space of the active constraint normals, then the QP is nonconvex, and SQP methods seek a local minimum of (3.1). The solution of the QP subproblem can become computationally expensive for large-scale problems because the null-space method for solving QPs requires the factorization of a dense reduced-Hessian matrix. This bottleneck has led to the development of other methods that use LP solves in the local model, and these approaches are described next.

Sequential Linear Programming (SLP) Methods: SLP methods construct a linear approximation to (1.1). In general, this LP will be unbounded, and SLP methods require the addition of a trust region (discussed in more detail in the next section):

$$\underset{d}{\text{minimize}}\ m_k(d) = g_k^T d \quad \text{subject to}\ c_k + A_k^T d = 0,\ x_k + d \geq 0,\ \text{ and }\ \|d\|_\infty \leq \Delta_k, \tag{3.2}$$

where $\Delta_k > 0$ is the trust-region radius. Griffith and Stewart (1961) used this method without a trust region but with the assumption that the variables are bounded. In general, $\Delta_k \to 0$ must converge to zero to ensure convergence. SLP methods can be viewed as steepest descent methods and typically converge only linearly. If, however there are exactly n active and linearly independent constraint normals at the solution, then SLP reduces to Newton's method for solving a square system of nonlinear equations and converges superlinearly.

Sequential Linear/Quadratic Programming (SLQP) Methods: SLQP methods combine the advantages of the SLP method (fast solution of the LP) and SQP methods (fast local convergence) by adding an equality-constrained QP to the SLP method (Fletcher and de la Maza, 1989; Chin and Fletcher, 2003; Byrd et al., 2004). SLQP methods thus solve two subproblems: first, an LP is solved to obtain a step for the next iteration and also an estimate of the active set $\mathcal{A}_k := \left\{ i : [x_k]_i + \hat{d}_i = 0 \right\}$ from a solution $\hat{d}$ of (3.2). This estimate of the active set is then used to construct an equality-constrained QP (EQP), on the active constraints,

$$\underset{d}{\text{minimize}}\, q_k(d) = g_k^T d + \frac{1}{2} d^T H_k d \quad \text{subject to}\ \ c_k + A_k^T d = 0,\ [x_k]_i + d_i = 0,\ \forall i \in \mathcal{A}_k. \tag{3.3}$$

If H_k is second-order sufficient (i.e., positive-definite on the null-space of the constraints), then the solution of (3.3) is equivalent to the following linear system obtained by applying the KKT

conditions to the EQP:

$$\begin{bmatrix} H_k & -A_k & -I_k \\ A_k^T & & \\ I_k^T & & \end{bmatrix} \begin{pmatrix} x \\ y_{\mathcal{A}} \end{pmatrix} = \begin{pmatrix} -g_k + H_k x_k \\ -c_k \\ 0 \end{pmatrix},$$

where $I_k = [e_i]_{i \in \mathcal{A}_k}$ are the normals of the active inequality constraints. By taking a suitable basis from the LP simplex solve, SLQP methods can ensure that $[A_k : I_k]$ has full rank. Linear solvers such as MA57 can also detect the inertia; and if H_k is not second-order sufficient, a multiple of the identity can be added to H_k to ensure descent of the EQP step.

Sequential Quadratic/Quadratic Programming (SQQP) Methods: SQQP methods have recently been proposed as SQP types of methods (Gould and Robinson, 2010, 2008). First, a convex QP model constructed by using a positive-definite Hessian approximation is solved. The solution of this convex QP is followed by a reduced inequality constrained model or an EQP with the exact second derivative of the Lagrangian.

Theory of Sequential Linear/Quadratic Programming Methods. If H_k is the exact Hessian of the Lagrangian and if the Jacobian of the active constraints has full rank, then SQP methods converge quadratically near a minimizer that satisfies a constraint qualification and a second-order sufficient condition (Boggs and Tolle, 1995). It can also be shown that, under the additional assumption of strict complementarity, all four methods identify the optimal active set in a finite number of iterations.

The methods described in this section are also often referred to as *active-set methods*, because the solution of each LP or QP provides not only a suitable new iterate but also an estimate of the active set at the solution.

3.2 Interior-Point Methods

Interior-point methods (IPMs) are an alternative approach to active-set methods. Interior-point methods are a class of perturbed Newton methods that postpone the decision of which constraints are active until the end of the iterative process. The most successful IPMs are primal-dual IPMs, which can be viewed as Newton's method applied to the perturbed first-order conditions of (1.1):

$$0 = F_\mu(x, y, z) = \begin{pmatrix} \nabla f(x) - \nabla c(x)^T y - z \\ c(x) \\ Xz - \mu e \end{pmatrix}, \tag{3.4}$$

where $\mu > 0$ is the barrier parameter, $X = \text{diag}(x)$ is a diagonal matrix with x along its diagonal, and $e = (1, \ldots, 1)$ is the vector of all ones. Note that, for $\mu = 0$, these conditions are equivalent to the first-order conditions except for the absence of the nonnegativity constraints $x, z \geq 0$.

Interior-point methods start at an "interior" iterate $x_0, z_0 > 0$ and generate a sequence of interior iterates $x_k, z_k > 0$ by approximately solving the first-order conditions (3.4) for a decreasing

sequence of barrier parameters. Interior-point methods can be shown to be polynomial-time algorithms for convex NLPs; see, for example, (Nesterov and Nemirovskii, 1994).

Newton's method applied to the primal-dual system (3.4) around x_k gives rise to the local model,

$$\begin{bmatrix} H_k & -A_k & -I \\ A_k^T & 0 & 0 \\ Z_k & 0 & X_k \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = -F_\mu(x_k, y_k, z_k), \tag{3.5}$$

where H_k approximates the Hessian of the Lagrangian, $\nabla^2 \mathcal{L}_k$, and the step $(x_{k+1}, y_{k+1}, z_{k+1}) = (x_k, y_k, z_k) + (\alpha_x \Delta_x, \alpha_y \Delta_y, \alpha_z \Delta_z)$ is safeguarded to ensure that $x_{k+1}, z_{k+1} > 0$ remain strictly positive.

Relationship to Barrier Methods: Primal-dual interior-point methods are related to earlier barrier methods (Fiacco and McCormick, 1990). These methods were given much attention in the 1960s but soon lost favor because of the ill-conditioning of the Hessian. They regained attention in the 1980s after it was shown that these methods can provide polynomial-time algorithms for linear programming problems. See the surveys (Wright, 1992; Forsgren et al., 2002) for further material. Barrier methods approximately solve a sequence of barrier problems,

$$\underset{x}{\text{minimize}} \;\; f(x) - \mu \sum_{i=1}^{n} \log(x_i) \quad \text{subject to} \;\; c(x) = 0, \tag{3.6}$$

for a decreasing sequence of barrier parameters $\mu > 0$. The first-order conditions of (3.6) are given by

$$\nabla f(x) - \mu X^{-1} e - A(x) y = 0 \;\; \text{and} \;\; c(x) = 0. \tag{3.7}$$

Applying Newton's method to this system of equations results in the following linear system:

$$\begin{bmatrix} H_k + \mu X_k^{-2} & -A_k \\ A_k^T & 0 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = - \begin{pmatrix} g_k - \mu X_k^{-1} e - A_k y_k \\ c_k \end{pmatrix}.$$

Introducing first-order multiplier estimates $Z(x_k) := \mu X_k^{-1}$, which can be written as $Z(x_k) X_k = \mu e$, we obtain the system

$$\begin{bmatrix} H_k + Z(x_k) X_k^{-1} & -A_k \\ A_k & 0 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = - \begin{pmatrix} g_k - \mu X_k^{-1} e - A_k y_k \\ c_k \end{pmatrix},$$

which is equivalent to the primal-dual Newton system (3.5), where we have eliminated

$$\Delta z = -X^{-1} Z \Delta x - Ze - \mu X^{-1} e.$$

Thus, the main difference between classical barrier methods and the primal-dual IPMs is that Z_k is not free for barrier methods but is chosen as the primal multiplier $Z(x_k) = \mu X_k^{-1}$. This freedom in the primal-dual method avoids some difficulties with ill-conditioning of the barrier Hessian.

Convergence of Barrier Methods: If there exists a compact set of isolated local minimizers of (1.1) with at least one point in the closure of the strictly feasible set, then it follows that barrier methods converge to a local minimum (Wright, 1992).

4 Globalization Strategy: Convergence from Remote Starting Points

The local improvement models of the preceding section guarantee convergence only in a small neighborhood of a regular solution. Globalization strategies are concerned with ensuring convergence from remote starting points to stationary points (and should not be confused with global optimization). To ensure convergence from remote starting points, we must monitor the progress of the local method. Monitoring is easily done in unconstrained optimization, where we can measure progress by comparing objective values. In constrained optimization, however, we must take the constraint violation into account. Three broad classes of strategies exist: augmented Lagrangian methods, penalty and merit-function methods, and filter and funnel methods.

4.1 Augmented Lagrangian Methods

The augmented Lagrangian of (1.1) is given by

$$\mathcal{L}(x, y, \rho) = f(x) - y^T c(x) + \frac{\rho}{2}\|c(x)\|_2^2, \tag{4.1}$$

where $\rho > 0$ is the penalty parameter. The augmented Lagrangian is used in two modes to develop algorithms for solving (1.1): by defining a linearly constrained problem or by defining a bound constrained problem.

Linearly constrained Lagrangian methods: These methods successively minimize a shifted augmented Lagrangian subject to a linearization of the constraints. The shifted augmented Lagrangian is defined as

$$\overline{\mathcal{L}}(x, y, \rho) = f(x) - y^T p_k(x) + \frac{\rho}{2}\|p_k(x)\|_2^2, \tag{4.2}$$

where $p_k(x)$ are the higher-order nonlinear terms at the current iterate x_k, that is,

$$p_k(x) = c(x) - c_k - A_k^T(x - x_k). \tag{4.3}$$

This approach results in the following local model:

$$\underset{x}{\text{minimize}}\ \overline{\mathcal{L}}(x, y_k, \rho_k) \quad \text{subject to } c_k + A_k^T(x - x_k) = 0,\ x \geq 0. \tag{4.4}$$

We note that if $c_k + A_k^T(x - x_k) = 0$, then minimizing the shifted augmented Lagrangian is equivalent to minimizing the Lagrangian over these constraints. Linearly constrained, augmented Lagrangian methods solve a sequence of problems (4.4) for a fixed penalty parameter. Multipliers are updated by using a first-order multiplier update rule,

$$y_{k+1} = y_k - \rho_k c(x_{k+1}), \tag{4.5}$$

where x_{k+1} solves (4.4).

Bound-constrained Lagrangian methods: These methods approximately minimize the augmented Lagrangian,

$$\underset{x}{\text{minimize}} \ \mathcal{L}(x, y_k, \rho_k) \quad \text{subject to } x \geq 0. \tag{4.6}$$

The advantage of this approach is that efficient methods for bound-constrained optimization can readily be applied, such as the gradient-projection conjugate-gradient approach (Moré and Toraldo, 1991), which can be interpreted as an approximate Newton method on the active inequality constraints.

Global convergence is promoted by defining two forcing sequences, $\omega_k \searrow 0$, controlling the accuracy with which every bound-constrained problems is solved, and $\eta_k \searrow 0$, controlling progress toward feasibility of the nonlinear constraints. A typical bound-constrained Lagrangian method can then be stated as follows:

Given an initial solution estimate (x_0, y_0), and an initial penalty parameter ρ_0.
while x_k *is not optimal* **do**
  Find an ω_k-optimal solution, x_k^c of (4.6).
  if $\|c(x_k^c)\| \leq \eta_k$ **then**
    Perform a first-order multiplier update: $y_{k+1} = y_k - \rho_k c(x_k^c)$
  else
    Increase penalty: $\rho_{k+1} = 10\rho_k$
  end
  Set $k = k + 1$
end

Algorithm 2: Bound-Constrained Augmented Lagrangian Method.

Theory of Augmented Lagrangian Methods. Conn et al. (1991) show that a bound-constrained Lagrangian method can globally converge if the sequence $\{x_k\}$ of iterates is bounded and if the Jacobian of the constraints at all limit points of $\{x_k\}$ has column rank no smaller than m. Conn et al. (1991) show that if some additional conditions are met, then their algorithm is R-linearly convergent. Bertsekas (1996) shows that the method converges Q-linearly if $\{\rho_k\}$ is bounded, and superlinearly otherwise. Linearly constrained augmented Lagrangian methods can be made globally convergent by adding slack variables to deal with infeasible subproblems (Friedlander and Saunders, 2005).

4.2 Penalty and Merit Function Methods

Penalty and merit functions combine the objective function and a measure of the constraint violation into a single function whose local minimizers correspond to local minimizers of the original problem (1.1). Convergence from remote starting points can then be ensured by forcing descent of the penalty or merit function, using one of the mechanisms of the next section.

Exact penalty functions are an attractive alternative to augmented Lagrangians and are defined as

$$p_\rho(x) = f(x) + \rho\|c(x)\|,$$

where $\rho > 0$ is the penalty parameter. Most approaches use the ℓ_1 norm to define the penalty function. It can be shown that a local minimizer, x^*, of $p_\rho(x)$ is a local minimizer of problem (1.1) if $\rho > \|y^*\|_D$, where y^* are the corresponding Lagrange multipliers and $\|\cdot\|_D$ is the dual norm of $\|\cdot\|$ (i.e., the ℓ_∞-norm in the case of the ℓ_1 exact-penalty function); see, for example, (Fletcher, 1987, Chapter 12.3). Classical approaches using $p_\rho(x)$ have solved a sequence of penalty problems for an increasing sequence of penalty parameters. Modern approaches attempt to steer the penalty parameter by comparing the predicted decrease in the constraint violation to the actual decrease over a step.

A number of other merit functions also exist. The oldest, the quadratic penalty function, $f(x) + \rho\|c(x)\|_2^2$, converges only if the penalty parameter diverges to infinity. Augmented Lagrangian functions and Lagrangian penalty functions such as $f(x) + y^T c(x) + \rho\|c(x)\|$ have also been used to promote global convergence. A key ingredient in any convergence analysis is to connect the local model to the merit function that is being used in a way that ensures a descent property of the merit function; see Section 5.1.

4.3 Filter and Funnel Methods

Filter and funnel methods provide an alternative to penalty methods that does not rely on the use of a penalty parameter. Both methods use step acceptance strategies that are closer to the original problem, by separating the constraints and the objective function.

Filter methods: Filter methods keep a record of the constraint violation, $h_l := \|c(x_l)\|$, and objective function value, $f_l := f(x_l)$, for some previous iterates, $x_l, l \in \mathcal{F}_k$ (Fletcher and Leyffer, 2002). A new point is acceptable if it improves either the objective function or the constraint violation compared to all previous iterates. That is, $\hat{x}$ is acceptable if

$$f(\hat{x}) \leq f_l - \gamma h_l \quad \text{or} \quad h(\hat{x}) \leq \beta h_l, \ \forall l \in \mathcal{F}_k,$$

where $\gamma > 0, 0 < \beta < 1$, are constants that ensure that iterates cannot accumulate at infeasible limit points. A typical filter is shown in Figure 1 (left), where the straight lines correspond to the region in the (h, f)-plane that is dominated by previous iterations and the dashed lines correspond to the envelope defined by γ, β.

The filter provides convergence only to a feasible limit because any infinite sequence of iterates must converge to a point, where $h(x) = 0$, provided that $f(x)$ is bounded below. To ensure convergence to a local minimum, filter methods use a standard sufficient reduction condition from unconstrained optimization,

$$f(x_k) - f(x_k + d) \geq -\sigma m_k(d), \tag{4.7}$$

where $\sigma > 0$ is the fraction of predicted decrease and $m_k(d)$ is the model reduction from the local model. It makes sense to enforce this condition only if the model predicts a decrease in the objective function. Thus, filter methods use the switching condition $m_k(d) \geq \gamma h_k^2$ to decide when (4.7) should be enforced. A new iterate that satisfies both conditions is called an f-type iterate, and an iterate for which the switching condition fails is called an h-type iterate to indicate that it mostly reduces the constraint violation. If a new point is accepted, then it is added to the current iterate to the filter, $\mathcal{F}_k$, if $h_k > 0$ or if it corresponds to an h-type iterations (which automatically satisfy $h_k > 0$).

Funnel methods: The method of Gould and Toint (2010) can be viewed as a filter method with just a single filter entry, corresponding to an upper bound on the constraint violation. Thus, the filter contains only a single entry, $(U_k, -\infty)$. The upper bound is reduced during h-type iterations, to force the iterates toward feasibility; it is left unchanged during f-type iterations. Thus, it is possible to converge without reducing U_k to zero (consistent with the observation that SQP methods converge locally). A schematic interpretation of the funnel is given in Figure 1 (left).

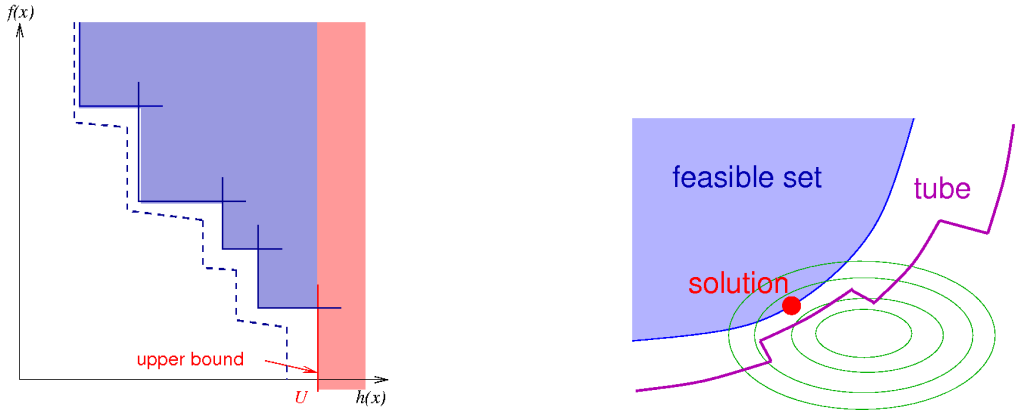

Figure 1: The left figure shows a filter where the blue/red area corresponds to the points that are rejected by the filter. The right figure shows a funnel around the feasible set.

4.4 Maratos Effect and Loss of Fast Convergence

One can construct simple examples showing that arbitrarily close to an isolated strict local minimizer, the Newton step will be rejected by the exact penalty function (Maratos, 1978), resulting in slow convergence. This phenomenon is known as the Maratos effect. It can be mitigated by computing a second-order correction step, which is a Newton step that uses the same linear system with an updated right-hand side (Fletcher, 1987; Nocedal and Wright, 1999). An alternative method to avoid the Maratos effect is the use of nonmonotone techniques that require descent over only the last M iterates, where $M > 1$ is a constant.

5 Globalization Mechanisms

In this section, we review two mechanisms to reduce the step that is computed by the local model: line-search methods and trust-region methods. Both mechanisms can be used in conjunction with any of the local models and any of the global convergence strategies, giving rise to a broad family of algorithms. In Sections 7–9, we describe how these components are used in software for NCOs.

5.1 Line-Search Methods

Line-search methods enforce convergence with a backtracking line search along the direction s. For interior-point methods, the search direction, $s = (\Delta_x, \Delta_y, \Delta_z)$, is obtained by solving the primal-dual system (3.5). For SQP methods, the search direction is the solution of the QP (3.1), $s = d$. It is important to ensure that the model produces a descent direction, e.g., $\nabla\Phi(x_k)^T s < 0$ for a merit or penalty function $\Phi(x)$; otherwise, the line search may not terminate. A popular line search is the Armijo search (Nocedal and Wright, 1999), described in Algorithm 3 for a merit function $\Phi(x)$. The algorithm can be shown to converge to a stationary point, detect unboundedness, or converge to a point where there are no directions of descent.

Given initial estimate $x_0 \in \mathbb{R}^n$, let $0 < \sigma < 1$, and set $k = 0$;
while x_k *is not optimal* **do**
  Approximately solve a local model of (1.1) around x_k to find a search direction s.
  Make sure that s is a descent direction, e.g. $\nabla\Phi(x_k)^T s < 0$.
  Set $\alpha^0 = 1$ and $l = 0$.
  repeat
    Set $\alpha^{l+1} = \alpha^l/2$ and evaluate $\Phi(x_k + \alpha^{l+1}s)$. Set $l = l + 1$.
  until $\Phi(x_k + \alpha^l s) \leq f_k + \alpha^l \sigma s^T \nabla\Phi_k$;
  set $k = k + 1$.
end

Algorithm 3: (Armijo) Line-Search Method for Nonlinear Optimization

Line-search methods for filters can be defined in a similar way. Instead of checking descent in the merit function, a filter method is used to check acceptance to a filter. Unlike merit functions, filter methods do not have a simple definition of descent; hence, the line search is terminated unsuccessfully once the step size α^l becomes smaller than a constant. In this case, filter methods switch to a restoration step, obtained by solving a local approximation of (2.2).

5.2 Trust-Region Methods

Trust-region methods explicitly restrict the step that is computed by the local model, by adding a trust-region constraint of the form $\|d\| \leq \Delta_k$ to the local model. Most methods use an ℓ_∞-norm trust region, which can be represented by bounds on the variables. The trust-region radius, $\Delta_k > 0$, is adjusted at every iteration depending on how well the local model agrees with the

NCO, (1.1).

Given initial estimate $x_0 \in \mathbb{R}^n$, choose $\Delta_0 > 0$, and set $k = 0$;
while x_k *is not optimal* **do**
  Reset Δ_k;
  repeat
    Approximately solve a local *trust-region* model with $\|d\| \leq \Delta_k$.
    if $x_k + d$ *is sufficiently better than* x_k **then**
      Accept the step: $x_{k+1} = x_k + d$; possibly increase Δ_k.
    else
      Reject the step and decrease the trust-region radius, e.g. $\Delta_k = \Delta_k/2$.
    end
  until *an improved solution estimate* x_{k+1} *is found* ;
  Check whether x_{k+1} is optimal; set $k = k + 1$.
end

Algorithm 4: Trust-Region Methods for Nonlinear Optimization

Trust-region methods are related to regularization techniques, which add a multiple of the identity matrix, $\sigma_k I$, to the Hessian, H_k. Locally, the solution of the regularized problem is equivalent to the solution of a trust-region problem with an ℓ_2 trust-region.

6 Nonlinear Optimization Software: Summary

Software for nonlinearly constrained optimization can be applied to problems that are more general than (1.1). In particular, solvers take advantage of linear constraints or simple bounds. Thus, a more appropriate model problem is of the form

$$\begin{cases} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & l_c \leq c(x) \leq u_c \\ & l_A \leq A^T x \leq u_A \\ & l_x \leq x \leq u_x, \end{cases} \tag{6.1}$$

where the objective function, $f : \mathbb{R}^n \to \mathbb{R}$, and the constraint functions, $c_i : \mathbb{R}^n \to \mathbb{R}$, for $i = 1, \ldots, m$, are twice continuously differentiable. The bounds, $l_c, l_A, l_x, u_c, u_A, u_x$, can be either finite or infinite Equality constraints are modeled by setting $l_j = u_j$ for some index j. Maximization problems can be solved by multiplying the objective by -1 (most solvers handle this transformation internally).

NCO solvers are typically designed to work well for a range of other optimization problems such as solving a system of nonlinear equations (most methods reduce to Newton's method in this case), bound-constrained problems, and LP or QP problems. In this survey, we concentrate on solvers that can handle general NCO problems possibly involving nonconvex functions.

Methods for solving (6.1) are iterative and contain the following four basic components: a *local model* that approximates (6.1), a *global convergence strategy* to promote convergence from remote

starting points, a *global convergence mechanism* to force improvement in the global convergence strategy, and a *convergence test* to detect the type of limit point that is reached (see Section 2). Solvers for NCO are differentiated by how each of these key ingredients is implemented. In addition there are a number of secondary distinguishing factors such as licensing (open-source versus commercial or academic), API and interfaces to modeling languages, sparse or dense linear algebra, programming language (Fortran/C/MATLAB), and compute platforms on which a solver can run.

The next sections list some solvers for NCOs, summarizing their main characteristics. A short overview can be found in Table 1. We distinguish solvers mainly by the definition of their local model.

7 Interior-Point Solvers

Interior-point methods approximately solve a sequence of perturbed KKT systems, driving a barrier parameter to zero. Interior-point methods can be regarded as perturbed Newton methods applied to the KKT system in which the primal/dual variables are kept positive.

7.1 CVXOPT

Algorithmic Methodology: CVXOPT (Dahl and Vandenberghe, 2010a) is a software package for *convex optimization*, with interfaces for linear algebra routines (BLAS and LAPACK), Fourier transforms (FFTW), system of equalities (CHOLMOD, UMFPACK), and other solvers (GLPK, MOSEK and DSDP5). It uses an interior-point barrier method that approximately solves a sequence of perturbed KKT systems in each step. Global convergence relies on the convergence of the barrier method and *does not generalize to nonconvex problems*.

Software and Technical Details: CVXOPT is available under GPL license from the CVXOPT webpage (Dahl and Vandenberghe, 2010b). It is implemented in Python. It can be used with the interactive Python interpreter, on the command line by executing Python scripts, or integrated in other software via Python extension modules. Existing Python classes can be used for matrices and arithmetic to input the problem.

7.2 IPOPT

Algorithmic Methodology: IPOPT is a line-search filter interior-point method (Wächter and Biegler, 2005b,a; Wächter and Biegler, 2006). The outer loop approximately minimizes a sequence of nonlinearly (equality) constrained barrier problems for a decreasing sequence of barrier parameters. The inner loop uses a line-search filter SQP method to approximately solve each barrier problem. Global convergence of each barrier problem is enforced through a line-search filter method, and the filter is reset after each barrier parameter update. Steps are computed by solving

Table 1: NCO Software Overview.

Name	Model	Global Method	Interfaces	Language
ALGENCAN	Aug. Lag.	augmented Lagrangian	AMPL, C/C++, CUTEr, Java, MATLAB, Octave, Python, R	f77
CONOPT	GRG/SLQP	line-search	AIMMS, GAMS	Fortran
CVXOPT	IPM	*only convex*	Python	Python
FilterSQP	SQP	filter/trust region	AMPL, CUTEr, f77	Fortran77
GALAHAD	Aug. Lag.	nonmonotone/ augmented Lagrangian	CUTEr, Fortran	Fortran95
IPOPT	IPM	filter/line search	AMPL, CUTEr, C, C++, f77	C++
KNITRO	IPM	penalty-barrier/ trust region	AIMMS, AMPL, GAMS, Mathematica, MATLAB, MPL, C, C++, f77, Java, Excel	C++
KNITRO	SLQP	penalty/trust region	s.a.	C++
LANCELOT	Aug. Lag.	augmented Lagrangian/ trust region	SIF, AMPL, f77	Fortran77
LINDO	GRG/SLP	*only convex*	C, MATLAB, LINGO	
LOQO	IPM	line search	AMPL, C, MATLAB	C
LRAMBO	SQP	ℓ_1 exact penalty/ line search	C	C/C++
MINOS	Aug. Lag.	augmented Lagrangian	AIMMS, AMPL, GAMS, MATLAB, C, C++, f77	Fortran77
NLPQLP	SQP	augmented Lagrangian/ line-search	C, f77, MATLAB	Fortran77
NPSOL	SQP	penalty Lagrangian/ line search	AIMMS, AMPL, GAMS, MATLAB, C, C++, f77	Fortran77
PATH	LCP	line search	AMPL	C
PENNON	Aug. Lag.	line search	AMPL, MATLAB	C
SNOPT	SQP	penalty Lagrangian/ line search	AIMMS, AMPL, GAMS, MATLAB, C, C++, f77	Fortran77
SQPlab	SQP	penalty Lagrangian/ line search	MATLAB	MATLAB

a primal-dual system, (3.5), corresponding to the KKT conditions of the barrier problem. The algorithm controls the inertia of this system by adding δI ($\delta > 0$) to the Hessian of the Lagrangian, ensuring descent properties. The inner iteration includes second-order correction steps and mech-

anisms for switching to a feasibility restoration if the step size becomes too small. The solver has an option for using limited-memory BFGS updates to approximate the Hessian of the Lagrangian.

Software and Technical Details: The solver is written in C++ and has interfaces to C, C++, Fortran, AMPL, CUTEr, and COIN-OR's NLPAPI. It requires BLAS, LAPACK, and a sparse indefinite solver (MA27, MA57, PARDISO, or WSMP). The user must provide function and gradient information, and possibly the Hessian of the Lagrangian (in sparse format). By using the PARDISO (Schenk et al., 2007) parallel linear solver, IPOPT can solve large problems on shared-memory multiprocessors. IPOPT is available on COIN-OR under Common Public License at its website (Wächter and Biegler, 2010).

7.3 KNITRO

KNITRO includes both IPM and SLQP methods. The SLQP version is described in the next section.

Algorithmic Methodology: KNITRO implements a trust-region interior-point penalty-barrier method (Byrd et al., 1999, 2000, 2006). It approximately solves a sequence of barrier subproblems for a decreasing sequence of barrier parameters, using a trust region and penalty-barrier function to promote global convergence. The barrier subproblems are solved by a sequence of linearized primal-dual equations, (3.5). KNITRO has two options for solving the primal-dual system: direct factorization of the system of equations and preconditioned conjugate gradient (PCG) method. The PCG method solves the indefinite primal-dual system by projecting onto the null space of the equality constraints. KNITRO also includes modules for solving mixed-integer nonlinear optimization problems and optimization problems with simple complementarity constraints. In addition, it contains crossover techniques to obtain an active set from the solution of the IPM solve and a multistart heuristic for nonconvex NCOs.

Software and Technical Details: KNITRO is written in C++. It has interfaces to a range of modeling languages, including AIMMS, AMPL, GAMS, Mathematica, MATLAB, and MPL. In addition, KNITRO has interfaces to C, C++, Fortran, Java, and Excel. It offers callback and reverse communication interfaces. KNITRO requires MA57 to solve the indefinite linear systems of equations. KNITRO is available from Zienna Inc., and the user's manual is available online (Waltz and Plantenga, 2009).

7.4 LOQO

Algorithmic Methodology: LOQO (Vanderbei and Shanno, 1999) uses an infeasible primal-dual interior-point method for solving general nonlinear problems. The inequality constraints are added to the objective by using a log-barrier function. Newton's method is used to obtain a solution to the system of nonlinear equations that is obtained by applying first-order necessary conditions to this barrier function. The solution of this system provides a search direction, and

a merit function is used to find the next iterate in this direction. The merit function is the barrier function and an additional ℓ_2 norm of the violation of constraints. The exact Hessian of the Lagrangian is used in the Newton's method. When the problem is nonconvex, this Hessian is perturbed by adding to it the matrix δI, where I is an identity matrix and $\delta > 0$ is chosen so that the perturbed Hessian is positive definite. This perturbation ensures that if the iterates converge, they converge to a local minimum.

Software and Technical Details: LOQO can be used to solve problems written in AMPL. The user can also implement functions in C and link to the LOQO library. LOQO can also read files in MPS format for solving LPs. A MATLAB interface for LOQO is available for LPs, QPs, and second-order cone programs. The library and binary files are available for a license fee at its homepage (Vanderbei, 2010a), while a limited student version is available freely. A user manual provides instructions on installing and using LOQO.

8 Sequential Linear/Quadratic Solvers

Sequential linear/quadratic solvers solve a sequence of LP and/or QP subproblems. This class of solvers is also referred to as active-set methods, because they provide an estimate of the active set at every iteration. Once the active set settles down, these methods become Newton methods on the active constraints (except SLP).

8.1 CONOPT

Algorithmic Methodology: CONOPT (Drud, 1985, 2007) implements three active-set methods. The first is a gradient projection method that projects the gradient of the objective onto a linearization of the constraints and makes progress toward the solution by reducing the objective. The second variant is an SLP method, and the third is an SQP method. CONOPT includes algorithmic switches that automatically detect which method is preferable. It also exploits triangular parts of the Jacobian matrix and linear components of the Jacobian to reduce the sparse factorization overhead. CONOPT is a line-search method.

Software and Technical Details: CONOPT has interfaces to GAMS and AIMMS and is available from either GAMS or AIMMS.

8.2 FilterSQP

Algorithmic Methodology: FilterSQP (Fletcher and Leyffer, 1998) implements a trust-region SQP method. Convergence is enforced with a filter, whose components are the ℓ_1-norm of the constraint violation, and the objective function. The solver starts by projecting the user-supplied initial guess onto the linear part of the feasible set and remains feasible with respect to the linear constraints. It uses an indefinite QP solver that finds local solutions to problems with negative

curvature. The solver switches to a feasibility restoration phase if the local QP model becomes inconsistent. During the restoration phase, a weighted sum of the constraint violations is minimized by using a filter SQP trust-region approach. The restoration phase either terminates at a local minimum of the constraint violation or returns to the main algorithm when a feasible local QP model is found. FilterSQP computes second-order correction steps if the QP step is rejected by the filter.

Software and Technical Details: The solver is implemented in Fortran77 and has interfaces to AMPL, CUTEr, and Fortran77. The code requires subroutines to evaluate the problem functions, their gradients, and the Hessian of the Lagrangian (provided by AMPL and CUTEr). It uses BQPD (Fletcher, 1999) to solve the possibly indefinite QP subproblems. BQPD is a null-space active-set method and has modules for sparse and dense linear algebra with efficient and stable factorization updates. The code is licensed by the University of Dundee.

8.3 KNITRO

Algorithmic Methodology: In addition to the interior point-methods (see Section 7.3), KNITRO implements an SLQP algorithm (Fletcher and de la Maza, 1989; Byrd et al., 2004). In each iteration, an LP that approximates the ℓ_1 exact-penalty problem is solved to determine an estimate of the active set. Those constraints in the LP that are satisfied as equalities are marked as active and are used to set up an equality-constrained QP (EQP), (3.3), whose objective is a quadratic approximation of the Lagrangian of (6.1) at the current iterate. An ℓ_2-norm trust-region constraint is also added to this QP. The solution of the LP and the EQP are then used to find a search direction (Byrd et al., 2006, 2004). A projected conjugate-gradient method is used to solve the EQP. The penalty parameter ρ_k is updated to ensure sufficient decrease toward feasibility.

Software and Technical Details: The SLQP method of KNITRO requires an LP solver and can be linked to CLP. For other details, see Section 7.3. This active-set algorithm is usually preferable for rapidly detecting infeasible problems and for solving a sequence of closely related problems.

8.4 LINDO

Algorithmic Methodology: LINDO provides solvers for a variety of problems including NCOs. Its nonlinear solver implements an SLP algorithm and a generalized gradient method. It provides options to randomly select the starting point and can estimate derivatives by using finite differences.

Software and Technical Details: LINDO (LINDO Systems Inc., 2010a) provides interfaces for programs written in languages such as C and MATLAB. It can also read models written in LINGO. The full version of the software can be bought online, and a limited trial version is available for free (LINDO Systems Inc., 2010b).

8.5 LRAMBO

Algorithmic Methodology: LRAMBO is a *total quasi-Newton* SQP method. It approximates both the Jacobian and the Hessian by using rank-1 quasi-Newton updates. It enforces global convergence through a line search on an ℓ_1-exact penalty function. LRAMBO requires as input only an evaluation program for the objective and the constraints. It combines automatic differentiation and quasi-Newton updates to update factors of the Jacobian, and it computes updates of the Hessian. Details can be found in (Griewank et al., 2007).

Software and Technical Details: LRAMBO uses the NAG subroutine E04NAF to solve the QP subproblems. It also requires ADOL-C (Griewank et al., 1996; Griewank and Walther, 2004) for the automatic differentiation of the problem functions. LRAMBO is written in C/C++.

8.6 NLPQLP

Algorithmic Methodology: NLPQLP is an extension of the SQP solver NLPQL (Schittkowski, 1985) that implements a nonmonotone line search to ensure global convergence. It uses a quasi-Newton approximation of the Hessian of the Lagrangian, which is updated with the BFGS formula. A nonmonotone line search is used to calculate the step length that minimizes an augmented Lagrangian merit function.

Software and Technical Details: NLPQLP is implemented in Fortran. The user or the interface is required to evaluate the function values of the objective and constraints. Derivatives, if unavailable, can be estimated by using finite differences. NLPQLP can evaluate these functions and also the merit function on a distributed memory system. A user guide with documentation, algorithm details, and examples is available (Schittkowski, 2009). NLPQLP can be obtained under academic and commercial license from its website.

8.7 NPSOL

Algorithmic Methodology: NPSOL (Gill et al., 1998) solves general nonlinear problems by using an SQP algorithm with a line search on the augmented Lagrangian. In each major iteration, a QP subproblem is solved whose Hessian is a quasi-Newton approximation of the Hessian of the Lagrangian using a *dense* BFGS update.

Software and Technical Details: NPSOL is implemented in Fortran, and the library can be called from Fortran and C programs and from modeling languages such as AMPL, GAMS, AIMMS, and MATLAB. The user or the interface must provide routines for evaluating functions and their gradients. If a gradient is not available, NPSOL can estimate it by using finite differences. It treats all matrices as dense and hence may not be efficient for large-sparse problems. NPSOL can be

warm started by specifying the active constraints and multiplier estimates for the QP. NPSOL is available under commercial and academic licenses from Stanford Business Software Inc.

8.8 PATHNLP

Algorithmic Methodology: PATH (Dirkse and Ferris, 1995; Ferris and Munson, 1999) solves mixed complementarity problems (MCPs). PATHNLP automatically formulates the KKT conditions of an NLP, (6.1), specified in GAMS as an MCP and then solves this MCP using PATH. The authors note that this approach is guaranteed only to find stationary points and does not distinguish between local minimizers and maximizers for nonconvex problems. Thus, it works well for convex problems. At each iteration, PATH solves a linearization of the MCP problem to obtain a Newton point. It then performs a search in the direction of this point to find a minimizer of a merit function. If this direction is not a descent direction, it performs a steepest descent step in the merit function to find a new point.

Software and Technical Details: The PATHNLP is available only through GAMS (Dirkse and Ferris, 2010b). The PATH solver is implemented in C and C++ (Dirkse and Ferris, 2010a).

8.9 SNOPT

Algorithmic Methodology: SNOPT (Gill et al., 2006a) implements an SQP algorithm much like NPSOL, but it is suitable for large, sparse problems as well. The Hessian of the Lagrangian is updated by using limited-memory quasi-Newton updates. SNOPT solves each QP using SQOPT (Gill et al., 2006b), which is a reduced-Hessian active-set method. It includes an option for using a projected conjugate gradient method rather than factoring the reduced Hessian. SNOPT starts by solving an "elastic program" that minimizes the constraint violation of the linear constraints of (6.1). The solution to this program is used as a starting point for the major iterations. If a QP subproblem is found to be infeasible or unbounded, then SNOPT tries to solve an elastic problem that corresponds to a smooth reformulation of the ℓ_1-exact penalty function The solution from a major iteration is used to obtain a search direction along which an augmented Lagrangian merit function is minimized.

Software and Technical Details: SNOPT is implemented in Fortran77 and is compatible with newer Fortran compilers. All functions in the SNOPT library can be used in parallel or by multiple threads. It can be called from other programs written in C and Fortran and by packages such as AMPL, GAMS, AIMMS, and MATLAB. The user or the interface has to provide routines that evaluate function values and gradients. When gradients are not available, SNOPT uses finite differences to estimate them. SNOPT can also save basis files that can be used to save the basis information to warm start subsequent QPs. SNOPT is available under commercial and academic licenses from Stanford Business Software Inc.

8.10 SQPlab

Algorithmic Methodology: SQPlab (Gilbert, 2009) was developed as a laboratory for testing algorithmic options of SQP methods (Bonnans et al., 2006). SQPlab implements a line-search SQP method that can use either exact Hessians or a BFGS approximation of the Hessian of the Lagrangian. It maintains positive definiteness of the Hessian approximation using the Wolfe or Powell condition. SQPlab uses a Lagrangian penalty function, $p_\sigma(x, y) = f(x) + y_c^T c(x) + \sigma \| \max\{0, c(x) - u_c, l_c - c(x)\} \|_1$, to promote global convergence. SQPlab has a feature that allows it to treat discretized optimal control constraints specially. The user can specify a set of equality constraints whose Jacobian has uniformly full rank (such as certain discretized optimal-control constraints). SQPlab then uses these constraints to eliminate the state variables. The user needs to provide only routines that multiply a vector by the inverse of the control constraints.

Software and Technical Details: SQPlab is written in MATLAB and requires quadprog.m from the optimization toolbox. The user must provide a function simulator to evaluate the functions and gradients. A smaller version is also available that does not require quadprog.m but instead uses qpal.m, an augmented Lagrangian QP solver for medium-sized convex QPs. All solvers are distributed under the Q public license from INRIA, France (Gilbert, 2010).

9 Augmented Lagrangian Solvers

Augmented Lagrangian methods solve (6.1) by a sequence of subproblems that minimize the augmented Lagrangian, either subject to a linearization of the constraints or as a bound-constrained problem.

9.1 ALGENCAN

Algorithmic Methodology: ALGENCAN is a solver based on an augmented Lagrangian-type algorithm (Andreani et al., 2007, 2008) in which a bound-constrained problem is solved in each iteration. The objective function in each iteration is the augmented Lagrangian of the original problem. This subproblem is solved by using a quasi-Newton method. The penalty on each constraint is increased if the previous iteration does not yield a better point.

Software and Technical Details: ALGENCAN is written in Fortran77, and interfaces are available for AMPL, C/C++, CUTEr, JAVA, MATLAB, Octave, Python, and R. The bound-constrained problem in each iteration is solved by using GENCAN, developed by the same authors. ALGENCAN and GENCAN are freely available under the GNU Public License from the TANGO project webpage (Martinez and Birgin, 2010).

9.2 GALAHAD

Algorithmic Methodology: GALAHAD (Gould et al., 2004b) contains a range of solvers for large-scale nonlinear optimization. It includes LANCELOT B, an augmented Lagrangian method with a nonmonotone descend condition; FILTRANE, a solver for feasibility problems based on a multi-dimensional filter (Gould et al., 2004a); and interior-point and active-set methods for solving large-scale quadratic programs (QPs). GALAHAD also contains a presolve routine for QPs (Gould and Toint, 2004) and other support routines.

Software and Technical Details: GALAHAD is a collection of thread-safe Fortran95 packages for large-scale nonlinear optimization. It is available in source form at (Gould et al., 2002). GALAHAD has links to CUTEr, and Fortran.

9.3 LANCELOT

Algorithmic Methodology: LANCELOT (Conn et al., 1992) is a large-scale implementation of a bound-constrained augmented Lagrangian method. It approximately solves a sequence of bound-constrained augmented Lagrangian problems by using a trust-region approach. Each trust-region subproblem is solved approximately: first, the solver identifies a Cauchy-point to ensure global convergence, and then it applies conjugate-gradient steps to accelerate the local convergence. LANCELOT provides options for a range of quasi-Newton Hessian approximations suitable for large-scale optimization by exploiting the group-partial separability of the problem functions.

Software and Technical Details: LANCELOT is written in standard ANSI Fortran77 and has been interfaced to CUTEr (Bongartz et al., 1995) and AMPL. The distribution includes installation scripts for a range of platforms in single and double precision. LANCELOT is available freely from Rutherford Appleton Laboratory, UK (Conn et al., 2010).

9.4 MINOS

Algorithmic Methodology: MINOS (Murtagh and Saunders, 1998) uses a projected augmented Lagrangian for solving general nonlinear problems. In each "major iteration" a linearly constrained nonlinear problem is solved where the linear constraints constitute all the linear constraints of (6.1) and also linearizations of nonlinear constraints. This problem is in turn solved iteratively by using a reduced-gradient algorithm along with a quasi-Newton algorithm. The quasi-Newton algorithm provides a search direction along which a line search is performed to improve the objective function and reduce the infeasibilities. MINOS does not guarantee convergence from any starting point. The user should therefore specify a starting point that is "close enough" for convergence. The user can also modify other parameters such as the penalty parameter in augmented Lagrangian to control the algorithm.

Software and Technical Details: MINOS is implemented in Fortran. The library can be called from Fortran and C programs and interfaces such as AMPL, GAMS, AIMMS, and MATLAB. The user must input routines for the function and gradient evaluations. If a gradient is not available, MINOS estimates it by using finite differences. MINOS is available under commercial and academic licenses from Stanford Business Software Inc.

9.5 PENNON

Algorithmic Methodology: PENNON (Kocvara and Stingl, 2003) is an augmented Lagrangian penalty-barrier method. In addition to standard NCOs, it can handle semidefinite NCOs that *include a semidefiniteness constraint on matrices of variables*. PENNON represents the semidefiniteness constraint by computing an eigenvalue decomposition and adds a penalty-barrier for each eigenvalue to the augmented Lagrangian. It solves a sequence of unconstrained optimization problems in which the inequality constraints appear in barrier functions and the equality constraints in penalty functions. Every unconstrained minimization problem is solved with Newton's method. The solution of this problem provides a search direction along which a suitable merit function is minimized. Even though the algorithm has not been shown to converge for nonconvex problems, it has been reported to work well for several problems.

Software and Technical Details: PENNON can be called from either AMPL or MATLAB. The user manual (Kocvara and Stingl, 2008) provides instructions for input of matrices in sparse format when using AMPL. PENNON includes both sparse factorizations using the algorithm in (Ng and Peyton, 1993) and dense factorization using LAPACK. Commercial licenses and a free academic license are available from PENOPT-GbR.

10 Termination of NCO Solvers

Solvers for NCOs can terminate in a number of ways. Normal termination corresponds to a KKT point, but solvers can also detect (locally) infeasible NCOs and unboundedness. Some solvers also detect failure of constraint qualifications. Solvers also may terminate because of errors, and we provide some simple remedies.

10.1 Normal Termination at KKT Points

Normal termination corresponds to termination at an approximate KKT point, that is, a point at which the norm of the constraint violation, the norm of the first-order necessary conditions, and the norm of the complementary slackness condition, (2.1), are less than a user-specified tolerance. Some solvers divide the first-order conditions of (2.1) by the modulus of the largest multiplier. If the problem is convex (and feasible), then the solution corresponds to a global minimum of (6.1). Many NCO solvers include sufficient decrease conditions that make it less likely to converge to local maxima or saddle points if the problem is nonconvex.

10.2 Termination at Other Critical Points

Unlike linear programming solvers, NCO solvers do not guarantee global optimality for general nonconvex NCOs. Even deciding whether a problem is feasible or unbounded corresponds to a global optimization problem. Moreover, if a constraint qualification fails to hold, then solvers may converge to critical points that do not correspond to KKT points.

Fritz-John (FJ) Points. This class of points corresponds to first-order points at which a constraint qualification may fail to hold. NCO solvers adapt their termination criterion by dividing the stationarity condition by the modulus of the largest multiplier. This approach allows convergence to FJ points that are not KKT points. We note that dividing the first-order error is equivalent to scaling the objective gradient by a constant $0 \leq \alpha \leq 1$. As the multipliers diverge to infinity, this constant converges to $\alpha \to 0$, giving rise to an FJ point.

Locally Inconsistent Solutions. To prove that an NCO is locally inconsistent requires the (local) solution of a feasibility problem, in which a norm of the nonlinear constraint residuals is minimized subject to the linear constraints; see (2.2). A KKT point of this problem provides a certificate that (6.1) is locally inconsistent. There are two approaches to obtaining this certificate. Filter methods switch to a feasibility restoration phase if either the stepsize becomes too small or the LP/QP subproblem becomes infeasible. Penalty function methods do not switch but drive the penalty parameter to infinity.

Unbounded Solutions. Unlike the case of linear programming where a ray along the direction of descent is necessary and sufficient to prove that the instance is unbounded, it is difficult to check whether a given nonlinear program is unbounded. Most NCO solvers use a user supplied lower bound on the objective and terminate if they detect a feasible point with a lower objective value than the lower bound.

10.3 Remedies If Things Go Wrong

If a solver stops at a point where the constraint qualifications appears to fail (indicated by large multipliers) or where the nonlinear constraints are locally inconsistent or at a local and not global minimum, then one can restart the solver procedure from a different initial point. Some solvers include automatic random restarts.

Another cause for failure is errors in the function, gradient, or Hessian evaluation (e.g., IEEE exceptions). Some solvers provide heuristics that backtrack if IEEE exceptions are encountered during the iterative process. In many cases, IEEE exceptions occur at the initial point, and backtracking cannot be applied. It is often possible to reformulate the nonlinear constraints to avoid IEEE exceptions. For example, $\log(x_1^3 + x_2)$ will cause IEEE exceptions if $x_1^3 + x_2 \leq 0$. Adding the constraint $x_1^3 + x_2 \geq 0$ does not remedy this problem, because nonlinear constraints may not be satisfied until the limit. A better remedy is to introduce a nonnegative slack variable, $s \geq 0$, and

the constraint $s = x_1^3 + x_2$ and then replace $\log(x_1^3 + x_2)$ by $\log(s)$. Many NCO solvers will honor simple bounds (and interior-point solvers guarantee $s > 0$), so this formulation avoids some IEEE exceptions.

11 Calculating Derivatives

All solvers described above expect either the user or the modeling environment (AMPL, GAMS, etc.) to provide first-order and sometimes second-order derivatives. Some solvers can estimate first-order derivatives by finite differences. If the derivatives are not available or if their estimates are not reliable, one can use several automatic differentiation (AD) tools that are freely available. We refer the readers to Griewank (2000) for principles and methods of AD and to Moré (2000) for using AD in nonlinear optimization. AD tools include ADOL-C (Griewank and Walther, 2004); ADIC, ADIFOR, and OpenAD (Hovland et al., 2009); and Tapenade (Hascoët and Pascual, 2004).

12 Web Resources

In addition to a range of NCO solvers that are available online, there exist an increasing number of web-based resources for optimization. The NEOS server for optimization (Czyzyk et al., 1998; NEOS, 2003) allows users to submit optimization problems through a web interface, using a range of modeling languages. It provides an easy way to try out different solvers. The NEOS wiki (Leyffer and Wright, 2008) provides links to optimization case studies and background on optimization solvers and problem classes. The COIN-OR project (COINOR, 2009) is a collection of open-source resources and tool for operations research, including nonlinear optimization tools (IPOPT and ADOL-C). A growing list of test problems for NCOs includes the CUTEr collection (Gould et al., 2010), Bob Vanderbei's collection of testproblems (Vanderbei, 2010b), the COPS testset (Dolan et al., 2010), and the GAMS model library (GAMS, 2010). The NLP-FAQ (Fourer, 2010) provides online answers to questions on nonlinear optimization.

Acknowledgments

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. This work was also supported by the U.S. Department of Energy through grant DE-FG02-05ER25694.

References

Andreani, R., Birgin, E., Martinez, J., and Schuverdt, M. (2007). On augmented Lagrangian methods with general lower-level constraints. *SIAM Journal on Optimization*, 18:1286–1309.

Andreani, R., Birgin, E., Martinez, J., and Schuverdt, M. (2008). Augmented Lagrangian methods under the constant positive linear dependence constraint qualification. *Mathematical Programming*, 111:5–32.

Bertsekas, D. (1996). *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, Belmont, Mass.

Boggs, P. and Tolle, J. (1995). Sequential quadratic programming. *Acta Numerica*, 4:1–51.

Bongartz, I., Conn, A. R., Gould, N. I. M., and Toint, P. L. (1995). CUTE: Constrained and Unconstrained Testing Enviroment. *ACM Transactions on Mathematical Software*, (21):123–160. http://cuter.rl.ac.uk/cuter-www/interfaces.html.

Bonnans, J., Gilbert, J., Lemaréchal, C., and Sagastizábal, C. (2006). *Numerical Optimization: Theoretical and Practical Aspects*. Springer, Berlin, 2nd edition.

Byrd, R. H., Gilbert, J. C., and Nocedal, J. (2000). A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89:149–185.

Byrd, R. H., Gould, N. I. M., Nocedal, J., and Waltz, R. A. (2004). An algorithm for nonlinear optimization using linear programming and equality constrained subproblems. *Mathematical Programming, Series B*, 100(1):27–48.

Byrd, R. H., Hribar, M. E., and Nocedal, J. (1999). An interior point algorithm for large scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900.

Byrd, R. H., Nocedal, J., and Waltz, R. A. (2006). Knitro: An integrated package for nonlinear optimization. In di Pillo, G. and Roma, M., editors, *Large-Scale Nonlinear Optimization*, pages 35–59. Springer-Verlag.

Chin, C. and Fletcher, R. (2003). On the global convergence of an SLP-filter algorithm that takes EQP steps. *Mathematical Programming*, 96(1):161–177.

COINOR (2009). COIN-OR: Computational infrastructure for operations research. http://www.coin-or.org/.

Conn, A. R., Gould, N. I. M., and Toint, P. L. (1991). A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal of Numerical Analysis*, 28(2):545–572.

Conn, A. R., Gould, N. I. M., and Toint, P. L. (1992). *LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A)*. Springer Verlag, Heidelberg.

Conn, A. R., Gould, N. I. M., and Toint, P. L. (2010). LANCELOT webpage: http://www.cse.scitech.ac.uk/nag/lancelot/lancelot.shtml.

Czyzyk, J., Mesnier, M., and Moré, J. (1998). The NEOS server. *IEEE Journal on Computational Science and Engineering*, 5:68–75. Try it at www-neos.mcs.anl.gov/neos/!

Dahl, J. and Vandenberghe, L. (2010a). CVXOPT user's guide. Available online at http://abel.ee.ucla.edu/cvxopt/userguide/.

Dahl, J. and Vandenberghe, L. (2010b). CVXOPT website: http://abel.ee.ucla.edu/cvxopt.

Dirkse, S. and Ferris, M. (1995). The PATH solver: A non-monotone stabilization scheme for mixed complementarity problems. *Optimization Methods and Software*, 5:123–156.

Dirkse, S. and Ferris, M. (2010a). PATH-AMPL binaries: ftp://ftp.cs.wisc.edu/math-prog/solvers/path/ampl/.

Dirkse, S. and Ferris, M. (2010b). *PATHNLP*. GAMS. http://www.gams.com/dd/docs/solvers/pathnlp.pdf.

Dolan, E. D., Moré, J., and Munson, T. S. (2010). COPS large-scale optimization problems: http://www.mcs.anl.gov/~more/cops/.

Drud, A. (1985). A GRG code for large sparse dynamic nonlinear optimization problems. *Mathematical Programming*, 31:153–191.

Drud, A. (2007). *CONOPT*. ARKI Consulting and Development, A/S, Bagsvaerd, Denmark. http://www.gams.com/dd/docs/solvers/conopt.pdf.

Ferris, M. and Munson, T. (1999). Interfaces to PATH 3.0: Design, implementation and usage. *Computational Optimization and Applications*, 12:207–227.

Fiacco, A. V. and McCormick, G. P. (1990). *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Number 4 in Classics in Applied Mathematics. SIAM. Reprint of the original book published in 1968 by Wiley, New York.

Fletcher, R. (1987). *Practical Methods of Optimization*. John Wiley & Sons, Chichester.

Fletcher, R. (1999). Stable reduced Hessian updates for indefinite quadratic programming. Numerical Analysis Report NA/187, University of Dundee, Department of Mathematics, Scotland, UK.

Fletcher, R. and de la Maza, E. S. (1989). Nonlinear programming and nonsmooth optimization by successive linear programming. *Mathematical Programming*, 43:235–256.

Fletcher, R. and Leyffer, S. (1998). User manual for filterSQP. Numerical Analysis Report NA/181, University of Dundee.

Fletcher, R. and Leyffer, S. (2002). Nonlinear programming without a penalty function. *Mathematical Programming*, 91:239–270.

Forsgren, A., Gill, P. E., and Wright, M. H. (2002). Interior methods for nonlinear optimization. *SIAM Review*, 4(4):525–597.

Fourer, R. (2010). Nonlinear programming frequently asked questions: http://wiki.mcs.anl.gov/NEOS/index.php/Nonlinear_Programming_FAQ.

Friedlander, M. P. and Saunders, M. A. (2005). A globally convergent linearly constrained lagrangian method for nonlinear optimization. *SIAM Journal on Optimization*, 15(3):863–897.

GAMS (2010). The GAMS model library index: http://www.gams.com/modlib/modlib.htm.

Gilbert, J. C. (2009). SQPlab A MATLAB software for solving nonlinear optimization problems and optimal control problems. Technical report, INRIA-Rocquencourt, BP 105, F-78153 Le Chesnay Cedex, France.

Gilbert, J. C. (2010). SQPlab website: http://www-rocq.inria.fr/~gilbert/modulopt/optimization-routines/sqplab/sqplab.html.

Gill, P., Murray, W., and Saunders, M. (2006a). User's guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming. Report, Dept. of Mathematics, University of California, San Diego.

Gill, P., Murray, W., and Saunders, M. (2006b). User's guide for SQOPT Version 7: Software for Large-Scale Nonlinear Programming. Report, Dept. of Mathematics, University of California, San Diego.

Gill, P., Murray, W., Saunders, M., and Wright, M. (1998). User's guide for NPSOL Version 5.0: A fortran package for nonlinear programming. Report SOL 86-1, Dept. of Mathematics, University of California, San Diego.

Gould, N., Orban, D., and Toint, P. (2010). CUTEr webpage: http://cuter.rl.ac.uk/cuter-www/.

Gould, N. I. M., Leyffer, S., and Toint, P. L. (2004a). A multidimensional filter algorithm for nonlinear equations and nonlinear least squares. *SIAM Journal on Optimization*, 15(1):17–38.

Gould, N. I. M., Orban, D., and Toint, P. L. (2002). *GALAHAD*. Rutherford Appleton Laboratory. http://galahad.rl.ac.uk/.

Gould, N. I. M., Orban, D., and Toint, P. L. (2004b). GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. *ACM Trans. Math. Software*, 29(4):353–372.

Gould, N. I. M. and Robinson, D. P. (2008). A second derivative SQP method: Local convergence. Numerical Analysis Report 08/21, Oxford University Computing Laboratory. To appear in SIAM Journal on Optimization.

Gould, N. I. M. and Robinson, D. P. (2010). A second derivative SQP method: Global convergence. *SIAM Journal on Optimization*, 20(4):2023–2048.

Gould, N. I. M. and Toint, P. L. (2004). Presolving for quadratic programming. *Mathematical Programming*, 100(1):95132.

Gould, N. I. M. and Toint, P. L. (2010). Nonlinear programming without a penalty function or a filter. *Mathematical Programming*, 122(1):155–196.

Griewank, A. (2000). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia.

Griewank, A., Juedes, D., Mitev, H., Utke, J., Vogel, O., and Walther, A. (1996). ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. *ACM TOMS*, 22(2):131–167.

Griewank, A. and Walther, A. (2004). ADOL-C a package for automatic differentiation of algorithms written in C/C++. https://projects.coin-or.org/ADOL-C.

Griewank, A., Walther, A., and Korzec, M. (2007). Maintaining factorized KKT systems subject to rank-one updates of Hessians and Jacobians. *Optimization Methods Software*, 22(2):279–295.

Griffith, R. and Stewart, R. (1961). A nonlinear programming technique for the optimization of continuous processing systems. *Management Science*, 7(4):379–392.

Han, S. (1977). A globally convergent method for nonlinear programming. *Journal of Optimization Theory and Applications*, 22(3):297–309.

Hascoët, L. and Pascual, V. (2004). TAPENADE 2.1 user's guide. Technical Report 0300, INRIA.

Hovland, P., Lyons, A., Narayanan, S. H. K., Norris, B., Safro, I., Shin, J., and Utke, J. (2009). Automatic differentiation at Argonne. http://wiki.mcs.anl.gov/autodiff/.

Kocvara, M. and Stingl, M. (2003). PENNON—A code for convex nonlinear and semidefinite programming. *Optimization Methods and Software*, 18(3):317–333.

Kocvara, M. and Stingl, M. (2008). PENNON user's guide (version 0.9). Available at http://www.penopt.com.

Leyffer, S. and Wright, S. (2008). *NEOS wiki*. Argonne National Laboratory. http://wiki.mcs.anl.gov/NEOS/.

LINDO Systems Inc. (2010a). LINDO API 6.0 user manual.

LINDO Systems Inc. (2010b). Webpage. http://www.lindo.com/.

Mangasarian, O. (1969). *Nonlinear Programming*. McGraw-Hill Book Company, New York.

Maratos, N. (1978). *Exact penalty function algorithms for finite dimensional and control optimization problems*. Ph.D. thesis, University of London.

Martinez, J. and Birgin, E. (2010). TANGO: Trustable algorithms for nonlinear general optimization. Webpage http://www.ime.usp.br/~egbirgin/tango/index.php.

Moré, J. (2000). Automatic differentiation tools in optimization software. Technical Report ANL/MCS-P859-1100, Mathematics and Computer Science Division, Argonne National Laboratory.

Moré, J. J. and Toraldo, G. (1991). On the solution of quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, 1(1):93–113.

Murtagh, B. and Saunders, M. (1998). MINOS 5.4 user's guide. Report SOL 83-20R, Department of Operations Research, Stanford University.

NEOS (2003). The NEOS server for optimization. Webpage, Argonne National Laboratory. http://www-neos.mcs.anl.gov/neos/.

Nesterov, Y. and Nemirovskii, A. (1994). *Interior Point Polynomial Algorithms in Convex Programming*. Number 13 in Studies in Applied Mathematics. SIAM.

Ng, E. and Peyton, B. (1993). Block sparse Cholesky algorithms on advanced uniprocessor computers. *SIAM Journal on Scientific Computing*, 14(5):1034–1056.

Nocedal, J. and Wright, S. (1999). *Numerical Optimization*. Springer, New York.

Powell, M. (1978). A fast algorithm for nonlinearly constrained optimization calculations. In Watson, G., editor, *Numerical Analysis, 1977*, pages 144–157. Springer–Verlag, Berlin.

Schenk, O., Waechter, A., and Hagemann, M. (2007). Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization. *Journal of Computational Optimization and Applications*, 36(2-3):321–341. http://www.pardiso-project.org/.

Schittkowski, K. (1985). NLPQL: A Fortran subroutine for solving constrained nonlinear programming problems. *Annals of Operations Research*, 5(2):485–500.

Schittkowski, K. (2009). NLPQLP: A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line search – User's guide, Version 3.1. Report, Department of Computer Science, University of Bayreuth. http://www.math.uni-bayreuth.de/~kschittkowski/nlpqlp.htm.

Vanderbei, R. (2010a). LOQO homepage: http://www.princeton.edu/~rvdb/loqo/LOQO.html.

Vanderbei, R. (2010b). Nonlinear optimization models: http://www.orfe.princeton.edu/~rvdb/ampl/nlmodels/.

Vanderbei, R. and Shanno, D. (1999). An interior point algorithm for nonconvex nonlinear programming. *COAP*, 13:231–252.

Wächter, A. and Biegler, L. (2005a). Line search filter methods for nonlinear programming: Local convergence. *SIAM Journal on Optimization*, 16(1):32–48.

Wächter, A. and Biegler, L. (2005b). Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM Journal on Optimization*, 16(1):1–31.

Wächter, A. and Biegler, L. (2010). IPOPT project homepage: https://projects.coin-or.org/Ipopt.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

Waltz, R. and Plantenga, T. (2009). KNITRO user's manual. Available online at http://ziena.com/documentation.htm.

Wright, M. (1992). Interior methods for constrained optimization. *Acta Numerica*, 1:341–407.